CCNP ENCOR 350-401 & More

Networking, Automation, Network Security, Network Assurance and more.

Table of Contents

Spanning Tree Protocol (STP) Overview

Root Bridge Election

- The Root Bridge is elected based on the lowest **Bridge ID (BID = Priority + MAC Address)**.
- BPDUs determine the election, with superior BPDUs overriding others.

Root Port Selection (Per Switch)

1. Port with the **lowest path cost** to the Root Bridge.
2. If tied, lowest **neighbor BID**.
3. If still tied, lowest **neighbor Port ID**.
4. Finally, lowest **local Port ID**.

Designated Port Selection (Per Link)

- The switch port with the **lowest cost to the Root Bridge** becomes the Designated Port.
- If costs are equal, the switch with the **lowest BID** wins.
- If still tied, the lowest **Port ID** is chosen.

Alternate & Backup Ports

- **Alternate Port**: Becomes Root Port if the primary Root Port fails (used in RSTP).
- **Backup Port**: Standby for Designated Port, only applies when multiple links exist between two switches.

STP Port States

1. **Blocking** (Stable) – No forwarding, only BPDUs received.
2. **Listening** (15 sec) – No MAC learning, BPDUs processed.
3. **Learning** (15 sec) – MAC learning enabled, no frame forwarding.
4. **Forwarding** (Stable) – Full operation, frame forwarding enabled.
5. **Disabled** – Manually shut down or in an error-disabled state.

Timers

- **Hello Timer:** BPDU transmission interval (Default: 2 sec).
- **Forward Delay:** Time spent in Listening & Learning (Default: 15 sec each).
- **Max Age:** Maximum BPDU retention without update (Default: 20 sec).

Topology Change Notification (TCN)

- Sent only by **Root & Designated Ports** when topology changes.
- Non-designated and non-root ports **do not generate TCNs**.
- The Root Bridge never originates TCNs but processes them.

STP Optimization & Enhancements

STP Tuning Methods

- **Root Bridge Selection:** Adjust priority to control Root election.
- **Path Selection:** Modify **cost**, **port priority**, or **BID**.
- **Cost Modification:** Prefer direct cost adjustment over bandwidth changes.

Cisco STP Toolkit

- **UplinkFast:** (For access switches) Immediately transitions an alternate port to Forwarding when a Root Port fails. Suppresses TCNs.
- **BackboneFast:** (For upstream failures) Speeds up convergence by ignoring Max Age when receiving inferior BPDUs. Uses **Root Link Query (RLQ)** to verify the Root Bridge.

Key Differences

- **UplinkFast:** Bypasses Forward Delay.
- **BackboneFast:** Bypasses Max Age.

Direct Topology Change - 30 Seconds (Forwarding Timer)

Indirect Topology Change – 30 Seconds (Forwarding Timer) + 20 Seconds (MaxAge)

Summary of PortFast, BPDU Guard, BPDU Filter, and Errdisable:

PortFast: Allows a port to immediately transition to forwarding state, bypassing STP listening and learning phases. It is meant for edge devices (PCs, servers). Although it receives BPDUs, receiving one may disable the port depending on platform behavior. It can be used on trunks in cases like Router-on-a-Stick or servers with multiple VLANs.

BPDU Guard: Used to protect PortFast-enabled ports. If a BPDU is received, the port is placed into **errdisable** state, preventing accidental loops.

BPDU Filter:

Per-port: Disables STP entirely (no BPDU send/receive).

Global mode: Blocks outgoing BPDUs but reverts to normal STP mode if a BPDU is received.

Errdisable: A state where a port is shut down due to an error (e.g., BPDU Guard triggering). Recovery is manual (shutdown/no shutdown) or automatic with a timer.

RSTP Summary

Fundamentals & Similarities with STP

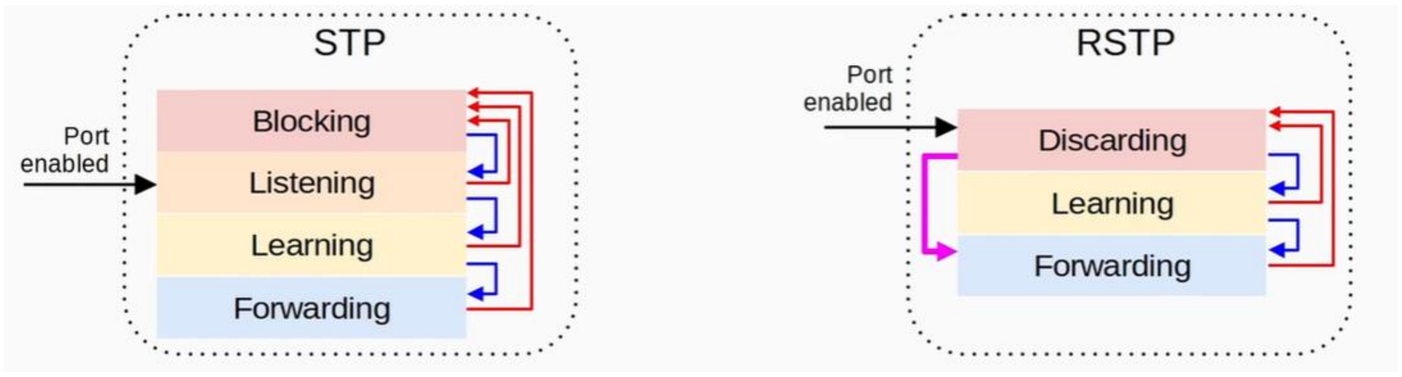
- Root bridge, root port, and designated/non-designated port selection remain unchanged.
- Tuning is done via **Bridge ID, port cost, and port priority**.
- Features like **BackboneFast and UplinkFast** are built into RSTP, while **PortFast, BPDU Guard, and BPDU Filter** function the same.

RSTP Port States

- **Discarding**: Combines STP's Listening and Blocking states. BPDUs are processed but no frames are forwarded. Stable for Alternative/Backup ports.
- **Learning**: Similar to STP's Learning state. MAC addresses are learned, but frames aren't forwarded. Occurs when RSTP sync fails.
- **Forwarding**: Normal operational mode.

Alternate vs. Backup Ports

- **Alternate Port**: A redundant port on a non-root switch pointing toward the root bridge. Acts as a backup **Root Port**.
- **Backup Port**: Exists when two ports on the same switch are connected to the same collision domain (e.g., a hub), providing redundancy for a **Designated Port**.



RSTP Link Types

- **P2P (Full Duplex)**: Direct switch-to-switch connections.
- **Shared (Half Duplex)**: Hubs and legacy network segments.
- **Edge**: End-user devices. (Equivalent to STP's PortFast)

RSTP Sync Process (Key to Fast Convergence)

- Each switch starts with all designated ports in Discarding.
- BPDUs are sent with the **Proposal Bit**, each switch trying to become the **Designated Switch** for the segment.
- A switch receiving a **Superior BPDU with a Proposal Bit** accepts it and syncs by making all non-edge ports **Discarding**.
- The switch responds with an **Agreement BPDU**, confirming the superior switch as the Designated Bridge.
- Ports transition quickly to **Forwarding**, significantly reducing convergence time compared to STP.

Link speed	STP/RSTP Short Path Cost	RSTP Long Path
10 M	100	2,000,000
100 M	19	200,000
1 G	4	20,000
10 G	2	2,000
100 G	N/A	200
1 T	N/A	20

Command to show and change cost method -

Show spanning-tree pathcost method

Spanning-tree pathcost method long

Multiple Spanning Tree (MST)

Why MST?

- Scalability Issue with PVST+/RSTP: Running a separate STP instance per VLAN (e.g., 100 VLANs → 100 BPDUs) is inefficient.
- MST Solution: Groups multiple VLANs into a smaller set of MST instances (MSTIs), reducing overhead. Each MSTI runs RSTP, offering fast convergence.

Key MST Components

1. **MST Region**: A group of switches with identical MST name, revision, and VLAN-to-instance mapping.
2. **IST (MST0)**: The Internal Spanning Tree instance that represents the entire MST region to external (non-MST) links. This instance is also responsible for sending BPDUs.
3. **MSTI**: Additional MST instances (e.g., MST1, MST2, etc.). Each MSTI can have its own root bridge, enabling load balancing by assigning different VLANs to different MSTIs.
4. **CST**: The Common Spanning Tree across the entire Layer 2 network, including legacy STP domains or multiple MST regions.
5. **CIST**: The Common and Internal Spanning Tree. Combines the CST outside the region with the IST inside the region.

Regional Root vs. CIST Root

1. **CIST Root**: The switch with the lowest bridge ID across **all** MST regions and any legacy STP domains.
2. **Regional Root**: Within each MST region, the switch with the **lowest cost** (external) reach the CIST Root is elected as the region's **regional root**.
   External Cost: Path cost from the region's boundary ports to the CIST Root **outside** the region.
   Internal Cost: Path cost **inside** the region from each switch to that boundary port.

Boundary Ports

1. **Definition**: Ports on MST switches connecting to either a different MST region or legacy STP domain.
2. **Behavior**: These ports handle external MST BPDUs or 802.1D BPDUs. Inside the MST region, ports that connect internal MST switches are considered regular (non-boundary) ports

MST Configuration Essentials

MST Mode:

```
spanning-tree mode mst
```

MST Region Definition (must match on all region switches):

```
spanning-tree mst configuration
  name <REGION_NAME>
  revision <NUMBER>
  instance 1 vlan 1-50
  instance 2 vlan 51-100
```

Setting Root Priority:

```
mst 1 priority 4096     # Lower=Root
spanning-tree mst 2 priority 8192
```

1. **VLAN-to-Instance Mapping**:
   Each MST instance covers one or more VLANs.
   MST0 (IST) typically mapped to VLANs **outside** your main operational range (e.g., 101–4094), but best practices vary.

2. **Port-Level Configuration**:
   spanning-tree mstp on trunk interfaces if required.
   PortFast & **BPDU Guard** for access ports.
   Root Guard on trunk ports where you don't expect a superior BPDU.

Load Balancing

By defining multiple instances and adjusting priorities, you can ensure different switches are root for different MSTIs.
VLANs mapped to MSTI 1 use one root switch, VLANs in MSTI 2 use another, helping optimize traffic flow.

Verification & Troubleshooting

Check MST Region Consistency:

```
show spanning-tree mst configuration
```

Instance Roles and States:

```
show spanning-tree mst
 show spanning-tree mst interface <interface> detail
```

Look for correct Root/Alternate/Designated states per MSTI and validate that boundary ports are recognized properly.

Crucial Points

Name, Revision, and VLAN Mapping must match exactly for switches to be in the same region.

MST inherits the rapid convergence features from RSTP.

The Regional Root is chosen based on the total path cost (external + internal) to the CIST Root.

Boundary Ports are pivotal in connecting MST regions or legacy STP areas, ensuring correct forwarding and root calculations.

Bottom Line: MST is a powerful way to reduce overhead and implement efficient, load-balanced spanning trees at scale. Mastering region definitions, root priorities, boundary ports, and cost manipulation will ensure a stable, optimized Layer 2 topology.

VTP (VLAN Trunking Protocol) Summary

VLAN Deletion & Revision Number Impact

Deleting vlan.dat resets the revision number to zero.
If performed on a VTP server, this does not affect clients, as they ignore updates with a lower or zero revision number.
Best practice: If vlan.dat is deleted, it should be done only on clients to avoid unintended consequences.

Impact of a Higher Revision Number Switch

A new or reintroduced switch with a higher revision number can overwrite all VLAN configurations across the domain.
This is a major security risk due to potential human error or a VTP attack.
Best practice: Before connecting a switch to a VTP domain, reset its VTP revision number (by changing the mode to Transparent and back).

VTP Message Processing Requirements

VTP updates are processed only if the following match:

Same VTP Domain Name
Same VTP Password (if configured)
Same VTP Version

Transparent and Off modes handle updates differently:

Transparent Mode: Forwards VTP messages **if the domain matches** but does not apply updates.
Off Mode: Ignores all VTP advertisements completely.

VTP Role & Revision Number Importance

Regardless of server or client mode, the switch with the highest revision number is the source of updates.
Client switches do not store VLANs in vlan.dat but will update their VLAN database from the highest revision advertisement.

Additional Best Practices for VTP Security & Stability

Use VTP Transparent Mode in production networks to prevent accidental VLAN overwrites.
Secure VTP with a strong password to prevent unauthorized changes.
Manually reset revision numbers before introducing a switch into an existing network.

Overview of EtherChannel

EtherChannel bundles multiple physical links into a single logical link to provide increased bandwidth and redundancy. It supports both Layer 2 (switching) and Layer 3 (routing) configurations.

Modes of EtherChannel

- **Static Mode (on)**: No negotiation, both sides must be set to on.
- **PAgP (Port Aggregation Protocol)** – Cisco Proprietary
  - desirable (actively negotiates)
  - auto (passively listens)
  - Requires both sides to be desirable or one desirable and the other auto.
- **LACP (Link Aggregation Control Protocol)** – IEEE 802.3ad Standard
  - active (actively negotiates)
  - passive (passively listens)
  - Requires at least one side to be active.

Load Balancing in EtherChannel

- Uses a **hashing algorithm** to distribute traffic.
- Can use **source/destination MAC, IP, or Layer 4 port numbers** for load balancing.
- Default is typically **src-dst-IP**.

Can be adjusted using:

```
port-channel load-balance {method}
```

- In networks with **low entropy** (e.g., few IPs), traffic may be unevenly distributed. Adjusting the method (e.g., using MAC addresses) may improve distribution.

EtherChannel Configuration & Verification

Basic LACP Configuration:

```
interface range GigabitEthernet1/0/1 - 2
  channel-group 1 mode active

interface Port-channel1
  switchport mode trunk
```

Checking EtherChannel Status:

```
show etherchannel summary
show etherchannel load-balance
show run interface port-channel X
```

Common Issues & Troubleshooting

Issue	Possible Cause	Troubleshooting Steps
Links not bundling	Mismatched EtherChannel modes (on vs. LACP/PAgP)	Verify using show etherchannel summary
Some links suspended	VLAN mismatch, trunk/access mode mismatch, speed/duplex mismatch	Use show run interface & show int trunk
Uneven traffic distribution	Hashing method not optimal	Change load balancing method
STP blocking an EtherChannel port	EtherChannel negotiation failed, treating links as separate	Verify using show spanning-tree
VLAN traffic not passing	VLAN not allowed, VLAN missing in database	Use show int trunk & show vlan brief
EtherChannel err-disabled	Port security applied to member interfaces	Remove port security from EtherChannel ports

EtherChannel & Multichassis Aggregation

- **VSS (Virtual Switching System)** and **StackWise Virtual** allow a switch stack to act as a single logical switch.
- **MEC (Multichassis EtherChannel)** supports bundling across different chassis in a stack.
- Traditional EtherChannel does not work across multiple physical switches unless VSS/Stack is used.

LACP & PAgP Timers

- **LACP Timer:**
  - Normal: **30 seconds**
  - Fast: **1 second** (lacp rate fast)
- **PAgP Timer:**
  - Normal: **30 seconds**
  - Fast: **1 second** (pagp timer fast)

Configuration Example:

```
interface Port-channel1
  lacp rate fast
```

Adding Links to an Active EtherChannel

1. Ensure the new interface has the **same configuration** as existing members.
2. Use channel-group to add the interface:
   ```
   interface Gi1/0/3
    channel-group 1 mode active
   ```
3. Verify the port joins the bundle (show etherchannel summary).

EtherChannel with Firewalls & Servers

- Firewalls/NIC teaming solutions may **limit the number of active links** in a bundle.
- If a firewall only supports **2 active links**, configure:
  ```
  lacp max-bundle 2
  ```
- Ensure the **server NIC teaming mode** matches the switch EtherChannel mode (Static, LACP, or PAgP).

Final Thoughts

EtherChannel is a powerful feature for increasing bandwidth and redundancy, but **proper configuration and verification are critical** to avoid issues like misconfigurations, VLAN mismatches, or STP blocking. **Using VSS or StackWise enables cross-switch aggregation**, ensuring better scalability.

Routing Fundamentals: A Comprehensive Overview

Routing is the core function of network infrastructure, enabling the forwarding of packets between different networks based on destination IP addresses. Routers utilize a **Routing Information Base (RIB)** and various packet-switching methods, such as **Cisco Express Forwarding (CEF), Fast Switching, and Process Switching**, to ensure efficient and scalable data transmission. The **longest prefix match (LPM)** algorithm determines the most specific route for forwarding decisions.

Types of Routes in the Routing Table

Directly Connected Routes

- Automatically added when an interface is configured with an IP address and is in an **up/up** state.
- These routes have an **Administrative Distance (AD) of 0**, making them the most preferred.
- **Example:**
  C 192.168.1.0/24 is directly connected, GigabitEthernet0/0

Static Routes

Static routes are manually configured by network administrators and have an **AD of 1**, making them highly reliable for predefined routing decisions. They are commonly used for **traffic engineering, network security, and redundancy**

Key Routing Concepts

- **Longest Prefix Match (LPM):** The router selects the most specific route (e.g., a **/30 route takes precedence over a /24**).
- **Recursive Lookup:** When a static route defines only a next-hop IP, the router must check the routing table to determine the correct exit interface.
- **Administrative Distance (AD):** A value used to rank route preference; lower AD values indicate higher trustworthiness.
- **First Hop Redundancy Protocols (FHRP):** Protocols such as **HSRP, VRRP, and GLBP** provide redundancy by allowing multiple routers to act as a default gateway.

Types of Static Routes:

1. **Recursive Static Route**
   a. Specifies **only the next-hop IP address**, requiring a recursive lookup to determine the exit interface.

ip route 192.168.2.0 255.255.255.0 192.168.1.2

2. **Attached Static Route**
   a. Specifies **only the exit interface** without a next-hop IP.
   b. **Not recommended for multi-access networks (Ethernet)** due to potential **ARP flooding** but is suitable for **point-to-point (P2P) links**.

ip route 192.168.3.0 255.255.255.0 Serial0/1/0

3. **Fully Specified Static Route**
   a. Defines **both the next-hop IP address and the exit interface** to eliminate recursive lookups and prevent ARP-related issues in multi-access networks.
   b. Recommended for Ethernet networks.

ip route 192.168.4.0 255.255.255.0 192.168.1.2 GigabitEthernet0/0

4. **Floating Static Route**
   a. Functions as a **backup route** and is assigned an **AD greater than the primary route**, ensuring it is used only in case of failure.

ip route 192.168.5.0 255.255.255.0 10.1.1.1 200

5. **Default Static Route**
   a. Acts as a **gateway of last resort**, forwarding traffic when no specific route exists in the routing table.

ip route 0.0.0.0 0.0.0.0 192.168.1.1

   b. **Example (Fully Specified):**

ip route 0.0.0.0 0.0.0.0 192.168.1.1 GigabitEthernet0/0

6. **Null Static Route**
   a. Used to **discard traffic** or prevent routing loops in route summarization.

ip route 192.168.6.0 255.255.255.0 Null0

7. **Summary Static Route**
   a. Aggregates multiple smaller subnets into a **single summarized route**, reducing the size of the routing table.

ip route 192.168.0.0 255.255.252.0 10.1.1.1

Best Practices for Static Routing

✔ **Use Fully Specified Static Routes** in multi-access networks (Ethernet) to prevent ARP-related issues.
✔ **Avoid Attached Static Routes on Ethernet**, as they can cause ARP flooding.
✔ **Deploy Floating Static Routes** with a higher AD to provide failover redundancy.
✔ **Utilize Null Static Routes** for traffic blackholing and preventing routing loops.
✔ **Implement Summary Static Routes** to optimize routing table size and efficiency.

Summary of Filtering Methods: Distribute Lists, Prefix Lists, and Route Maps

Network filtering and policy control can be achieved using **Distribute Lists (ACLs), Prefix Lists, and Route Maps**. Each method has its specific use case, offering different levels of granularity and flexibility.

Filtering Method	Function	Operation	Best Use Case
Distribute List	Permits or Denies Specific routes	Uses standard and extended ACLs to filter in or out routes	Simple network based filtering
Prefix List	Filters routes based on network and prefix length	Matches prefixes with specific subnet lengths	More granular control over prefixes
Route Maps	Advanced Filtering and route manipulation	Modification based on prefix, metric, next-hop	Traffic Engineering, policy based routing and redistribution

Distribute Lists (ACL-Based Filtering)

Distribute lists apply access control lists (ACLs) to permit or deny specific networks before they are processed.

Example: Allow Only 192.168.1.0/24

```
access-list 10 permit 192.168.1.0 0.0.0.255

router ospf 1
 distribute-list 10 in
```

- Only **192.168.1.0/24** will be accepted.
- All **other routes are implicitly denied**.

Pros: Simple configuration

Cons: Cannot filter based on prefix length, limited flexibility

Prefix Lists (More Efficient Filtering)

Prefix lists provide **better control over filtering** by allowing matches based on **prefix length**.

Example: Block 10.1.1.0/24

```
ip prefix-list BLOCK seq 5 deny 10.1.1.0/24

router bgp 65000
 distribute-list prefix BLOCK in
```

Example: Allow Only /24 to /30 Prefixes in a /16 Range

```
ip prefix-list ALLOW_NETS seq 10 permit 192.168.0.0/16 ge 24 le 30
```

Allows only **prefixes between /24 and /30** within 192.168.0.0/16.

Pros: More efficient filtering, supports prefix-length ranges

Cons: Cannot modify routes, only filters prefixes

Route Maps (Advanced Policy Control)

Route maps provide **the most flexibility**, allowing not only filtering but also **modifications** to route attributes.

Example: Block 10.1.1.0/24

```
route-map FILTER deny 10
  match ip address 10

router bgp 65000
  distribute-list route-map FILTER in
```

Example: Modify a Route Instead of Blocking

```
route-map MODIFY_METRIC permit 10
  match ip address 10
  set metric 200
```

Instead of blocking, this modifies the metric for 10.1.1.0/24, influencing routing decisions.

Pros: Can modify routes, supports multiple match conditions

Cons: More complex than ACLs or prefix lists

Distribute-lists are used to filter on incoming routes while Redistribute is used to leak Protocol specific route into other protocol’s table

Policy-Based Routing (PBR) Summary

PBR provides granular control over traffic by overriding the Routing Table (RIB) and CEF. It applies only to **ingress traffic** and redirects packets based on defined policies.

Configuration Steps

Define ACL for Matching Traffic

```
ip access-list extended PBR-TRAFFIC
  permit ip host 192.168.30.1 host 192.168.10.1
```

Create a Route-Map with Next-Hop Policy

```
route-map PBR-POLICY permit 10
  match ip address PBR-TRAFFIC
  set ip next-hop 1.1.1.13
```

(Optional fallback if next-hop is unreachable, Default means it'll use the RIB for operations like normal):

```
set ip default next-hop 1.1.1.13
```

Apply Route-Map on Ingress Interface

```
interface Ethernet0/0
  ip policy route-map PBR-POLICY
```

Verification Commands

```
show ip policy                  # Verify PBR is applied to an interface
show route-map                  # Check if traffic matches the policy
show access-lists PBR-TRAFFIC   # Verify ACL hit counts
```

Key Considerations

- **PBR applies only to ingress traffic, not locally generated packets.**
- **Overrides RIB but is still processed by CEF.**
- **Packets drop if next-hop is unreachable unless set ip default next-hop is used.**
- **Ensure bidirectional routing to prevent asymmetric paths.**

PBR is commonly used for **traffic steering, load balancing, and security enforcement.**

Virtual Routing and Forwarding (VRF) Summary

What is VRF?

VRF allows multiple **isolated routing tables (RIBs)** on the same router, enabling traffic segmentation for **multi-tenant environments, security, and routing control**.

Key Features

- **Multiple Routing Tables:** Each VRF has its own independent routes.
- **Traffic Isolation:** Packets in different VRFs do not mix.
- **Supports Static & Dynamic Routing:** VRFs can use OSPF, BGP, or static routes.
- **VRF Leaking (Optional):** Allows communication between VRFs when needed.

Configuration Steps

Define a VRF

```
ip vrf <VRF-NAME>
```

Assign an Interface to a VRF

```
interface GigabitEthernet0/0
 vrf forwarding <VRF-NAME>
 ip address <IP> <MASK>
```

(All existing interface configurations are removed when assigning a VRF.)

Configure Routing within the VRF

```
 router ospf 1 vrf <VRF-NAME>
```

(Or use static routes with ip route vrf.)

Verification Commands

```
show ip vrf                # Displays VRF instances
 show ip route vrf <VRF-NAME>   # Shows the routing table for a VRF
 show run interface <int>       # Checks VRF assignment on an interface
```

VRF Leaking (Inter-VRF Communication)

By default, **VRFs do not share routes**, but you can leak routes:

```
ip route vrf VRF1 192.168.2.0 255.255.255.0 10.1.1.1 global
```

The following commands, ensures VRF1 is leaked into the global RIB.

CEF

Cisco Express Forwarding (CEF) is an advanced Layer 3 IP switching technology that enhances packet forwarding efficiency and scalability in Cisco devices. It operates by utilizing a precomputed data structure instead of relying on CPU-intensive per-packet lookups.

Key Components of CEF:

1 – Forwarding Information Base (FIB):

- Acts as an optimized version of the routing table (RIB).
- Stores precomputed next-hop information for fast lookups.
- Updated dynamically in sync with routing protocol changes.

2 – Adjacency Table:

- Maintains Layer 2 next-hop information.
- Populated via ARP and other Layer 2 discovery mechanisms.
- Ensures efficient packet forwarding without redundant lookups.

CEF Switching Process:

1 – When a packet arrives, CEF uses the FIB to determine the best next-hop.

2 – The corresponding adjacency table entry provides Layer 2 information.

3 – The packet is forwarded without requiring slow, CPU-intensive route lookups.

Benefits of CEF:

- Improved Performance: Reduces CPU usage by precomputing forwarding paths.
- Scalability: Handles large-scale networks more efficiently than traditional switching methods.
- Deterministic Forwarding: Avoids process switching delays.
- Load Balancing: Supports per-destination and per-packet load sharing.

CEF Modes:

- Central CEF: Uses a single forwarding engine in hardware/software.
- Distributed CEF (dCEF): Spreads forwarding tasks across line cards for enhanced performance.

CEF is the default switching method on modern Cisco platforms, ensuring optimal IP packet forwarding with minimal processing overhead.

SDM (Switching Database Manager) Overview

SDM is a feature in Cisco switches that allows administrators to allocate TCAM (Ternary Content Addressable Memory) resources based on specific network requirements.

SDM Templates

Cisco switches provide different SDM templates, each optimized for different use cases. These templates vary based on:

- Platform: Different Cisco switch models support different templates.
- IOS Version: The available templates may differ across software versions.

Template Functionality

Each SDM template determines how TCAM memory is allocated for:

- IP unicast routing
- MAC address table
- IPv6-related addresses
- Multicast traffic
- QoS (Quality of Service)

For instance:

- LANBase-routing provides a balanced allocation of TCAM resources for IP unicast, MAC, IPv6, multicast, and QoS.
- Layer 2 Only Mode prioritizes MAC addresses over IP routing entries, making it ideal for purely Layer 2 switches.

Key Considerations

1 - Switch Reboot Required – Changing the SDM template requires a device restart to apply the new configuration.

2 - Feature Enablement/Removal – Some templates may disable certain features. For example, a template that reduces IPv6 allocations might prevent IPv6-related commands from functioning.

EIGRP Fundamentals

EIGRP Overview

- Type: Advanced Distance Vector (sometimes called a “hybrid” protocol).
- Protocol Number: 88 (EIGRP has its own protocol number, unlike OSPF which uses IP protocol 89).
- Transport Mechanism: Uses Reliable Transport Protocol (RTP), NOT TCP or UDP.
- Multicast Address: 224.0.0.10 for sending updates to neighbors.
- Port Used: EIGRP does NOT use TCP or UDP but operates directly over IP (Protocol 88).
- Metric Calculation: Uses Bandwidth and Delay by default (K1 & K3). Can also consider Load and Reliability (K2, K4, K5, but not used by default).
- Administrative Distance (AD):
  - Internal EIGRP: 90
  - External EIGRP (redistributed routes): 170
  - EIGRP Summary Routes: 5 (if manually summarized)

EIGRP Packet Types

EIGRP uses five different packet types:

1 - Hello Packets → Used for neighbor discovery and keepalives (sent via multicast 224.0.0.10).

2 - Update Packets → Contain route updates (sent multicast/unicast depending on topology).

3 - Query Packets → Sent when a route is lost to find an alternative path.

4 - Reply Packets → Sent in response to a Query (acknowledging the best available route).

5 - ACK Packets → Acknowledges receipt of Update, Query, and Reply packets (sent unicast).

EIGRP Neighborship & Metric Calculation

- Routers only forward the Advertised Distance (AD) (also called Reported Distance, RD) to neighbors.
- Each router calculates its own Feasible Distance (FD) by adding its own link cost to the received AD.
- Successor: Route with the lowest FD, installed in the RIB.
- Feasible Successor (FS): Backup route that meets the Feasibility Condition (FC).

Feasibility Condition (FC) - Ensuring Loop-Free Backup Routes

- Rule: A route can be an FS if RD < Successor’s FD.
- Why? Ensures the backup route is truly closer and avoids routing loops.
- Result: FS is stored in the EIGRP topology table and used for instant failover if the Successor fails.

EIGRP Load Balancing

- Equal-Cost Load Balancing: Automatically used when multiple Successors exist.
- Unequal-Cost Load Balancing: Enabled via the Variance command.
- Formula: Variance × Successor FD = Maximum Allowed FD
- Paths meeting this condition are added to forwarding.
- Variance does NOT affect FS selection, only traffic forwarding.

EIGRP Key Concepts Summary

Path Metric Calculation

- EIGRP uses Bandwidth (slowest link) and Delay (cumulative) by default.
- K-values and attributes:
- K1 = Bandwidth, K3 = Delay (default, used).
- K2 = Load, K4 = Reliability
- Changing metrics requires uniform configuration across routers.

Failure Detection and Timers

EIGRP vs. OSPF Timers:

- EIGRP Hello = 5 sec, Hold = 15 sec (3x Hello)
- OSPF Hello = 10 sec, Hold = 40 sec (4x Hello)
- Timers are per-interface, configured as:

```
ip hello-interval eigrp 1 5

ip hold-time eigrp 1 15
```

- DUAL (Diffusing Update Algorithm):
- Uses Feasible Successors for instant failover.
- If no backup, sends queries → replies with infinity metric.

Route Summarization

- Manual Summarization (on any interface):

```
ip summary-address eigrp 1 192.168.0.0 255.255.252.0
```

- Reduces routing table size.
- Creates a Null0 route for loop prevention.

Auto-Summary (Legacy, Disabled by Default):

```
router eigrp 1

 auto-summary
```

- Summarizes at Classful Boundaries (/8, /16, /24).

EIGRP Topology Table

EIGRP maintains a topology table, which is distinct from a standard distance vector protocol. It contains:

- Network prefix
- EIGRP neighbors that have advertised the prefix
- Metrics from each neighbor (Reported Distance, Feasible Distance)
- Values used for metric calculation (Bandwidth, Delay, Load, Reliability)

Each route in the topology table can be in one of two states:

- Passive (P) → The route is stable, and no recalculations are occurring.
- Active (A) → EIGRP is searching for an alternative path (triggered by a topology change).

Feasible Successor & Feasibility Condition

- Successor → The best path (lowest feasible distance).
- Feasible Successor → A backup path that satisfies the Feasibility Condition (the Reported Distance of the next hop must be less than the local Feasible Distance).

To check the topology table:

```
show ip eigrp topology
```

Example output:

```
P 10.1.1.0/24, 1 successors, FD is 3072

via 192.168.1.2 (3072/2816), FastEthernet0/0
```

- P = Passive (stable)
- FD (Feasible Distance) = 3072
- RD (Reported Distance) = 2816 (meets feasibility condition)

If no Feasible Successor exists, the router queries its neighbors.

EIGRP Route Filtering

Route filtering controls which prefixes are advertised or accepted from neighbors. This is useful for:

- Traffic engineering (manipulating flows)
- Security (limiting route distribution)
- Reducing memory usage (smaller routing tables)

Filtering Methods:

1 - Distribute Lists (uses ACLs to permit/deny routes)

```
access-list 10 permit 192.168.1.0 0.0.0.255

router eigrp 100

distribute-list 10 in
```

Limits incoming routes.

2 - Prefix Lists (matches prefixes more efficiently than ACLs)

```
ip prefix-list FILTER seq 5 deny 192.168.2.0/24

router eigrp 100

distribute-list prefix FILTER in
```

Controls which routes are learned.

3 - Route Maps (advanced filtering based on multiple conditions)

```
route-map EIGRP_FILTER deny 10

match ip address 10

router eigrp 100

distribute-list route-map EIGRP_FILTER in
```

More flexible than ACLs or prefix lists.

Summary of Route Redistribution

Route redistribution is used to share routes between different routing protocols or autonomous systems. This is necessary in multi-protocol environments, where some routers may not support newer protocols, or in cases where different parts of the network run separate routing domains.

Key Concepts:

- Redistribution can occur in both directions:
- From EIGRP to another protocol (e.g., OSPF, BGP, RIP)
- From another protocol into EIGRP
- EIGRP requires metric values for redistributed routes to calculate the best path.
- Redistribution must be controlled to avoid routing loops or suboptimal paths.

Redistributing Routes into EIGRP

When bringing routes from another protocol (e.g., RIP) into EIGRP, a metric must be provided:

```
router eigrp 1

 redistribute rip metric 10000 10 255 1 1500
```

- 10000 = Bandwidth (Kbps)
- 10 = Delay (in tens of microseconds)
- 255 = Reliability (highest value)
- 1 = Load (lowest value)
- 1500 = MTU (not used in metric calculation but required)

Without specifying a metric, routes will not be redistributed into EIGRP.

Redistributing EIGRP into Another Protocol

When moving routes from EIGRP into OSPF, the subnets keyword is required to redistribute both classful and subnetted networks:

```
router ospf 1

 redistribute eigrp 100 subnets
```

This ensures that all learned routes (including non-classful subnets) are advertised into OSPF.

Verifying Redistribution

To confirm the redistribution process, use:

```
show ip route eigrp

show ip eigrp topology
```

These commands verify that the redistributed routes appear in the EIGRP topology table and routing table

Load Balancing in EIGRP

By default, most routing protocols install only equal-cost routes into the routing table. However, EIGRP supports both equal-cost and unequal-cost load balancing.

Equal-Cost Load Balancing (ECMP)

- If EIGRP learns multiple paths with the same metric to a destination, all of them are installed in the routing table.
- The router then distributes traffic equally across these paths.
- The maximum number of equal-cost paths that EIGRP supports can be configured using:

```
router eigrp 100
 maximum-paths 4
```

- The default value is 4 paths.
- Can be configured up to 32 paths.

Unequal-Cost Load Balancing (Variance)

EIGRP is unique in its ability to perform unequal-cost load balancing through the variance command.

- Variance allows suboptimal paths to be installed in the routing table.
- The router calculates a variance multiplier based on the feasible successor's feasible distance (FD).
- The formula for allowing a backup route is:

Variance * FD (Successor) ≥ FD (Feasible Successor)

Any feasible successor route that meets this condition will be installed and used proportionally.

Example: Configuring Unequal-Cost Load Balancing

```
router eigrp 100
 variance 2
```

This means that any feasible successor whose metric is less than or equal to twice the best path metric will be used

Traffic Share Distribution

- The router will distribute traffic proportionally based on the route metrics.
- Example output showing traffic share counts:

```
R1# show ip route eigrp
D   10.3.3.0/24 [90/3328] via 10.14.1.4, 120 packets
            [90/5632] via 10.12.1.2, 71 packets
```

Summary of EIGRP Authentication and Security

The primary purpose of EIGRP authentication and security is to prevent unauthorized routers from joining an EIGRP topology. This ensures that only trusted devices exchange routing information, reducing the risk of route manipulation or spoofing attacks.

Authentication Methods

EIGRP supports two types of authentication:

1 - MD5 Authentication – Older, widely supported method.

2 - HMAC-SHA-256 Authentication – More secure, recommended for modern deployments.

Configuring MD5 Authentication

MD5 authentication uses a key chain to manage shared secrets.

Step 1: Create a Key Chain

```
key chain EIGRP_AUTH
 key 1
 key-string SecureKey123
```

- The key ID and key-string must match between routers.
- Keys are case-sensitive.

Step 2: Apply to an Interface

```
interface Ethernet0/0
 ip authentication key-chain eigrp 100 EIGRP_AUTH
 ip authentication mode eigrp 100 md5
```

This forces authentication for all EIGRP packets on this interface.

Step 3: Ensure NTP Synchronization

```
ntp server 192.168.1.1
clock timezone UTC 0
```

if routers are using time-based key rotation, NTP must be configured to prevent mismatches.

Configuring HMAC-SHA-256 Authentication (Named Mode)

For modern deployments, HMAC-SHA-256 is preferred over MD5.

```
router eigrp SECURE
 address-family ipv4 unicast autonomous-system 100
  af-interface Ethernet0/0
   authentication mode hmac-sha-256 StrongPass123
```

No key chain is required; the key is entered directly.

Verification and Troubleshooting

To check neighbor authentication status:

```
show ip eigrp neighbors
```

To confirm authentication settings:

```
show ip eigrp interface detail | include Authen
```

If authentication fails, routers will not form EIGRP adjacencies.

For troubleshooting:

```
debug eigrp packets authentication
```

Key Considerations

- Case-sensitive keys must match exactly.
- Authentication is per-interface, requiring configuration on both neighbors.
- MD5 is older but still widely used; HMAC-SHA-256 is more secure.
- NTP is crucial for time-sensitive key rotations.

OSPF Overview

- OSPF (Open Shortest Path First) is a classless, link-state routing protocol.
- Uses the Dijkstra Shortest Path First (SPF) algorithm to determine the best path.
- Metric is based on Cost, which is inversely proportional to bandwidth.

Classless Nature

- OSPF supports VLSM and CIDR, meaning it can distinguish between subnets like 192.168.1.0/25 and 192.168.1.128/25 as separate networks.
- Unlike classful protocols (e.g., RIPv1), OSPF sends subnet masks with LSAs.

OSPF Packet Types

Packet Type	Purpose
Hello	Establishes and maintains neighbor relationships.
DBD (Database Description)	Summary of LSDB (list of LSAs). Used during adjacency formation.
LSR (Link-State Request)	Requests missing LSAs.
LSU (Link-State Update)	Contains LSAs; used to update neighbors.
LSAck (Link-State Acknowledgment)	Confirms receipt of LSUs.

OSPF Neighbor States

1 - Down – No Hello received.

2 - Init – Hello received, but neighbor has not acknowledged it.

3 - 2-Way – Bi-directional communication established.

4 - ExStart – Master/Slave relationship established for DBD exchange.

5 - Exchange – Routers exchange DBD packets (summary of LSDB).

6 - Loading – Missing LSAs are requested via LSR and received via LSU.

7 -Full – Neighbors fully synchronized and ready to forward traffic.

LSAs & LSUss

- LSAs (Link-State Advertisements) contain routing information (network, cost, mask, etc.).
- LSUs (Link-State Updates) are used to carry LSAs and update neighbors.
- LSA Sequence Numbers ensure routers only accept newer LSAs.

Router ID (RID) Selection

- Manual Configuration (router-id x.x.x.x) – Highest priority.
- Highest Loopback IP (if no manual ID).
- Highest Physical Interface IP (if no loopback).
- Interfaces must be UP to be considered.
-

Key OSPF Features

- Supports Hierarchical Design (multi-area with Area 0 as the backbone).
- Uses DR/BDR elections on multi-access networks to reduce flooding.
- Supports Route Summarization (at ABRs/ASBRs).

Show ip ospf neighbor – displays neighbor relationship

Show ip ospf database – shows LSAs in the Link-state Database

Show ip ospf interface – Lists OSPF- enabled interfaces and network types

Show ip ospf hello – shows real-time hello packets

OSPF Default Route Advertisement – Summary

OSPF does not automatically advertise a default route (0.0.0.0/0). It must be manually injected, typically by an ASBR, and propagated by ABRs into other areas.

There are two main methods to advertise a default route in OSPF:

1 - default-information originate – Distributes only the default route and requires an existing static route (0.0.0.0/0). The always option forces advertisement even if no default route exists.

2 - redistribute static subnets – Injects all static routes, including 0.0.0.0/0, requiring filtering mechanisms like route-maps for control.

ASBRs inject external routes using LSA Type 5, and ABRs generate LSA Type 4 to inform routers in other areas where the ASBR is located. As a result, routers in non-backbone areas learn the default route but see the ABR as their next-hop, not the ASBR.

OSPF Cost Tuning – Understanding and Configuration

Hello and Dead Interval Timers

Hello interval: Determines how often OSPF Hello packets are sent.

Dead interval: If no Hello packets are received within this time, the neighbor is declared down.

Default values depend on the network type:

- Broadcast & Point-to-Point: Hello = 10s, Dead = 40s
- NBMA & Other Networks: Hello = 30s, Dead = 120s

These values can be tuned to improve convergence.

Modifying OSPF Timers

Command to change Hello timer:

```
interface GigabitEthernet0/1
ip ospf hello-interval 5
```

Command to change Dead timer:

```
interface GigabitEthernet0/1
ip ospf dead-interval 20
```

OSPF DR/BDR Election & Multicast Communication

OSPF elects a DR and BDR on Broadcast and NBMA networks to minimize LSA flooding and improve efficiency.

Multicast Address Usage:

224.0.0.6 (AllDRouters) → All routers listen on this address, but only the DR and BDR process LSAs from DROthers.

224.0.0.5 (AllSPFRouters) → The DR floods LSAs to all OSPF routers.

LSA Flow in a Multi-Access Network:

DROthers send LSAs to 224.0.0.6 (processed by the DR and BDR).

The DR floods LSAs to 224.0.0.5 (received by all DROthers).

The BDR does not flood LSAs unless the DR fails.

DROthers only form Full adjacency with DR/BDR and remain in 2-Way state with other DROthers to prevent unnecessary LSA exchanges.

OSPF Priority and DR/BDR Elections

ip ospf priority X (interface-level) determines DR/BDR election (higher priority = preferred DR).

Routers with priority 0 do not participate in DR/BDR elections.

OSPF elections occur in the 2-Way state and are determined by:

- Highest priority (interface-level setting).
- Router ID (used as a tiebreaker if priorities are equal).

Changing ip ospf priority does not take effect until the OSPF process is restarted or the interface is reset.

OSPF LSA Types – Summary

OSPF uses **Link-State Advertisements (LSAs)** to exchange routing and topology information, forming the **Link-State Database (LSDB)**.

Intra-Area LSAs (Stay Within the Area)

- **Type 1 (Router LSA)** – Generated by all routers, advertises directly connected links and costs.
- **Type 2 (Network LSA)** – Generated by the DR, lists all routers on a broadcast or NBMA network.

Inter-Area LSAs (Cross Areas via ABRs)

- **Type 3 (Summary LSA)** – Generated by ABRs, advertises networks from one area to another.
- **Type 4 (ASBR Summary LSA)** – Generated by ABRs, provides ASBR location for Type 5 LSAs.

External LSAs (Injected by ASBRs, Reach Non-OSPF Routes)

- **Type 5 (External LSA)** – Generated by ASBRs, advertises external routes (E1/E2) but does not specify ASBR location.
- **Type 7 (NSSA External LSA)** – Used in NSSAs instead of Type 5; converted to Type 5 by ABRs when leaving NSSA.

Key Points

- Type 1 and Type 2 do not leave their area.
- Type 3, 4, and 5 cross areas, enabling inter-area and external route propagation.
- Type 4 is required for Type 5 LSAs to be usable in other areas.
- Type 7 allows external routes in NSSAs and is converted to Type 5 by ABRs.

OSPF Area Types & Key Concepts

OSPF Area Type	Blocks Type 3?	Blocks Type 4?	Blocks Type 5?	Uses Type 7?	Default Route Injected?	Key commands
Regular Area	No	No	No	No	No	area x
Stub Area	No	Yes	Yes	No	Yes (ABR)	area x stub
Totally Stubby Area	Yes	Yes	Yes	No	Yes (ABR)	area x stub no-summary
NSSA	No	Yes	Yes	Yes	No (Unless default information-originate)	area x nssa
Totally NSSA	Yes	Yes	Yes	Yes	Yes (ABR)	area x nssa no summary

Commands –

Area x stub – Convert area to stub

Area x stub no-summary – convert area to Totally stubby

Area x nssa – convert area to NSSA

Area x nssa no-summary – convert area to Totally NSSA

Area x nssa default-information-originate – Inject Default route into NSSA

Area x nssa translate type7 always – Force an ABR to convert Type 7 to Type 5 LSAs

Area x nssa range <prefix> <mask> - Summarize Type 7 LSAs before conversion.

Summary-address <prefix> <mask> - Summarize external routes (Type 5) from ASBR

OSPFv3 Summary

- **Neighbor Discovery:** Uses **IPv6 link-local addresses** instead of global/unicast.
- **Multiple Instances:** Supports **multiple OSPF instances per interface** (Instance ID).
- **IPv4 Support:** Can route **IPv4 and IPv6** using **address families**.
- **LSA Changes:**
    - **LSA Type 3** replaces OSPFv2's **Type 3 & 4** (Summary & ASBR Location).
    - **LSA Type 8** advertises **link-local addresses** (neighbor discovery).
    - **LSA Type 9** advertises **prefixes** (replaces OSPFv2's Type 1 & 2 prefix info).
- **Configuration:**
    - **No network command**, OSPF is enabled **per interface** (ospfv3 1 ipv6 area 0).
    - Uses **IPv6 unicast-routing** and requires a **32-bit Router ID**.
- **Security:** Authentication is removed from OSPF; uses **IPsec for security**.

OSPFv3 Essential Commands

Enable IPv6 Routing

```
R1(config)# ipv6 unicast-routing
```

Configure OSPFv3 Process and Router ID

```
R1(config)# router ospfv3 1
R1(config-router)# router-id 1.1.1.1
```

Enable OSPFv3 on Interfaces

```
R1(config)# interface GigabitEthernet0/0
R1(config-if)# ospfv3 1 ipv6 area 0
```

Verify OSPFv3 Neighbors and Interfaces

```
R1# show ospfv3 neighbor
R1# show ospfv3 interface
```

Verify OSPFv3 Routing

```
R1# show ipv6 route ospf
```

OSPFv3 for IPv4 (Address-Family)

Enable OSPFv3 for IPv4 Routing

```
R1(config)# router ospfv3 1
R1(config-router)# address-family ipv4 unicast
R1(config-router-af)# router-id 1.1.1.1
```

Apply OSPFv3 for IPv4 on an Interface

```
R1(config)# interface GigabitEthernet0/0
R1(config-if)# ospfv3 1 ipv4 area 0
```

Verify OSPFv3 for IPv4

```
R1# show ospfv3
R1# show ip route ospf
```

BGP Fundamentals – CCNP ENCOR Summary

Protocol Overview

- **Border Gateway Protocol (BGP)** is a **path vector routing protocol** used for interdomain routing across autonomous systems (ASes).
- BGP is **classless**, supports CIDR, and is designed for scalability and policy control.
- It uses **TCP port 179** for session establishment and reliability.
- Common use cases include **ISPs**, **enterprise multi-homing**, and **datacenter overlays** (e.g., VXLAN EVPN).

BGP Neighbor Relationships

- BGP neighborships must be **manually configured** using neighbor <IP> remote-as <ASN>.
- **eBGP** is formed between routers in different ASes; **iBGP** between routers in the same AS.
- eBGP expects directly connected peers unless **ebgp-multihop** is configured.
- When peering via loopbacks:
    - **update-source Loopback0** is required (not supported in Packet Tracer).
    - Loopback addresses must be **reachable via IGP**.
- BGP uses a **TCP 3-way handshake**. If the TCP session fails (e.g., no RST-ACK or unreachable source), BGP fails to establish.

Neighbor State Machine

1. **Idle** – Initial state, waiting to start TCP.
2. **Connect** – TCP handshake initiated.
3. **Active** – TCP retry in progress.
4. **OpenSent** – Open message sent, waiting for response.
5. **OpenConfirm** – Keepalive received; waiting to confirm.
6. **Established** – Peering complete; Update messages exchanged.

BGP forms adjacencies with **stable TCP sessions** and transitions to the **Established** state only after mutual Open/Keepalive exchanges.

BGP Message Types

- **Open** – Contains version, AS number, router ID, hold time, and optional parameters.
- **Keepalive** – Maintains session liveliness; default interval is 60 seconds.
- **Update** – Carries route advertisements, withdrawals, and path attributes.
- **Notification** – Used to signal errors and tear down sessions.

Route Advertisement

- Prefixes are injected into BGP via:
    - network <prefix> mask <mask> (must exist in RIB).
    - redistribute from another routing protocol (e.g., OSPF, static).
- Redistributed routes must be **present in the local RIB**.
- BGP **does not originate routes** unless instructed via network or redistribution.

Path Vector and Loop Prevention

- BGP uses **AS_PATH** as a loop prevention mechanism.
- A router discards routes that contain its own AS in the AS_PATH.
- Each BGP Update includes:
  - **Next-hop IP**
  - **AS_PATH**
  - **Origin** (IGP, EGP, Incomplete)
  - **Other attributes** (Local Preference, MED, etc.)

BGP Clearing Techniques

The general concept of BGP clearing is to reprocess routing information and policies exchanged between neighbors. This can be done in hard or soft methods, each with different operational and convergence impacts.

Hard Reset

```
clear ip bgp {neighbor-address}
```

- Terminates the TCP session between BGP neighbors.
- Forces a full BGP session re-establishment, going through the BGP finite state machine (FSM):

Idle → Connect → Active → OpenSent → OpenConfirm → Established

- All learned routes from that neighbor are flushed from the BGP table and RIB.
- New UPDATE messages must be exchanged again.
- This is a disruptive and CPU-intensive operation, especially in large-scale environments such as ISPs.

Soft Reset

Soft reset allows policy re-evaluation without tearing down the TCP session. This includes both inbound and outbound reprocessing.

Soft Reset Inbound

```
clear ip bgp {neighbor} in
```

- Triggers re-evaluation of all received routes from the neighbor against updated inbound policies (such as prefix-lists and route-maps).
- The BGP session remains up during this process.
- Requires one of the following:
  - Route Refresh capability, which is enabled by default on modern Cisco IOS versions.
  - Or pre-configured inbound soft reconfiguration:

```
neighbor {ip-address} soft-reconfiguration inbound
```

This stores received routes locally to allow policy re-evaluation.

Soft Reset Outbound

```
clear ip bgp {neighbor} out
```

- Triggers re-evaluation of routes advertised to the neighbor based on updated outbound policies.
- Does not require special configuration or route caching.
- Does not interrupt the BGP session.

Route Refresh Capability

- Defined in RFC 2918.
- Allows a BGP peer to request the other side to resend its UPDATE messages.
- This enables soft resets without storing all received routes locally, making it more memory-efficient than the legacy soft-reconfiguration method.

BGP Path Selection Process

Attribute	Type	Default	Scope	Description
Weight	Cisco Proprietary	0	Local	Locally significant; highest wins; not propagated.
Local Preference	Well-known Discretionary	100	AS-wide (iBGP)	Indicates preferred exit point; higher is better.
Locally Originated	Not an Attribute	N/A	Local	Routes originated via network/redistribute/aggregate are preferred.
AS_PATH	Well-known Mandatory	N/A	Inter-AS	Shortest AS_PATH preferred; used for loop prevention.
Origin	Well-known Mandatory	IGP (i)	Inter-AS	Origin type preference: IGP < Incomplete < EGP.
MED	Optional Non-Transitive	0	Inbound, Inter-AS	Lower is better; by default only compared if from the same AS.
eBGP over iBGP	Not an Attribute	N/A	Global	eBGP paths preferred over iBGP if all else equal.
IGP Metric to Next Hop	Not an Attribute	N/A	RIB/IGP	Path with lower IGP cost to next hop wins.
Oldest Path	Not an Attribute	N/A	Local	Oldest BGP path in the table preferred.
Router ID	Not an Attribute	Lowest wins	Inter-AS	Lowest originating BGP Router-ID wins tie.
Neighbor IP (iBGP only)	Not an Attribute	Lowest wins	iBGP	Final tie-breaker among equal iBGP peers.

Type	Recognized	Propegated	Examples	Notes
Well-known Mandatory	Yes	Yes	AS_PATH, ORIGIN, NEXT_HOP	Always present in UPDATEs and recognized by all BGP speakers.
Well-known Discretionary	Yes	Yes	LOCAL_PREF, ATOMIC_AGGREGATE	May or may not appear in UPDATEs, but must be recognized.
Optional Transitive	No	Yes (Partial bit if unknown)	COMMUNITY, AGGREGATOR	Unknown attributes are marked partial and still forwarded.
Optional Non-Transitive	No	No	MED, ORIGINATOR_ID (in Confederations/RR)	Unknown attributes are discarded and not forwarded.

The BGP Path selection process refers to the process that takes place when an identical destination is received from 2 different neighbors, the process determines which of the routes will be implemented in the RIB, and the other will stay in the BGP Topology table.

The process occurs 1 attribute at a time, where if an attribute is accepted the prcoess stops.

BGP Remove-AS

remove-private-as is a BGP configuration command used to **strip private AS numbers** from the **AS_PATH** when advertising routes to external (eBGP) neighbors, such as ISPs.

Use Cases:

- Hiding internal AS structure from outside networks
- Maintaining administrative and regional segmentation across an enterprise
- Providing a clean AS_PATH to simplify external route processing

Behavior Overview:

Command: remove-private-as

- Removes **only adjacent private ASNs** from the end of the AS_PATH
- Commonly used to hide the **last internal AS** before advertising externally

Command: remove-private-as all

- Removes **all occurrences** of private ASNs from the entire AS_PATH
- Useful for completely concealing internal AS hierarchy

Example Scenario:

- Enterprise with multiple internal ASNs:
  AS1300 → AS1400 → AS1500 → AS20 (ISP)
- AS1500 advertises a route learned from AS1300 to the ISP (AS20).
  Original AS_PATH:

  AS1500 AS1400 AS1300

With remove-private-as (default mode):

- Removes only the last private ASN (AS1300), resulting in:

  AS1500 AS1400

With remove-private-as all:

- Removes all private ASNs (AS1300 and AS1400), resulting in:

  AS1500

Additional Notes:

- Only affects **outbound eBGP updates**
- Applies only to **private ASNs** (e.g., 64512–65535 for 16-bit ASN)
- Reduces AS_PATH length (which can influence best path selection)
- Does not cause routing loops when properly used, as loop prevention relies on valid AS_PATHs at the receiving router

BGP Communities

BGP communities are 32-bit tags used to group routes and influence routing policy within or across ASes.
They are **optional transitive**, meaning:

Routers **can ignore** the tag if unsupported

But they **must forward** it in UPDATEs

Common Use Case:

Influence routing by matching on a community and applying policy (e.g., changing local-preference).

Example:

- AS5000 sets a community 5000:20 on prefix 192.168.1.0/24:

```
ip access-list extended MatchLAN
 permit ip 192.168.1.0 0.0.0.255

ip community-list standard TAG_PREF permit 5000:20

route-map SET_COMMUNITY permit 10
 match ip address MatchLAN
 set community 5000:20

router bgp 5000
 neighbor 1.1.1.1 remote-as 1000
 neighbor 1.1.1.1 send-community
 neighbor 1.1.1.1 route-map SET_COMMUNITY out
```

AS1000 **matches that community** and increases local preference:

```
route-map MATCH_TAG permit 10
 match community TAG_PREF
 set local-preference 200

router bgp 1000
 neighbor 1.1.1.1 route-map MATCH_TAG in
```

Key Uses:

- Modify attributes: local-preference, MED, AS-path
- Route control: selective redistribution, filtering, blackholing

- Signaling intent to upstream/downstream peers

BGP Multihoming

Multihoming provides **redundancy** and/or **load balancing** by connecting an enterprise to **one or more ISPs**.

Types of Multihoming

TYPE	DESCRIPTION	PATH CONTROL TOOLS
SINGLE-HOMED	One ISP, one physical link	Static route, default
DUAL-HOMED	Two links to the same ISP	MED, Weight, Local Pref
SINGLE-MULTIHOMED	Two ISPs, one link to each	AS-PATH prepend, MED
DUAL-MULTIHOMED	Two links to each of two ISPs	Full policy control

- Active/Passive vs Dual-Active
- Active/Passive: One link used unless failure occurs (basic failover)
- Dual-Active: Both links carry traffic (load-sharing based on policy)

Path Control Tools

ATTRIBUTE	DIRECTION	USE CASE EXAMPLE
MED	Outbound to ISP	Suggest preferred ingress path into your AS
AS-PATH PREPEND	Outbound to ISP	Make a path less attractive from ISP perspective
LOCAL PREFERENCE	Inbound from ISP	Prefer one outbound exit over another
WEIGHT	Local only	Prefer path on local router

Example: Dual-Homed to Single ISP

Enterprise (AS 65001) → ISP (AS 100)

```
router bgp 65001
 neighbor 10.1.1.1 remote-as 100
 neighbor 10.2.2.2 remote-as 100
 neighbor 10.1.1.1 route-map PREFER_THIS_LINK out

route-map PREFER_THIS_LINK permit 10
 set metric 10
```

This sets **lower MED** for link 10.1.1.1, so the ISP prefers that for incoming traffic.

Well Known BGP Communities

COMMUNITY	DECIMAL (ASN:VALUE)	HEX	MEANING
NO-EXPORT	65535:65281	0xFFFFFF01	Do not advertise to any eBGP peers
NO-ADVERTISE	65535:65282	0xFFFFFF02	Do not advertise to any BGP peer (eBGP or iBGP)
NO-EXPORT-SUBCONFED	65535:65283	0xFFFFFF03	Do not advertise outside of a BGP confederation
LOCAL-AS	65535:65284	0xFFFFFF04	Do not advertise outside the local AS, including confederation peers

These well known BGP communities work natively and don't require any route-maps on the receiving end.

Example:

```
route-map TAG_COMMUNITY permit 10
 set community no-export
!
router bgp 65000
 neighbor 10.1.1.2 route-map TAG_COMMUNITY out
 neighbor 10.1.1.2 send-community
```

Multicast

Multicast enables efficient **one-to-many** or **many-to-many** data distribution.

Compared to:

- **Unicast**: one-to-one.
- **Broadcast**: one-to-all (on local subnet).

Multicast avoids flooding, prevents unnecessary duplication, and limits processing to interested hosts.

Efficiency Goals

Hosts must explicitly "join" a multicast group to receive traffic.

Reduces:

- Bandwidth waste
- CPU usage on non-interested hosts
- Layer 2 flooding

Multicast is **connectionless**, built over **UDP**, with **no built-in reliability** — suitable for real-time or best-effort applications (e.g., IPTV, stock feeds, VoIP).

Multicast IP Addressing (Class D)

Range: 224.0.0.0/4 (binary prefix 1110)

- **224.0.0.0/24**: Reserved for **link-local** (e.g., routing protocols: OSPF, EIGRP)
- **233.0.0.0/8**: **SSM (Source-Specific Multicast)** per IANA
- **239.0.0.0/8**: **Admin-scoped** (private multicast use)

Hosts send and receive **to/from group addresses**, not specific hosts.

MAC Address Mapping

IPv4 multicast maps to **Ethernet MAC**:

01:00:5E:0X:XX:XX

First 25 bits fixed: 01:00:5E:0

Lower **23 bits of IP** used for mapping — introduces **MAC address overlap**:

- Multiple multicast IPs can map to the **same MAC**

Multicast Trees

- **Source Tree (S,G)**: Built from source to receivers (shortest path).
- **Shared Tree (*,G)**: Uses **Rendezvous Point (RP)** — scalable for multiple sources.

Join Process (IGMP + PIM)

1. **Host application** triggers a **Join** via **IGMP** (v2/v3).
2. **Switch (IGMP Snooping)** tracks interested ports.
3. **Router (PIM-enabled)** sends a Join toward:
    - RP (*,G) in **PIM-SM**, or
    - Source (S,G) in **PIM-SSM**
4. **Tree built**, multicast forwarded **only to requesting receivers**.

Key Properties

- **Stateless transmission**
- **Receiver-driven** model
- **Efficient bandwidth use**
- **Minimal CPU impact on non-members**

IGMP Operations and Roles

Multicast in LANs vs WANs

PIM: Used between routers to build and maintain multicast distribution trees across routed networks.

IGMP: Used within a **broadcast domain** (Layer 2 segment) to allow **hosts to signal multicast group membership** to their local router.

IGMP Roles

ROLE	FUNCTION
HOST	Joins or leaves multicast groups based on application demand.
QUERIER (ROUTER)	Periodically queries the LAN for active group members and handles **group pruning** when interest ends.

Querier Election:

Based on lowest IP address on an interface in the subnet (not loopback).

Tiebreaking isn’t needed — IP uniqueness is required.

Querier must be directly connected (Layer 2 adjacency).

IGMP Versions Recap

VERSION	FEATURE SET
IGMPV1	Only Joins; no Leave messages or Querier election. Routers timeout membership (default 3 minutes).
IGMPV2	Adds **Leave** messages and **Querier election** (lowest IP wins).
IGMPV3	Adds **source filtering**: hosts can Include/Exclude specific sources (**required for SSM**).

IGMPv2 Message Types

Membership Report (Join) — Host signals interest in a group (e.g., 239.0.0.200).

Leave Group — Host signals intent to leave a group.

General Query — Sent by Querier to 224.0.0.1; all groups/all hosts.

Group-Specific Query — Sent when a **Leave** is received; targeted to the group.

Group-and-Source-Specific Query — IGMPv3 only, used for source filtering validation.

IGMPv2 Operational Flow

1. Host Joins:

- Sends Membership Report for a group.
- Switch updates its IGMP Snooping table (port to group mapping).
- Router updates multicast state and may send PIM Join upstream.

2. Periodic General Queries:

- Querier sends them every 125 seconds (default).
- Hosts reply with group reports if still interested (within 10 sec max response).

3. Host Leaves:

- Sends a Leave Group message to 224.0.0.2.
- Only the Querier responds.

4. Router Sends Group-Specific Query:

- Probes the group: "Are there any members still subscribed?"
- Waits for responses within the Last Member Query Interval (default: 1 second, 2 retries).

5. Forwarding Decision:

- If no response → group is removed from OIL, router may send PIM Prune upstream.
- If there are responses → multicast continues flowing.

Additional Notes

- **Only one host responds** to General Query due to **randomized delay** (0–10 sec).
- IGMP snooping switches rely on these messages to restrict multicast flooding.
- Querier must continuously maintain group memberships or the state expires.

Protocol Independent Multicast – Dense Mode (PIM-DM)

Operational Model: Whitelist by Default

PIM-DM operates on a flood-by-default approach, assuming all routers want multicast traffic unless explicitly told otherwise. Best suited for densely populated multicast environments, such as video distribution in closed networks.

PIM-DM Operation Flow

1. Source begins transmitting multicast traffic.
2. The First-Hop Router (FHR) floods multicast traffic to all PIM-enabled interfaces, except –
    - The incoming interface (RPF interface)
    - Interfaces in a pruned state (if applicable).
3. Routers that have no local IGMP group members, and Receive no PIM Joins from downstream neighbors, send a PIM Prune upstream toward the source.
4. Upon receiving a Prune, the upstream router removes that interface from its Outgoing Interface List (OIL) for the multicast group.
5. If all interfaces are pruned, the router removes the (S,G) entry from its mroute table.

Prune Timeout and Reflooding

Prune entries expire after 3 minutes (default). Once expired, the router refloods traffic on the interface, this triggers new Prune if there’s still no interest. This loop continues unless interest reappears.

Dynamic Re-subscription

If a host joins after a Prune, The downstream router sends a PIM Graft message upstream. The upstream router restores the interface to Forwarding and replies with a Graft-Ack.

PIM-DM Message Types

MESSAGE	PURPOSE
PRUNE	Downstream router says “I have no interested receivers.”
GRAFT	Re-activates a previously pruned interface.
GRAFT-ACK	Acknowledges successful reactivation of the path.

RPF Integration

Every packet undergoes an RPF check based on the source IP. Uses CEF to verify correct incoming interface; otherwise, the packet is dropped. This check is per-packet but hardware-accelerated.

Pros and Cons of PIM-DM

STRENGTHS	LIMITATIONS
SIMPLE CONFIGURATION	Flooding is inefficient in sparse networks
NO NEED FOR RENDEZVOUS POINT	Lack of receiver-based control
GOOD FOR LAB/TESTING ENVIRONMENTS	High bandwidth waste on unused links
EASY TO TROUBLESHOOT	Limited scalability

PIM-DM Config -

```
! Enable multicast globally
ip multicast-routing

! Configure all participating interfaces (routers and LANs)
interface Ethernet0/0
 ip address 192.168.1.1 255.255.255.0
 ip pim dense-mode

interface Ethernet0/1
 ip address 10.1.1.1 255.255.255.0
 ip pim dense-mode

! (On source router) generate multicast traffic
ping 239.1.1.1 repeat 1000

! (On receiver router) simulate IGMP membership
interface Ethernet0/1
 ip igmp join-group 239.1.1.1
```

Optional Troubleshooting -

```
show ip mroute
show ip igmp groups
debug ip pim
debug ip mpacket
```

Protocol Independent Multicast – Sparse Mode (PIM-SM)

Operational Model: Blacklist by Default

PIM-SM assumes no receivers unless explicitly told otherwise. Multicast traffic is not forwarded unless receivers join the group. Best suited for sparse, scalable networks where receivers are distributed.

PIM-SM Operation Flow

1. Source sends multicast to a group (e.g., 239.1.1.1).
2. Source's DR (Designated Router) encapsulates multicast packets into PIM Register messages and sends them unicast to the RP (Rendezvous Point).
3. Receiver joins using IGMP → its DR sends PIM (*,G) Join toward the RP.
4. Once the RP has both:
    - (S,G) from the source (via Register or native)
    - (*,G) from a receiver → RP decapsulates and forwards multicast down the shared tree.
5. After a few packets, the RP may send Register-Stop to the source's DR and switch to native multicast.
6. The receiver's DR may build an optimized (S,G) Shortest Path Tree (SPT) directly to the source, bypassing the RP.
7. When the host leaves the group, IGMP triggers a PIM Prune, and the multicast state is removed.

PIM-SM Message Types

MESSAGE	PURPOSE
PIM REGISTER	Source DR informs RP about multicast traffic (encapsulated)
PIM REGISTER-STOP	RP tells source DR to stop sending encapsulated traffic
PIM JOIN (*,G)	Receiver DR joins shared tree via RP
PIM JOIN (S,G)	Receiver DR builds direct path to source (SPT)
PIM PRUNE	Receiver DR or RP stops receiving traffic for (S,G) or (*,G)

Timeouts and State Expiration

- Join/Prune State Timeout: ~3 minutes by default (can vary by implementation).
- Register-Stop holdtime: Typically 60 seconds, renewable.
- SPT switch threshold: Can be immediate or based on traffic volume (ip pim spt-threshold).

RPF (Reverse Path Forwarding) Check

Every multicast packet undergoes an RPF check to ensure it arrived on the correct interface based on the source address.

- If RPF fails → packet is dropped.
- Check is per-packet but performed via CEF (hardware accelerated).

Pros and Cons of PIM-SM

STRENGTHS	LIMITATIONS
SCALABLE FOR LARGE, DISTRIBUTED NETWORKS	Requires proper RP configuration
RECEIVER-DRIVEN – NO UNNECESSARY TRAFFIC	RP can become bottleneck or single point of failure
SUPPORTS SPT OPTIMIZATION	More complex to configure than PIM-DM

Basic PIM-SM Configuration

```
! Enable multicast globally

ip multicast-routing


! Configure interfaces with PIM-SM

interface Ethernet0/0

 ip address 10.1.1.1 255.255.255.0

 ip pim sparse-mode


interface Ethernet0/1

 ip address 192.168.1.1 255.255.255.0

 ip pim sparse-mode


! (Optional - Static RP)

ip pim rp-address 10.1.1.1


! (On receiver router - join group)

interface Ethernet0/1

 ip igmp join-group 239.1.1.1


Optional Troubleshooting Commands

show ip mroute            ! View multicast routing table

show ip pim neighbor        ! Verify PIM neighbors

show ip pim rp mapping        ! RP assignment

show ip igmp groups         ! IGMP membership on interfaces

debug ip pim              ! Live PIM message debugging

debug ip mpacket    ! Display real-time information of mpackets
```

RP Election process

The process is either manually via configuring a static RP Address on all participating routers –

Ip pim rp-address <RP-IP>

Or via Cisco’s implemntation of Auto-RP –

Ip pim send-rp-announce loopback0 scope 10

#on the mapping agent (Can be same router):

Ip pim send-rp-discovery loopback0 scope 10

#on all sparse-mode routers (To receive Auto-RP in sparse-mode only networks)

Ip pim autorp listener

Either works fine, the crucial part is remembering to put it on all participating interfaces.

SPT Threshold Side Note

If both (S,G) and (RP,G) entries remain active, the SPT switchover may not have completed due to threshold behavior, It’s likely safer to wait 3~ minutes for it to disappear due to timers but if not, the following commands can be used -

Fix -

ip pim spt-threshold 0

clear ip mroute <group>

Verify -

show ip mroute <group>

QoS

Why QoS Was Invented?

QoS (Quality of Service) was developed to solve problems in modern IP networks that don't inherently guarantee timely or fair delivery of traffic. It became essential due to:

- **Bandwidth Constraints** — Especially on WAN links and oversubscribed uplinks.
- **Legacy Infrastructure** — Older access layers and low-speed interfaces (e.g., T1, DSL).
- **Traffic Competition** — Mixed traffic types (voice, video, bulk data) require differentiated handling.
- **Application Sensitivity** — VoIP, video conferencing, and control traffic are delay- and loss-sensitive.

Latency and Jitter

- **Latency** = Total time for a packet to travel from source to destination.
- **Jitter** = Variation in delay between consecutive packets — particularly harmful to real-time traffic.

The Four Horseman of Latency

1. **Propagation Delay**
   - Time for bits to physically travel across the medium.
   - Depends on **distance** and **media type** (fiber, copper).
2. **Serialization Delay**
   - Time required to push all bits of a packet onto the wire.
   - Formula: Packet size / Link bandwidth
3. **Processing Delay**
   - Time taken by network devices (routers, switches, firewalls) to inspect headers and make forwarding decisions.
4. **Queuing Delay**
   - Time packets wait in output buffers due to **congestion** or **scheduling policies**.

Why QoS Focuses on Queuing Delay

Propagation, serialization, and processing are largely hardware- and topology-bound.

Queuing delay is:

- Variable and congestion-dependent
- **Directly controllable** through QoS tools:
    - Classification
    - Marking
    - Scheduling (e.g., LLQ, WFQ)
    - Congestion avoidance (e.g., WRED)
    - Shaping and policing

Thus, **QoS is primarily the control and optimization of queuing behavior**, ensuring timely delivery for prioritized traffic and fairness across classes.

QoS Models

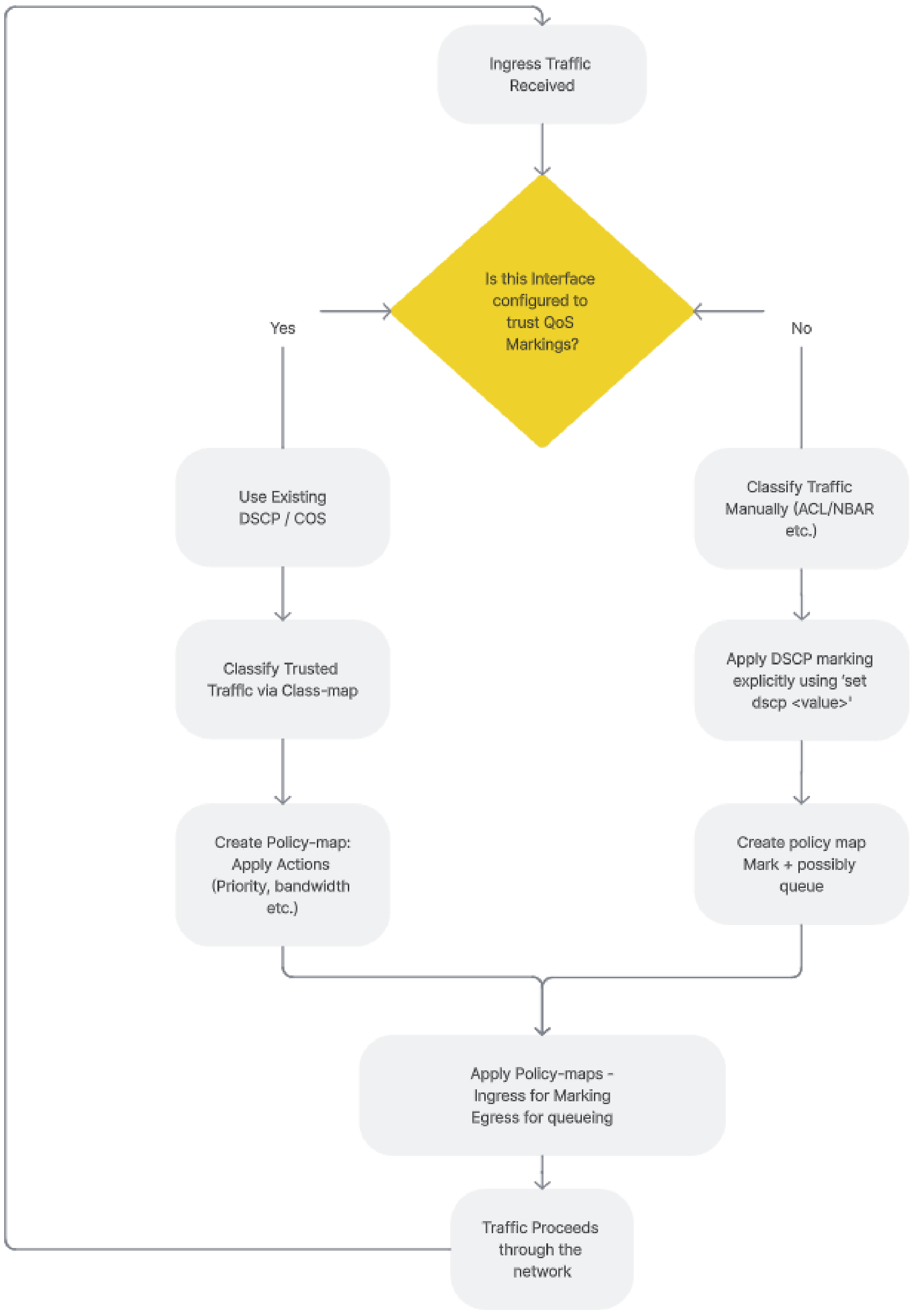
- Best Effort (No QoS) – All traffic is treated equally, there is no preferational treatment and it works in a FIFO model where traffic that arrives first is processed first.
- IntServ – Using "RSVP" mechanism, QoS is enacted on a per-flow basis, This also means the entire path needs to have the ame RSVP reservastions, Can be used for applications such as VoIP, Not scalable.
- DiffServ – Class-based QoS, devices mark packets with DSCP tags, for "PHB" (Per hop behavior) actions, better scalability.

TLDR;

- IntServ = RSVP, Per-flow reservation, not scalable, Rarely used.
- DiffServ = Mark once, enforce at each hop via PHBs, Common
- Best Effort = Everyone waits equally, Default.

QoS Marking (DSCP/CoS)

```
#Enable QoS (Switches)

conf t

mls qos


#trust DSCP/CoS (At edge)

interface e0/0

mls qos trust dscp ! Trust DSCP (L3)

mls qos trust cos ! Trust CoS (L2)

mls qos trust device cisco-phone ! Trust only Cisco Phones via CDP


#create class-map (define what to match, either match-any or match-all)

class-map match-any VOICE

match dscp ef ! Match based on Tag

match access-group name VOICE-RTP ! ACLs can be used to match


#create policy-map (define actions)

#Input (Marking with ef)

policy-map MARK-VOICE

class VOICE

set dscp ef

#Output (Queuing)

policy-map QOS-OUT

class voice

priority percent 30

class class-default

fair queue


#apply policy

Interface e0/1

service-policy input MARK-VOICE

service-policy output QOS-OUT
```



Ingress Traffic Received
Is this Interface configured to trust QoS Markings?
Yes
No
Use Existing DSCP / COS
Classify Trusted Traffic via Class-map
Create Policy-map: Apply Actions (Priority, bandwidth etc.)
Classify Traffic Manually (ACL/NBAR etc.)
Apply DSCP marking explicitly using 'set dscp <value>'
Create policy map Mark + possibly queue
Apply Policy-maps - Ingress for Marking Egress for queueing
Traffic Proceeds through the network

The following flowchart showcases the process for dealing with trusted and untrusted traffic.

It's important to rememeber –

Ingress traffic – Tagged & Marked

Egress traffic – Queued & Schedualed.

CoS & DSCP

CoS & DSCP are 2 forms of QoS control, While similar, each has his own area of influence and quirks.

ASPECT	COS	DSCP
OSI LAYER	Layer 2 (Data Link)	Layer 3 (Network)
FIELD SIZE	3 bits	6 bits
LOCATION IN FRAME	802.1Q VLAN Tag (PCP field)	IP Header (DS field)
PRESERVED ON	Trunk Ports Only	End-to-End (IP Packet)
USED FOR	Switch-to-Switch Prioritization	End-to-End QoS across IP Network
TRUST REQUIREMENT	mls qos trust cos	mls qos trust dscp
MARKING SCOPE	Local (L2 domains)	Global (IP domains)

DSCP Tag Reference

DSCP Name	Decimal	Binary	Usage / Behavior
Default (BE)	0	000000	Best Effort (unmarked traffic)
EF	46	101110	Expedited Forwarding (VoIP, low delay)
CS0	0	000000	Default class (same as BE)
CS1	8	001000	Scavenger / Low priority
CS2	16	010000	OAM / Network Control (sometimes)
CS3	24	011000	Signaling / Transactional
CS4	32	100000	Real-time streaming (video)
CS5	40	101000	Interactive voice or high-priority
CS6	48	110000	Network control (routing, OSPF, BGP)
CS7	56	111000	Reserved (highest internal priority)
AF11	10	001010	AF class 1, low drop precedence
AF12	12	001100	AF class 1, medium drop precedence
AF13	14	001110	AF class 1, high drop precedence
AF21	18	010010	AF class 2, low drop precedence
AF22	20	010100	AF class 2, medium drop precedence
AF23	22	010110	AF class 2, high drop precedence
AF31	26	011010	AF class 3, low drop precedence
AF32	28	011100	AF class 3, medium drop precedence
AF33	30	011110	AF class 3, high drop precedence
AF41	34	100010	AF class 4, low drop precedence
AF42	36	100100	AF class 4, medium drop precedence
AF43	38	100110	AF class 4, high drop precedence

CoS Tag Reference & DSCP Mappings

CoS Value	Name	Typical DSCP Mapping	Use Case
0	Best Effort	0 (DF)	Default traffic
1	Background	8 (CS1)	Scavenger, low priority
2	Excellent Effort	16 (CS2)	Business apps
3	Critical Applications	24 (CS3)	Call signaling
4	Video (Streaming)	32 (CS4)	Streaming video
5	Voice	46 (EF)	Voice, low-latency
6	Internetwork Control	48 (CS6)	Routing protocols
7	Network Control	56 (CS7)	Switch/infra control

Policing vs Shaping

Both **policing** and **shaping** are QoS mechanisms that enforce bandwidth control using the same foundational parameters:

Core Variables

CIR (Committed Information Rate):
Defines the **average allowed bandwidth** for traffic, in bits per second.

Bc (Committed Burst):
The amount of traffic (in bytes) allowed to temporarily exceed CIR within a short window — essentially your **burst cushion**.

Be (Excess Burst):
Additional, optional burst allowance beyond Bc — provides **emergency tolerance** or grace during spikes.

Tc (Time Interval):
The time between burst windows, calculated as:

Tc=Bc/CIR

It determines **how often bursts can occur**.

Behavioral Differences

FEATURE	POLICING	SHAPING
DIRECTION	Ingress (incoming traffic)	Egress (outgoing traffic)
WHEN RATE EXCEEDS CIR	Traffic is either **dropped** or **remarked**	Traffic is **buffered and delayed**
QUEUING	No queuing – packets evaluated and acted on immediately	Uses queues to smooth transmission over time
EFFECT ON LOSS	Can cause **packet loss**	Prevents loss, but may introduce **latency/jitter**
USE CASE	Enforce SLAs, drop unwanted traffic early	Smooth traffic to rate-limited links (e.g. WAN)

TL;DR

CIR defines the logical average traffic rate.

Bc and **Be** define how much burst is allowed.

Tc determines how often you're allowed to burst.

Policing is about **dropping or marking** traffic at ingress.

Shaping is about **delaying and smoothing** traffic at egress.

Both tools serve **different parts of the QoS pipeline**, but are built on the **same control model** — understanding one means you nearly understand both.

Queue Management Methods

Queuing methods manage how packets are transmitted when the **egress interface is congested**. These techniques only apply to **packets that enter the output queue**.

FIFO (First In, First Out)

- Default method: first packet in = first out
- No prioritization; simple but not ideal for mixed traffic

#Default, Validated via

Show queueing interface e0/0

WFQ (Weighted Fair Queuing)

- Automatically identifies flows and **distributes bandwidth fairly** adjust for packet size
- No config required; not class-aware

Interface e0/0

Fair-queue

CBWFQ (Class-Based WFQ)

- Admin-defined **traffic classes** with assigned bandwidth shares
- All flows in a class share its bandwidth
- Enables precise control over service levels

#Configured in policy-map

Policy-map QoS-policy

Class VOIP

Bandwidth 3000

LLQ (Low-Latency Queuing)

- Adds a **strict priority queue** to CBWFQ
- Real-time traffic (e.g. VoIP) is always sent first
- Rate-policed to protect other classes

#Configured in policy-map

Policy-map QoS-policy

Class VOIP

Priority 1000

Queue Avoidance

Queue avoidance techniques are designed to **prevent output queues from filling completely**, reducing the risk of **tail-drop** and global TCP slowdown.

The primary mechanism used is **WRED (Weighted Random Early Detection)**.

Key Concepts

WRED monitors the **average queue depth** (not instantaneous) using an **Exponential Weighted Moving Average (EWMA)**.

Based on this average, packets are **dropped probabilistically** to signal congestion before the queue is full.

Thresholds define drop behavior:

- **Min-threshold**: below = no drops
- **Max-threshold**: at or above = 100% drop rate
- **Mark-probability denominator**: controls how aggressively packets are dropped between min/max

WRED is class-aware when used with random-detect dscp-based, allowing higher-priority traffic (based on DSCP or IP precedence) to be **protected longer**.

WRED vs RED

FEATURE	RED	WRED
TRAFFIC AWARENESS	None	DSCP/IP precedence aware
DROP BEHAVIOR	Uniform	Differentiated per class
USE CASE	Basic congestion	Modern, class-based networks

Design note

Do not apply WRED on LLQ (priority) classes — those should be rate-policed, not randomly dropped.

WRED is ideal for TCP traffic where early drops trigger congestion control.

Configure WRED

```
#WRED can be configured per class and per Tag

Class bulk-traffic

Bandwidth 300

Random-detect dscp-based

Random-detect dscp 10 30 60 10 ! DSCP 10 Drop starts after 30

Random-detect dscp 0 20 40 50 ! DSCP 0 Drops start after 20


Class class-default

Random-detect ! default RED for remaining traffic.
```

NTP Fundamentals Summary

NTP (Network Time Protocol) ensures consistent time synchronization across network devices, Accurate time is critical for logging, security (certificates), correlation, and troubleshooting.

Stratum

- 0 - Reference clocks (e.g., atomic, GPS)
- 1 - Directly connected to Stratum 0
- 2+ Learned from higher stratum NTP server

Lower stratum = more accurate

Each hop introduces potential time deviation (delay, jitter, dispersion).

NTP Roles

```
ntp server 192.168.1.1     ! Configure the device to sync from a specific NTP server

ntp master 1               ! Makes the device a time source at stratum 1 using its own clock

ntp peer 1.1.1.1           ! Establish symmetric time exchange with another peer (equal roles)
```

Key NTP Concepts

- NTP uses UDP/123.
- Gradual clock correction via slewing, not abrupt jumps.
- You can sync to multiple servers; use prefer to mark the preferred one.
- Ensures consistent source IP for NTP packets (Typically Loopback):

```
ntp source <interface>
```

Verification Commands

```
show ntp associations ! Shows who you're syncing to, and who they are syncing to.

show ntp status ! Gives your own sync status.
```

Example:

```
Clock is synchronized, stratum 2, reference is 10.0.0.1

clock offset is -0.12 msec, root delay is 30.00 msec
```

Reference Clock

- If your NTP server is using its own internal clock:
  - ref clock = 127.127.1.1 (pseudo-loopback)

- If your NTP server syncs to another server:
  - ref clock = That upstream server's IP

- If you are the master:
  - ref clock = none shown

Example Configuration

```
clock set 12:00:00 22 April 2025 ! On a master

ntp master 2

ntp server 10.0.0.1 prefer ! On a client

ntp source Loopback0
```

FHRP

FHRP protocols provide gateway redundancy at Layer 3 by creating a virtual IP and MAC shared among routers. Ensures uninterrupted default gateway service for hosts.

Protocol Comparison

FEATURE	HSRP (V1/V2)	VRRP (V2/V3)	GLBP
STANDARD	Cisco proprietary	Open standard (RFC 5798)	Cisco proprietary
IPV6 SUPPORT	Only in HSRPv2	Supported in VRRPv3	Supported
ROLES	Active / Standby / Listen	Master / Backup	AVG (gateway), AVFs (forwarders)
DEFAULT PRIORITY	100	100 (255 if owns VIP)	100
PREEMPTION	Manual (standby x preempt)	Enabled by default	Enabled by default
LOAD BALANCING	No	No	Yes — per-host, weighted, etc.
MAX PARTICIPANTS	v1: 0–255 v2: 0–4095	0–255	4 AVFs per group

Multicast & MAC Addresses

PROTOCOL	MULTICAST IP	VIRTUAL MAC FORMAT
HSRP V1	224.0.0.2	0000.0C07.ACxx
HSRP V2	224.0.0.102	0000.0C07.ACxx
VRRP	224.0.0.18	0000.5E00.01xx
GLBP	224.0.0.102	0000.0C9F.Fxxx (F = forwarder)

Operational Highlights

Virtual IP: Shared among all routers in the group. Hosts set this as their default gateway.

Virtual MAC: Dynamically owned by the Active/Master/AVFs and used for L2 forwarding.

Failover: Occurs when the primary role fails; tracking interfaces can dynamically reduce priority.

ARP Handling:

- HSRP/VRRP: Only the Active/Master responds to ARP
- GLBP: AVG responds to ARP and assigns unique virtual MACs per host

NAT

Inside Static NAT

Purpose –

- Translate a private inside local address (LAN) to a globally routable inside global address.

Use Case –

- Allow inside hosts to initiate Internet communication and/or allow outside users to reach inside servers.

Translation –

Inside Local -> Inside Global

Behavior –

- Source IP address is translated when the packet leaves the LAN.
- Destination IP (outside) is usually untouched (outside local = outside global).

Example:

LAN host 192.168.1.10 mapped to 203.0.113.10.

Internet users or servers reply to 203.0.113.10; NAT device forwards it to 192.168.1.10.

Outside Static NAT

Purpose –

- Allow LAN users to connect to fake IP addresses that the router transparently translates into real external IP addresses.

Translation –

- Fake private IP (Outside Local) ➔ Real public IP (Outside Global)

Behavior –

- LAN hosts connect to fake IPs.
- NAT device swaps the fake IP to the real IP before sending the packet to the Internet.

Example:

LAN host sends traffic to 10.1.1.8.

NAT router rewrites destination to 8.8.8.8.

Real server 8.8.8.8 replies normally.

Key Concepts

NAT TYPE	AFFECTED FIELD	MAIN PURPOSE
INSIDE STATIC NAT	Source IP	Provide Internet connectivity for internal devices; allow inbound access to servers
OUTSIDE STATIC NAT	Destination IP	Mask external public servers with internal fake addresses for LAN access

NAT Address Types

TERM	DEFINITION
INSIDE LOCAL	Private LAN IP address (e.g., 192.168.1.10)
INSIDE GLOBAL	Public IP representing inside device (e.g., 203.0.113.10)
OUTSIDE LOCAL	Fake private IP representing external device (e.g., 10.1.1.8)
OUTSIDE GLOBAL	Real public IP of external server (e.g., 8.8.8.8)

Key Points to Remember

- Inside NAT = internal hosts get translated to public IPs.
- Outside NAT = external servers appear as private IPs to internal hosts.

In Inside NAT, the outside local usually equals outside global (no destination translation).

In Outside NAT, the outside local is a private IP and different from the outside global (destination translation occurs).

NAT translations happen on the router/firewall — end devices are unaware of the real IP addresses unless explicitly configured otherwise.

Inside Static NAT Example –

Ip nat inside source static {inside-local} {inside-global}

Outside Static NAT Example –

Ip nat outside source static {Outside global} {Fake-IP} add-route

Important to note, the add-route is only needed when using a fake IP with no routing entry (the command will cause a route-lookup), Without the add-route, and no Routing entry, NAT won’t work.

Outside NAT is not common and typically used to overcome problems related to duplicate addresses.

Dynamic NAT (D-NAT)

Dynamic NAT introduces flexibility by allowing devices to temporarily borrow an inside global address from a predefined pool. Unlike static NAT (1:1 permanent mapping), Dynamic NAT creates mappings only as needed, and releases the global IP when the session ends.

Advantages of Dynamic NAT:

- Flexibility: Any eligible internal client can use any available global address.
- Address Efficiency: Global addresses are reused dynamically.
- Operational Simplicity (for basic access): No need to configure static rules per host.

Caveats and Limitations:

- No control over which address is assigned - Clients are assigned an available IP randomly from the pool — no way to predict which.
- Pool Exhaustion - If all IPs in the pool are consumed, new connections fail (no fallback like PAT), Leads to dropped sessions or denied service (typically indicated by %NAT-6-NATPOOL messages).
- State Table Growth - Every session (ICMP, TCP, UDP) requires a NAT translation entry. Especially in *pure Dynamic NAT* (without PAT), every *new flow* (even ping replies) consumes TCAM (or software CPU) resources. This can fill NAT tables very fast on large networks, making management harder.
- Configuration Complexity - a pool of inside global addresses (ip nat pool), a matching ACL to define which internal addresses are eligible for NAT.

Inside Dynamic NAT Example –

```
ip nat pool MYPOOL 203.0.113.1 203.0.113.10 netmask 255.255.255.0
ip access-list standard NAT_INSIDE
 permit 192.168.1.0 0.0.0.255
ip nat inside source list NAT_INSIDE pool MYPOOL
```

Port Address Translation (PAT)

PAT solves the biggest limitation of Dynamic NAT — the need for multiple inside global IPs — by multiplexing thousands of sessions onto a single public IP using port numbers as additional identifiers.

Many-to-One Translation –

Multiple inside local addresses → a single inside global IP, differentiated by unique TCP/UDP port numbers.

Each flow is uniquely identified by:

- Source IP
- Source Port
- Destination IP
- Destination Port
- Transport Protocol (TCP/UDP/ICMP "ID field")

Advantages of PAT –

- Massive scalability: Thousands of sessions can share one IP.
- Minimal public IP consumption: Only one inside global IP is needed for potentially thousands of users.
- Simple configuration: Only one IP or an interface needed (overload keyword).

Drawbacks –

- Port Exhaustion: A single IP has ~65,000 ports. Realistic usage (~50,000 usable ports) may still get exhausted in high-scale networks.
- If all ports are exhausted: If multiple IPs are configured (via pool or secondary IPs), PAT can load-balance among available addresses. If no backup IPs, new sessions fail (usually logged).
- State Table Bloat: Like Dynamic NAT, every session adds an entry to the NAT translation table. This can grow very large, filling TCAM or software resources.
- Session Stickiness: PAT sessions must be maintained carefully — if the router reloads or loses state, ongoing sessions may break.

GRE

GRE is a tunneling protocol, operating at the Network Layer (Layer 3) of the TCP/IP stack. More precisely, it implements IP-in-IP encapsulation, meaning a full IP packet is wrapped inside another IP packet with a new header.

Why GRE Exists

GRE was originally designed to carry non-IP or legacy protocols (such as AppleTalk, IPX, or multicast) across IP-only networks. As the modern internet became IP-dominant, GRE evolved into a generic overlay mechanism to connect disparate IP subnets over intermediate routed networks.

Overlay Concept

GRE is part of the overlay networking model, where encapsulation creates virtual topologies on top of existing physical infrastructure. Other overlay examples include:

- IPSec: Encrypted IP-over-IP
- VXLAN: MAC-in-UDP for L2 extensions across IP
- MPLS, LISP, etc.

A general rule: overlays encapsulate "data within data", often seen as:

Inner MAC → Inner IP → Outer IP → Outer MAC

GRE Behavior and Forwarding

GRE forms a point-to-point tunnel between routers:

1. Source router receives a packet (e.g., from a host in LAN1).
2. It encapsulates it inside a GRE+IP header:
    - Source IP = its own WAN IP
    - Destination IP = remote router's WAN IP

The packet traverses intermediate routers without being altered, since those routers route only based on the outer IP.

At the remote tunnel endpoint, the outer headers are stripped (de-encapsulation), and the inner packet is forwarded to the true destination.

Security Note

GRE is not encrypted. It provides basic isolation via encapsulation, but no confidentiality or integrity. GRE can be combined with IPSec to provide encryption.

GRE Pre-requisites

- Underlying IP connectivity between the tunnel endpoints is mandatory (PING from tunnel source to destination must succeed).
- Tunnel interface IPs must be in the same subnet, as they behave like directly connected interfaces.
- Tunnel source should be a WAN-facing IP/interface (public or routed), and the tunnel destination must be the remote router's WAN IP.
  - Do not use private/LAN IPs for GRE source/destination.
  - They must be globally reachable via underlay routing.

Recursive Routing Consideration:

- The router must be able to reach the tunnel destination via global routing table — GRE relies on recursive routing.
- You must advertise:
  - The tunnel network
  - The internal (LAN) network
- Never advertise the underlay (WAN) interfaces into the overlay's routing process — that would cause recursive routing failures.

Recursive Routing – GRE

If the WAN network (underlay) is accidentally advertised inside the same routing protocol instance (such as OSPF or EIGRP) that operates over the GRE tunnel, a recursive routing lookup failure can occur.

This leads to GRE tunnel flaps, link instability, and in some cases, more unpredictable routing behavior.

Root Cause:

- Advertising the WAN network (physical tunnel endpoints) over the tunnel causes the router to believe that the tunnel destination is reachable via the tunnel itself.
- When the router attempts to send GRE packets to the tunnel destination, it performs a recursive lookup.
- The routing table points the destination back to Tunnel0, which depends on itself.
- This circular dependency cannot be resolved.

The router logs messages such as:

```
%TUN-5-RECURDOWN: Tunnel0 temporarily disabled due to recursive routing
```

Technical Flow:

1. Tunnel destination IP (e.g., 1.1.1.2) needs to be reached.
2. Routing table says: "Route to 1.1.1.2 is via Tunnel0."
3. GRE needs to encapsulate and send to 1.1.1.2.
4. But Tunnel0 itself depends on reaching 1.1.1.2.

IPSec

IPsec Fundamentals

IPsec (Internet Protocol Security) is a framework of open standards that provides secure communication over IP networks. It enables confidentiality, integrity, authentication, and anti-replay protection between two IPsec peers by negotiating security policies and applying cryptographic services.

IPsec uses two core protocols to apply these services:

Authentication Header (AH) provides –

- Data integrity
- Peer authentication
- Replay protection

Does not provide encryption (no confidentiality), Incompatible with NAT (modifies IP header, which AH protects)

Encapsulating Security Payload (ESP) Provides –

- Encryption (confidentiality)
- Integrity and hashing (optional)
- Peer authentication
- Replay protection

Does not protect the outer IP header, Compatible with NAT (NAT-T support) — making it the most commonly used IPsec protocol

Because ESP supports encryption and NAT compatibility, it is preferred in modern deployments. AH is rarely used outside of niche environments where IP header integrity is essential.

IPsec Modes of Operation

IPsec supports two modes for forwarding traffic:

Tunnel Mode

- Commonly used between gateways or firewalls (site-to-site VPNs)
- Entire original IP packet (IP header + payload) is encrypted
- A new outer IP header is added for routing
- Used with ESP (most common)

Transport Mode

- Typically used between end hosts
- Only the payload (Layer 4 and above) is encrypted
- The original IP header is preserved and visible
- Also supported by ESP

Key Cryptographic Components

Encryption Algorithms (Confidentiality)

DES: 56-bit symmetric cipher (insecure by modern standards)

3DES: Runs DES three times for added security (legacy)

AES: Modern symmetric encryption supporting 128, 192, or 256-bit key lengths (widely used)

Hashing Algorithms (Integrity)

MD5: 128-bit hash function (no longer considered secure)

SHA-1: 160-bit hash function (better, but still considered weak today)

SHA-2 / HMAC-SHA256+: Modern standards offering strong integrity (recommended)

Key Exchange

Diffie-Hellman (DH): Allows two peers to securely derive a shared secret over an insecure channel

DH Groups determine key length and cryptographic strength

- Group 1: 768-bit (obsolete)
- Group 14: 2048-bit (commonly used)
- Group 19+: Elliptic Curve options (faster/stronger)

Peer Authentication Methods

Pre-Shared Key (PSK): Manually configured shared secret string

RSA Digital Certificates: Public Key Infrastructure (PKI)-based identity verification using X.509 certificates

IKEv1 Summary

IKEv1 (Internet Key Exchange version 1) is a protocol used to negotiate and establish secure tunnels in IPsec. It operates in two distinct phases, each with its own role:

- Phase 1: Establishes a secure, encrypted channel called the ISAKMP SA, used to protect further negotiation.
- Phase 2: Uses that channel to negotiate the actual IPsec SAs, which define how traffic is encrypted and authenticated.

Both Phase 1 and Phase 2 create a tunnel, the Tunnel in Phase 1 is used to negotiate the Transform-set for Phase 2

IKEv1 uses either Main Mode or Aggressive Mode during Phase 1.

Phase 1 – ISAKMP SA Negotiation

Main Mode (6 messages), A step-by-step, secure negotiation process:

- MM1: Initiator proposes algorithms (encryption, hash, DH group, authentication method).
- MM2: Responder selects compatible proposal.
- MM3: Initiator sends Diffie-Hellman (DH) public value.
- MM4: Responder sends DH public value.
- MM5: Initiator sends encrypted identity + authentication.
- MM6: Responder sends encrypted identity + authentication.

Provides peer identity protection, Derives a shared session key using DH, Results in an ISAKMP SA (Phase 1 complete)

Aggressive Mode (3 messages), A faster, less secure method:

- AM1: Initiator sends proposal + DH public value + ID/auth payload
- AM2: Responder replies with chosen parameters, DH value, and ID/auth
- AM3: Optional confirmation

Peer identity is sent in clear text (no encryption before DH), Still results in an ISAKMP SA

Phase 2 – Quick Mode and IPsec SA Establishment

Once Phase 1 is complete, the two peers use Quick Mode to negotiate IPsec SAs for actual data traffic.

Quick Mode (3 messages)

- QM1: Initiator proposes transform-set (ESP/AH, encryption, integrity) and traffic selectors (proxy IDs/ACLs)
- QM2: Responder agrees or responds with acceptable transform-set and selectors
- QM3: Initiator confirms SA establishment

Traffic is protected inside the ISAKMP SA (already encrypted), Results in two unidirectional IPsec SAs (one per direction), Optional Perfect Forward Secrecy (PFS) can force new DH exchange

KEv2

IKEv2 (Internet Key Exchange version 2) is a modern and streamlined improvement over IKEv1, designed to simplify and enhance the process of establishing secure IPsec tunnels. It is defined in [RFC 7296] and addresses many of the limitations of IKEv1.

Key Improvements Over IKEv1:

- Reduces the number of message exchanges
- Merges Phase 1 and Phase 2 into a single negotiation
- Simplifies the state machine and error handling
- Supports modern cryptographic algorithms (e.g., AES-GCM, SHA-512, ECDH)
- Provides inbuilt anti-DoS protection
- Supports asymmetric authentication (different methods per peer)
- Enables dynamic creation of additional IPsec SAs (Child SAs)

IKEv2 Exchange Flow

IKEv2 reduces the negotiation process to just 4 total messages (2 from each peer) for initial tunnel setup:

1. IKE_SA_INIT
    - First exchange, Equivalent to IKEv1 messages MM1–MM4, Negotiates:
        - Encryption and integrity algorithms
        - Diffie-Hellman key exchange
        - Nonces (random values for key derivation)
        - No authentication yet
2. IKE_AUTH
    - Second exchange, Equivalent to MM5–MM6 + Quick Mode (QM1–QM2), Negotiates:
        - Peer authentication (PSK, RSA, or EAP)
        - IPsec SA (Child SA) proposal for actual data traffic
        - Traffic selectors (like proxy ACLs)
        - Entire message is encrypted and integrity-protected using keys from IKE_SA_INIT
3. Result:
    - IKE SA: Secure channel for control-plane negotiation (like ISAKMP SA)
    - Child SA: Data-plane tunnel (IPsec SA for encrypted traffic)

Creating Additional IPsec SAs

IKEv2 supports the creation of additional Child SAs using the CREATE_CHILD_SA exchange.

CREATE_CHILD_SA = 1 message from each peer (2 total)

It is used for Rekeying existing IPsec Sas and adding new IPsec tunnels without restarting the IKE SA

Each message includes:

- New transform-set proposal
- Optional new DH exchange (for PFS)
- Updated traffic selectors (clients)

Cryptographic and Security Enhancements

Stronger algorithm support:

- AES-GCM (encryption + integrity)
- SHA-256, SHA-384, SHA-512
- ECDH (Elliptic Curve Diffie-Hellman)

Flexible authentication:

- PSK
- RSA Certificates
- EAP (Extensible Authentication Protocol)

Anti-DoS: Requires peer to prove legitimacy before resource-intensive operations

Asymmetric authentication: Each peer may use a different method (e.g., one PSK, one certificate)

IPSec Config

This section outlines the configuration steps for setting up IPsec in transport mode over a GRE tunnel using IKEv1. included ACL examples for both GRE-based and pure IPsec scenarios.

1. Define the ACL for Interesting Traffic, This access list defines which traffic should be protected by IPsec.

For IPsec over GRE: Match GRE traffic between tunnel source and destination (external IPs):

```
ip access-list extended VPN
 permit gre host <Tunnel-Source-IP> host <Tunnel-Destination-IP>
```

For Pure IPsec: Match actual internal subnets being routed:

```
ip access-list extended VPN
 permit ip <Local-Network> <Wildcard-Mask> <Remote-Network> <Wildcard-Mask>
```

2. Configure the IKE Phase 1 (ISAKMP SA) Policy, This defines the control-plane encryption and authentication parameters.

```
crypto isakmp policy <priority>
 encryption aes 192
 hash sha256
 authentication pre-share
 group 15
```

This phase negotiates the ISAKMP Security Association, establishing a secure channel for negotiating the IPSec SA.

3. Set the Pre-Shared Key, This key is shared with the remote peer for authentication.

```
crypto isakmp key <your-key-string> address <peer-public-IP>
crypto isakmp key <your-key-string> address 0.0.0.0 0.0.0.0
```

Best practice is to specify the peer IP to limit exposure.

4. Create the IPSec Transform Set, This defines how traffic will be protected in the IPSec Phase 2 SA.

```
crypto ipsec transform-set VPN-SET esp-aes 192 esp-sha256-hmac
 mode transport
```

mode transport is required for IPsec over GRE (default is tunnel mode).

5. Configure the Crypto Map, Bind the ACL, transform set, and peer together under a named crypto map:

```
crypto map VPN-MAP 10 ipsec-isakmp
 match address VPN
 set peer <peer-public-IP>
 set transform-set VPN-SET
```

6. Apply the Crypto Map to the External Interface, Attach the crypto map to the physical interface that routes GRE traffic.

```
interface GigabitEthernet0/0
 crypto map VPN-MAP
```

Must be the physical interface used as the GRE tunnel source, Never apply crypto maps to the tunnel interface.

Pre-Requisites for IPsec over GRE:

- GRE tunnel must be fully functional first (ping test recommended).
- IPSec crypto map should never be configured under the tunnel interface.
- Transform-set and ISAKMP policy parameters must match on both sides.
- Use show crypto isakmp sa and show crypto ipsec sa for verification.

IKEv2 Site-to-Site IPsec Configuration Steps

1. Define Interesting Traffic, Defines traffic to be encrypted.

```
ip access-list extended <ACL-NAME>
 permit ip <LOCAL-SUBNET> <WILDCARD> <REMOTE-SUBNET> <WILDCARD>
```

2. IKEv2 Keyring (Pre-shared Key)

```
crypto ikev2 keyring <KEYRING-NAME>
 peer <PEER-NAME>
  address <PEER-IP>
  pre-shared-key <KEY>
```

3. IKEv2 Proposal (IKE SA Parameters)

```
crypto ikev2 proposal <PROPOSAL-NAME>
 encryption <ENCRYPTION-METHOD>
 integrity <INTEGRITY-METHOD>
 group <DH-GROUP>
```

4. IKEv2 Policy

```
crypto ikev2 policy <POLICY-NAME>
 proposal <PROPOSAL-NAME>
```

5. IKEv2 Profile

```
crypto ikev2 profile <PROFILE-NAME>
 match identity remote address <PEER-IP> 255.255.255.255
 authentication remote pre-share
 authentication local pre-share
 keyring <KEYRING-NAME>
 dpd <INTERVAL> <RETRY> on-demand
```

6. IPsec Transform Set (IPsec SA)

```
crypto ipsec transform-set <SET-NAME> esp-aes 256 esp-sha256-hmac
 mode tunnel
```

7. Crypto Map

```
crypto map <MAP-NAME> 10 ipsec-isakmp
 set peer <PEER-IP>
 set ikev2-profile <PROFILE-NAME>
 set transform-set <SET-NAME>
 match address <ACL-NAME>
```

8. Apply to WAN Interface

```
interface <WAN-INTERFACE>
 crypto map <MAP-NAME>
```

Apply to the external interface where encrypted traffic exits.

TLDR:

1. **ACL** – Defines traffic to encrypt
2. **Keyring** – Stores pre-shared keys for peers
3. **IKEv2 Proposal** – Crypto settings for the IKE SA (parent tunnel)
4. **IKEv2 Policy** – Binds proposal, required by profile
5. **IKEv2 Profile** – Handles peer authentication and validation
6. **Transform Set** – Crypto settings for IPsec SA (data plane)
7. **Crypto Map** – Binds ACL + IKEv2 + IPsec settings
8. **Interface** – Apply crypto map to WAN-facing interface

LISP Overview Summary

Purpose of LISP

Designed to solve Internet routing scalability by separating:

- EID (Endpoint Identifier) = *who you are* (logical identity)
- RLOC (Routing Locator) = *where you are* (network location)

Reduces routing table bloat in the core by shifting routing state to control plane lookups

LISP Roles

ROLE	FUNCTION
EID	Non-routable endpoint IP (logical identity)
RLOC	Routable IP of xTR (location used in outer IP header)
ITR	Encapsulates packets into LISP format (egress)
ETR	De-encapsulates packets (ingress)
XTR	Combo of ITR + ETR (most common in practice)
PITR	Encapsulates packets from non-LISP → LISP (for Internet ingress)
PETR	De-encapsulates packets from LISP → non-LISP (for Internet egress)
PXTR	PITR + PETR (bridges LISP↔non-LISP)
MS	Map Server: Stores EID→RLOC mappings registered by ETRs
MR	Map Resolver: Receives Map-Requests from ITRs, forwards to correct ETR
MS/MR	Combo of above (typical deployment)

Fundamentally, ETRs are the receivers whilst ITRs are the senders (Due to LISP Architecture), this is important to note throughout this prcoess, the ETRs cannot encapsulate, therefore they cannot be source of traffic, so traffic path always moves from “Left to right” or more accurately, ITR to ETR

Control Plane (Pull Model)

ITRs do not carry full routing tables.

Instead, they:

1 - Send Map-Request to MR for unknown EID.

2 - Receive Map-Reply from authoritative ETR.

3 - Cache the EID→RLOC mapping (TTL-based).

Like DNS for routing, allowing massive scaling.

Analogy: LISP = Pull-based (like PIM Sparse) vs traditional Push-based routing.

Data Plane

Packets are IP-in-UDP encapsulated:

- UDP Destination Port: 4341
- Random Source Port = entropy for ECMP
- Encapsulation occurs at ITR, de-encapsulation at ETR

Similar in function to GRE/IPsec tunnels, but dynamically built based on mapping system

Mobility and Multihoming

Since EID stays constant, LISP supports:

- Mobility: Only RLOC updates — no IP renumbering
- Multihoming: Multiple RLOCs per EID, LISP manages selection

Only Map-Server needs full visibility — sites (xTRs) operate with minimal config/state

Deployment Notes

Works great in hub-and-spoke topologies (central MS/MR)

Fits into SD-Access as its underlay/control mechanism

PxTRs are essential when integrating LISP with traditional Internet/legacy domains

LISP Packet Summary

Encapsulation: LISP wraps the entire original IP packet in a new IP + UDP header.

Outer IP Header:

- Src IP = Source RLOC (local xTR/PxTR)
- Dst IP = Destination RLOC (remote xTR/PxTR)

UDP Header:

- Dst Port = 4341 (signals LISP data packet)
- Src Port = Randomized (used for ECMP load balancing)
- Transport Protocol: UDP, Chosen for being stateless, lightweight, and NAT/ECMP friendly, TCP not used to avoid state tracking and head-of-line blocking

Inner Payload: The original IP packet (can be TCP, UDP, etc.)

IP interoperability - Fully supports IPv4-over-IPv6, IPv6-over-IPv4

VXLAN

VXLAN (Virtual Extensible LAN) is a Layer 2 overlay protocol that allows Ethernet segments (VLANs) to be extended across a Layer 3 IP network. It's designed to overcome traditional VLAN and STP limitations in large-scale data centers.

Why VXLAN?

- VLAN ID limit: Traditional VLANs support only 4096 segments (12-bit). VXLAN uses a 24-bit VNI, supporting ~16 million segments.
- No STP: VXLAN runs over Layer 3 → no spanning tree required, better link utilization.\
- MAC Mobility: Supports workload mobility by preserving MACs across sites.
- ECMP Support: Because it uses IP underlay, VXLAN supports Equal-Cost Multipath Routing for scalability and performance.

Encapsulation Summary

GRE for example, Encapsulates IP packets, strips MAC headers (L3 overlay).

VXLAN, Encapsulates entire Ethernet frames (MAC-in-UDP/IP), preserving original L2 headers (L2 overlay).

Think of GRE like:

> *"Open the letter, copy the message into a new envelope."*
> *Only the message (IP) survives.*

Think of VXLAN like:

> *"Take the entire original envelope (MAC + payload), put it in a padded shipping box (UDP/IP), and send it to the destination intact."*

Key VXLAN Components

- VTEP (VXLAN Tunnel Endpoint): Encapsulates/decapsulates Ethernet frames.
- VNI (VXLAN Network Identifier): 24-bit ID, maps VLAN ↔ VNI on the VTEP.

Control Plane (optional):

- EVPN with MP-BGP – recommended
- LISP – legacy
- Flood-and-learn – not scalable (Static)

TL;DR

VXLAN is an Ethernet-in-UDP overlay that extends Layer 2 across Layer 3 using VNI encapsulation, enables massive scalability, and allows multi-path, loop-free data center fabrics without STP. Think of it as putting Ethernet frames in IP shipping boxes.

* This is all is needed for VXLAN at ENCOR, More VXLAN related topics are in additional topics.

Wireless Theory

Unlike electrical or optical cables—where electricity or light travels along a physical medium—**wireless technologies rely on electromagnetic waves** to transmit data through the air.

A simple analogy is dropping a pebble into a still lake: circular waves propagate outward from the impact point. Similarly, **electromagnetic waves radiate** from a transmitting antenna and propagate through space.

Wave Properties

Electromagnetic waves **oscillate** in a repeating pattern, forming peaks and valleys as they travel. These oscillations can be described by several properties:

- **Wavelength (λ)**: The distance between two identical points on consecutive cycles (e.g., peak to peak).
- **Frequency (f)**: The number of complete wave cycles (passes through a baseline) in one second, measured in **Hertz (Hz)**.
- **Amplitude**: The height of the wave from the baseline, representing signal strength or power.
- **Baseline**: A reference axis from which the wave's oscillation is measured.

Example: A wave completing **4 cycles per second** has a frequency of **4 Hz**.

Frequency and Signal Behavior

- **Higher frequencies** (more cycles per second) = **shorter wavelengths** = **faster data rates**, but **shorter range** and **higher attenuation**.
- **Lower frequencies** = **longer wavelengths**, **better penetration** through obstacles like walls, and **longer range**, but **lower throughput**.

NAME	ABBREVIATION	VALUE
HERTZ	Hz	1 cycle/sec
KILOHERTZ	kHz	1,000 Hz
MEGAHERTZ	MHz	1,000,000 Hz
GIGAHERTZ	GHz	1,000,000,000 Hz

WAVE TYPE	FREQUENCY RANGE	USE CASES / NOTES
X-RAYS	~10^17 Hz	Medical imaging, radiation
VISIBLE LIGHT	~10^15 - 10^16 Hz	Human vision, optical transmission
INFRARED	~10^12 – 10^14 Hz	Heat, thermal cameras, IR remotes
MICROWAVES	~1 GHz – 30 GHz	Wi-Fi (2.4, 5, 6 GHz), radar, satellite comms
RADIO WAVES	~30 kHz – 300 MHz	AM/FM radio, VHF/UHF TV, LTE/5G lower bands
SOUND WAVES	~20 Hz – 20 kHz	Human hearing range – *not electromagnetic*

Note: **Sound is not electromagnetic** — it's a **mechanical wave** requiring a medium (like air or water) to propagate.

Understanding Bands in Wireless Networking

In wireless communications, **bands** are used to simplify how we reference and manage different portions of the radio frequency spectrum. Instead of memorizing specific frequency ranges (e.g., 2.400 – 2.4835 GHz), engineers and vendors refer to **bands** like “2.4 GHz,” “5 GHz,” or “6 GHz.”

This abstraction makes **network design, device configuration, and troubleshooting** significantly easier. For instance, saying that a device operates in the **2.4 GHz band** instantly tells us it's using the legacy-friendly, longer-range spectrum, as opposed to the **5 GHz band**, which is known for high throughput and shorter range. Bands also help separate **use cases** (e.g., indoor vs outdoor), and **regulatory constraints** such as Dynamic Frequency Selection (DFS) are often enforced per band.

Understanding Channels within Bands

Within each band, the spectrum is divided into **channels**, which represent specific **center frequencies** used for data transmission. These channels are spaced apart to reduce **interference** between neighboring signals.

In the **2.4 GHz band**, there are 14 possible channels (region-dependent), spaced **5 MHz** apart. However, each channel occupies approximately **22 MHz** of bandwidth, meaning adjacent channels **overlap**, causing interference.

To solve this, networks typically use **non-overlapping channels** — most commonly **channels 1, 6, and 11** — which are spaced far enough apart to avoid this overlap.

In the **5 GHz and 6 GHz bands**, channels are typically spaced wider (20 MHz or more) and overlap is less of an issue due to greater spectral availability and tighter channel control (e.g., 20/40/80/160 MHz wide channels).

STANDARD	2.4 GHZ BAND	5 GHZ BAND	6 GHZ BAND	MAX PHY RATE	KEY FEATURES
802.11B	✓	✗	✗	11 Mbps	DSSS, legacy only
802.11G	✓	✗	✗	54 Mbps	OFDM, backward-compatible with 11b
802.11N	✓	✓	✗	600 Mbps	Dual-band, MIMO, channel bonding
802.11A	✗	✓	✗	54 Mbps	Early OFDM at 5 GHz
802.11AC	✗	✓	✗	~1.3 Gbps+	80/160 MHz, 256-QAM, beamforming
802.11AX	✓	✓	✓ (6E)	~9.6 Gbps (theoretical)	OFDMA, MU-MIMO, BSS Coloring

Phase - is a property that becomes relevant when comparing two or more waves. It describes the timing relationship or positional offset between them as they cycle through their waveforms. If waves are in phase (aligned), they strengthen each other (constructive interference). If they are 180° out of phase (opposite alignment), they cancel each other out (destructive interference). In simple terms, phase is the difference in "wave timing" between two signals.

Wavelength - is the real-world distance between wave cycles. It's inversely proportional to frequency: lower frequencies have longer wavelengths, and vice versa. For example, a 2.4 GHz wave has a wavelength of ~12.5 cm (~5 inches).

RF Amplitude and dB

In wireless communication, it's not enough to just send a signal — it must be strong enough to reach the receiver and be recognized above the noise floor. The strength of a signal is related to its amplitude — how "tall" the wave is — and determines how far and how reliably the signal travels.

Analogy: Lake and Pebble, Imagine a calm lake:

- You drop a pebble at one end (transmitter), and ripples travel outward.
- A boat nearby (client) feels the wave strongly.
- A boat far away barely notices it — the wave has lost amplitude due to distance and resistance (air, water friction).

This represents path loss in wireless: distance and environmental factors reduce signal strength.

RF Power – Milliwatts and Watts

Wireless transmitters (like APs or radios) use electrical power to generate these waves. That power is often measured in:

- Watts (W) — large systems like AM radio towers (e.g., 50,000 W)
- Milliwatts (mW) — small systems like Wi-Fi APs (e.g., 100 mW)

Example:

- A typical Wi-Fi AP might transmit at 100 mW
- A low-power IoT device might transmit at 1 mW

Comparing Power – Logarithmic Scale (dB)

Comparing large power values directly (like 1000× difference) is hard to grasp, so we use decibels (dB) — a logarithmic scale that shows power ratios.

The easiest way i found is as follows –

dB = 10 * Log10(P2/P1)

- P1 = Reference power (Base-line)
- P2 = Power of interest (What you're comparing to)
- Log10 = base-10 Logarithm

Example –

P1 - Reference point (sender) - 52mW

P2 - Intrest (Client) - 12mW

to calculate dB =

Step 1: 12/52 = 0.230

Step 2: Log(0.230) = -0.638

*Step 3: 10 * -0.638 = dB*

dB = -6.38

This means P1's transmitting power is 6.38 dB higher then P2.

Calculating Signal Loss

In real-world wireless networks, electromagnetic signals weaken as they travel through the air. This weakening is called signal loss, and it's caused by factors like:

- Distance between transmitter and receiver
- Air resistance and environmental absorption
- Obstacles like walls, people, or furniture
- Device hardware limitations

Because of this, it’s important to ensure that enough power is transmitted to overcome these losses — while also staying within regulatory limits.

Why We Use dB

Rather than comparing power levels in raw milliwatts (which get hard to manage when large or small), we use dB (decibels) to measure the ratio between what was transmitted and what was received.

This makes it easy to:

- Compare how much signal was lost
- Make adjustments to Tx power or antenna gain
- Plan AP placement for consistent signal quality

Example:

Let’s say an AP transmits at 100 mW, and a client 15 meters away receives 2.5 mW.

We can calculate the dB loss like this:

Sent: 100 mW

Received: 2.5 mW

Step 1: 2.5 / 100 = 0.025

Step 2: log10(0.025) = -1.6

*Step 3: 10 * -1.6 = -16 dB*

This means there is a loss of 16 dB between the AP and the client. The signal was reduced to 1/40th of its original strength due to air resistance, distance, and obstructions.

dB reflects **amplitude**, not frequency.
You can have **high amplitude at any frequency** — but **lower-frequency waves** benefit more from strong amplitude due to **better propagation** characteristics.
That’s why 2.4 GHz Wi-Fi often seems “stronger” than 5 GHz, even at equal power.

EIRP – Effective Isotropic Radiated Power

In real-world wireless systems, signal strength isn't just about what the transmitter outputs — it's affected by everything between the transmitter and the air, and even how the antenna shapes the wave, This total radiated power is called EIRP.

Why EIRP Matters

EIRP tells you the real dB output of your system — the actual amplitude the wave presents in space.

It's critical because:

- Government agencies (like FCC, ETSI) enforce maximum EIRP limits
- It determines how far your signal can travel

It's key in point-to-point wireless links, mesh setups, and outdoor deployments

EIRP Formula

EIRP (dBm) = Tx Power (dBm) - Cable Loss (dB) + Antenna Gain (dBi)

Example 1 – Simple Setup

- Tx Power: 10 dBm
- Cable Loss: 5 dB
- Antenna Gain: +8 dBi

EIRP = 10 - 5 + 8 = 13 dBm

This is the final wave amplitude, not just what the transmitter pushed out, Rather the dB that comes out of the Antenna

Example 2 – Full Link Budget (Point-to-Point)

Let's say you have:

◈ Site A (Transmitter):

- Tx Power = 10 dBm
- Tx Cable Loss = 4 dB
- Tx Antenna Gain = 9 dBi

◈ Site B (Receiver):

- Rx Antenna Gain = 6 dBi
- Rx Cable Loss = 3 dB

✈ Free-Space Path Loss:

2.4 GHz @ 1 km ≈ 100 dB

Link Budget Calculation:

EIRP (Site A) = 10 - 4 + 9 = 15 dBm

Received Power = 15 - 100 + 6 - 3 = -82 dBm

This tells you the signal arrives just above the minimum sensitivity of many receivers (~-85 dBm).

FSPL – Free Space Path Loss

FSPL (Free Space Path Loss) is a way to quantify how much signal strength is lost purely due to propagation through space — no walls, cables, or interference, just distance and frequency.

Key Properties of FSPL:

FSPL is logarithmic (exponential) — signal strength drops sharply near the transmitter and flattens over longer distances.

It depends only on:

- Distance (in meters or km)
- Frequency (in MHz)

Why FSPL Is So Important

FSPL defines the baseline loss you *always* have, even in perfect air.

It's arguably the most important loss to consider in AP design.

Real-world FSPL varies due to:

- Weather (rain, fog)
- Air moisture/density
- Obstructions (trees, poles, buildings)
- Ground reflections (especially for long outdoor links)

Even at 1 meter, FSPL for 5–6 GHz can exceed 40 dB loss, especially with short high-frequency wavelengths.

Real-World Test Comparison

A test was performed using 2.4 GHz, 5 GHz, and 6 GHz Wi-Fi:

BAND	DISTANCE UNTIL -67 DBM RSSI
2.4 GHZ	~140 feet (42 meters)
5 GHZ	~80 feet (24 meters)
6 GHZ	~50 feet (15 meters)

This test highlights:

- Higher frequencies degrade faster over distance
- Lower bands (like 2.4 GHz) provide better range and penetration
- All other factors being equal, FSPL alone limits 5G/6G reach

FSPL is a fundamental physical limitation — not a design flaw.
It reminds us that frequency and distance always work against you in wireless design.
Knowing how to estimate FSPL is key to planning for EIRP, antenna gain, and client coverage.

Power Levels at the Receiver – Understanding RSSI, Sensitivity, and Noise Floor

In Wi-Fi communication, the **transmit power** (EIRP) from LAN devices typically ranges from:

- **20 dBm (100 mW)** down to **0 dBm (1 mW)**

But due to **free space path loss (FSPL)**, the **receiver** often sees only a **fraction of a milliwatt**, sometimes even as low as **nanowatts**.

Despite this, the **client device** must still be able to **decode the signal** and extract usable data. This is where **RSSI** and **receiver sensitivity** come in.

RSSI (Received Signal Strength Indicator)

RSSI is a **relative scale** (typically 0–255) used by receivers to indicate how strong the incoming signal is.

Important: RSSI is **not standardized**. Each vendor (e.g., Cisco, Intel, Dell) interprets the scale differently.

Many systems **convert RSSI to dBm** for clarity (e.g., -30 dBm = strong, -85 dBm = weak).

RSSI is affected by:

- Distance from AP
- Antenna gain
- FSPL and environmental loss

Receiver Sensitivity

Sensitivity is the **minimum signal strength (in dBm or RSSI)** at which a device can correctly decode data, It **varies per vendor and chipset**.

Example: A Dell wireless card might define **RSSI 120** or **-85 dBm** as the lowest usable signal.

If signal falls **below sensitivity**, the frame is **dropped or retried**.

Noise Floor

The **noise floor** is the ambient background RF energy — not signal, just **environmental "noise"**.

Sources:

- Microwaves
- Nearby Wi-Fi networks
- Bluetooth, Zigbee, even poorly shielded electronics

Measured in **dBm** (e.g., -90 dBm), Noise floor is **added on top of FSPL** — it doesn’t replace it., If your signal is **too close to the noise floor**, it can’t be decoded

Carrying Data over RF – Modulation Concepts

In wireless communication, the RF signal used to transmit information is called a carrier signal. The carrier itself doesn't contain meaningful data until it's modulated.

What is Modulation?

Modulation is the process of altering the properties of the carrier wave to encode data, while still maintaining the core frequency. The modulation process is:

- Performed by the transmitter
- Reversed by the receiver using demodulation

Both the sender and receiver must agree on the modulation scheme, or the data cannot be correctly interpreted.

Modifiable Wave Properties

Modulation only adjusts specific attributes of the RF wave:

- Amplitude (Strength of the wave)
- Frequency (cycles per second, only slightly)
- Phase (Timing / angle of the wave)

Spread Spectrum

To send data more efficiently and reliably, we use something called **spread spectrum** — this means we spread the signal across a wider chunk of the available frequency range. There are two main types:

1. DSSS, mostly used in older bands (2.4GHz), the signal is spread over a wider bandwidth from the center channel, purposefully overlapping other channels, it sends each bit of data more then once, resulting in increasing resiliency

2. OFDM, used in most modern bands (5GHz, 6GHz and 2.4GHz), it takes a single channel and splits it into sub-channels, each wave uses one of the smaller sub-channels, many waves are sent at once paralelly, resuling in increased data transfer.

Both DSSS and OFDM require that the sender and receiver both “understand” how to extract the data.

Wi-Fi Standard	*Wi-Fi Gen*	*Bands*	*Max Data Rate*	*Channel Widths*
802.11b	Wi-Fi 1	2.4 GHz	11 Mbps	20 MHz
802.11a	Wi-Fi 2	5 GHz	54 Mbps	20 MHz
802.11g	Wi-Fi 3	2.4 GHz	54 Mbps	20 MHz
802.11n	Wi-Fi 4	2.4 & 5 GHz	600 Mbps	20 / 40 MHz
802.11ac	Wi-Fi 5	5 GHz	6.93 Gbps	20 / 40 / 80 / 160 MHz
802.11ax	Wi-Fi 6	2.4 & 5 GHz	9.6 Gbps	20 / 40 / 80 / 160 MHz
802.11ax (6 GHz)	Wi-Fi 6E	6 GHz	9.6 Gbps	20 / 40 / 80 / 160 MHz
802.11be	Wi-Fi 7	2.4 / 5 / 6 GHz	23.05 Gbps	20 / 40 / 80 / 160 / 320 MHz
802.11bn (future)	Wi-Fi 8	2.4 / 5 / 6 GHz	100 Gbps (projected)	Likely up to 320 MHz

When picking APs, match them to the clients’ capabilities. The AP must speak each client's language — meaning it must support their Wi-Fi standard and features.

Spatial Multiplexing

In the past, devices had only one transmitter and one receiver — this is called **SISO** (Single Input, Single Output). It meant there was only **one radio chain** to send and receive data.

A key reason for faster speeds in modern Wi-Fi is the use of **MIMO** (Multiple Input, Multiple Output). This means devices have **multiple transmitters and receivers**, allowing them to send and receive data over **several radio chains at once**.

A simple way to look at it:

- **SISO** is like connecting two switches with a **single cable**.
- **MIMO** is like connecting them with **multiple cables**, increasing both speed and reliability.

The process of using all these radio chains at the same time is called **spatial multiplexing**.

Wi-Fi standards like **802.11n, ac, and ax** all use MIMO in different ways. For example:

- A **2x2 MIMO** device has 2 transmitters and 2 receivers.
- If both the AP and client are 2x2, they can use **2 spatial streams**, sending two data streams in parallel.

Example –

Assume I'm using **2.4 GHz** over a **1 km** distance.

Using **SISO**, I'll get a baseline of **X Kbps or Mbps**, depending on conditions (signal strength, interference, etc.). That's **one radio chain**, so the throughput is limited.

With **MIMO**, I can boost performance by sending and receiving multiple streams in parallel:

- **2x3 MIMO** → 2 transmitters, 3 receivers
  Focus is on improving **reception reliability** — extra receivers help catch more of the signal, especially over longer distances or with signal reflections.
- **3x2 MIMO** → 3 transmitters, 2 receivers
  Focus is on **throughput** — sending more data in parallel (up to 2 spatial streams), even if the receive side is more limited.

The actual number of spatial streams used is the **lower of the Tx and Rx counts** — so both 2x3 and 3x2 setups would max out at **2 spatial streams**, but with different strengths: one optimized for reception, the other for sending capacity.

Spatial streams are sent over the same frequency, and while their modulation (amplitude, phase, etc.) can differ per stream, they are separated by spatial paths—not by using different frequencies.

Transmit beamforming is the technique of sending multiple signals over different antennas with carefully adjusted phases, so they combine in-phase at the receiver. This strengthens the signal and improves reliability. It's used to send the same traffic over different antennas with different phases to ensure the signals arrive together and leverage constructive interference.

Other receivers get the out-of-sync waves, usually either at lower quality or outright un-readable.

Maximal Ratio Combining (MRC) – Summary Explanation

MRC is a **receive diversity technique** that improves wireless signal reliability by **combining multiple received copies** of the *same transmission*, each distorted differently due to noise, fading, and FSPL.

- Each signal copy is **weighted by its SNR** and **aligned in phase.**
- The result is a **stronger, cleaner composite signal**.
- Think of it like **receiving several burned photos** through a fire — no single photo is perfect, but piecing together unburned parts from all gives you the original image.

MIMO gives you more lanes, MRC keeps each lane clean, OFDM spreads traffic intelligently across those lanes.

Dynamic Rate Selection (DRS)

DRS is a standard Wi-Fi mechanism that dynamically adjusts the **modulation and coding scheme (MCS)** to ensure optimal throughput and reliability based on **RSSI and SNR**.
Vendors may enhance DRS with proprietary tuning, but its core behavior is defined by the **802.11 standard**.

Autonomous Wireless Deployment Model

Autonomous Access Points (APs) operate as standalone devices, each managing its own configuration, SSIDs, and client associations. They support multiple SSIDs mapped to different VLANs and connect to the LAN using trunk ports. Each AP has a management IP and is configured independently, similar to a traditional switch or router.

While they offer a low-latency, direct data path without tunneling overhead (e.g., CAPWAP), they are not scalable for large deployments due to manual provisioning and lack of centralized management. Autonomous APs are best suited for small networks or isolated wireless segments.

Lightweight Wireless Deployment Model

Lightweight Access Points (APs) delegate their control plane to a centralized Wireless LAN Controller (WLC), connecting over the network via a **CAPWAP** tunnel. While they retain the ability to handle time-sensitive 802.11 tasks locally, advanced features such as configuration, security policies, and mobility are managed by the WLC.

This model simplifies large-scale management and enables centralized policy enforcement. However, it introduces reliance on the controller, and data traffic may traverse a suboptimal path if fully tunneled through the WLC, potentially impacting performance.

Both **Lightweight** and **Autonomous** Access Points serve different roles in wireless deployments:

- **Autonomous APs** offer full local control and independent operation, making them ideal for small or isolated networks, but are difficult to scale and manage in large environments.

- **Lightweight APs**, managed by a central **WLC**, simplify large-scale deployments with unified management and policy enforcement, but depend on the controller and may introduce suboptimal data paths due to tunneling.

Each model has trade-offs between **control, scalability, and performance**, and should be chosen based on network design requirements.

Cisco Lightweight AP Operational Modes

1. **Local Mode**
   - **Default mode** for lightweight APs.
   - Serves clients using one or more BSSs on a single channel.
   - When idle, it scans other channels for **noise, interference**, and **rogue detection** (background scanning).
2. **FlexConnect Mode**
   - Designed for **remote sites** with WAN-based WLCs.
   -
   - Maintains a **CAPWAP control tunnel** to the WLC.
   - If the tunnel fails, AP continues **L2 switching** locally (SSID ↔ VLAN mapping) in **standalone mode**.
3. **Monitor Mode**
   - AP does **not serve clients**.
   - Used for **WIPS/WIDS**, rogue detection, and site surveys.
   - Scans all channels continuously.
4. **Sniffer Mode**
   - Acts like a **dedicated packet capture device**.
   - Captures 802.11 traffic and forwards it to a **remote analyzer** (e.g., Wireshark).
5. **Rogue Detector Mode**
   - AP doesn't transmit.
   - Monitors wired and wireless MAC addresses.
   - Flags devices whose MACs appear in **both planes** as potential **rogues** (MAC spoofing).
6. **Bridge Mode**
   - Enables **point-to-point** or **point-to-multipoint** wireless bridging.
   - Common in **outdoor mesh** or **wireless backhaul** scenarios.
7. **Flex + Bridge Mode**
   - Combines **FlexConnect features** with **mesh networking**.
   - Useful for **branch sites** needing both bridging and WAN-failure survivability.
8. **SE-Connect (Spectrum Expert) Mode**
   - AP acts as a **spectrum analyzer**.
   - Used for **RF analysis** and interference detection.
   - Streams data to apps like **Cisco Spectrum Expert** or **MetaGeek Chanalyzer**.

Wireless Deployment Models

It's important to remember that **latency is critical** between APs and the WLC. If **RTT exceeds 100ms**, the AP may **drop the CAPWAP tunnel**, assuming the WLC is unreachable, and attempt to join another available controller.

Cisco supports four main wireless deployment architectures:

Centralized WLC

Description: A single WLC placed at a central location serving all lightweight APs.

Best for: Small to medium enterprises or single-site deployments.

Advantages:

- Cost-effective
- Centralized control and management
- Secure policy enforcement
- Scalable (within reason)

Disadvantages:

- No local redundancy unless paired with HA
- Adds latency if APs are physically distant
- Requires careful WLC placement (≤100ms RTT)

Cloud-Based WLC

Description: Controller(s) hosted in the cloud or remote data center.

Best for: Organizations with many remote branches and limited IT staff.

Advantages:

- No physical WLC hardware needed
- Flexible cloud scalability and updates
- Central management for remote sites

Disadvantages:

- APs must operate in **FlexConnect local switching**
- Heavily dependent on WAN/ISP availability
- Latency-sensitive (control traffic only)

Distributed WLCs

Description: Multiple WLCs distributed across regions or core sites.

Best for: Large enterprises with many APs across geographic regions.

Advantages:

- Low latency to nearby Aps
- Improved scalability and client distribution
- Supports controller redundancy and roaming

Disadvantages:

- High deployment and licensing cost
- More complex WLC management and mobility design
- Requires robust synchronization (e.g., mobility groups)

Controller-less (Embedded WLC)

Description: The WLC function is embedded in one AP (e.g., Mobility Express).

Best for: Small offices or single-floor environments.

Advantages:

- No separate WLC appliance needed
- Cost-effective for small deployments
- Simple setup

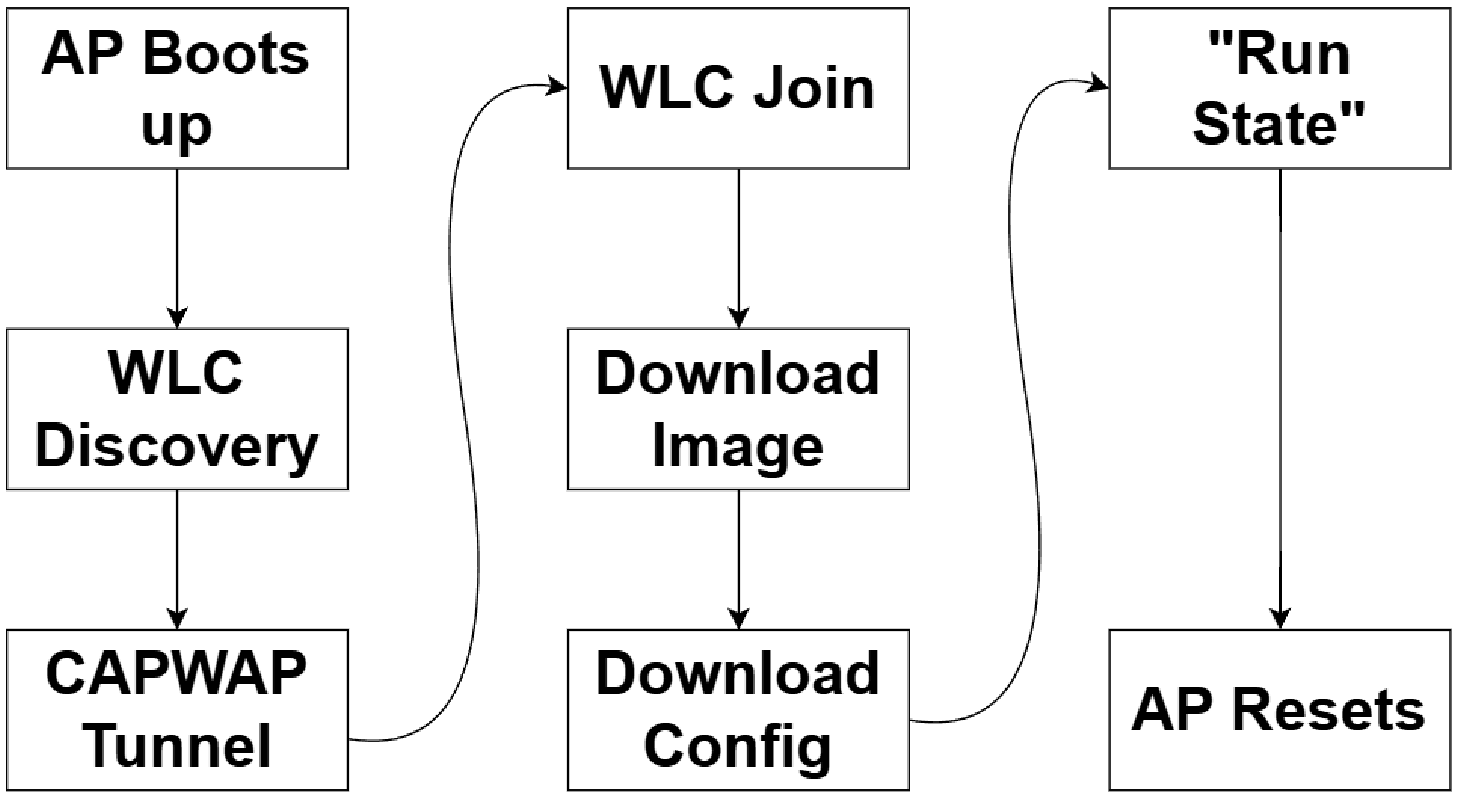
Disadvantages:

- Limited scalability and performance
- No true high availability
- Lacks advanced features of dedicated WLCs

Pairing Lightweight APs with WLC

 AP Boots

1. Initializes radios, hardware checks, loads IOS image.
   - Attempts to obtain IP address via DHCP.
   - Resolves WLC via: DHCP Option 43, DNS (CISCO-CAPWAP-CONTROLLER), static config, or broadcast.
2. WLC Discovery
   - AP sends Discovery Requests to potential WLCs.
   - WLCs respond with Discovery Responses.
   - AP builds a WLC candidate list (sorted by priority, capacity, or response time).
3. CAPWAP Tunnel Creation
   - AP selects a WLC and initiates a CAPWAP control connection (UDP port 5246).
   - This is a control tunnel only at this stage.
4. Join Request
   - AP sends a CAPWAP Join Request to the selected WLC.
   - WLC responds with a CAPWAP Join Response, accepting the AP.
5. Image Check & Download (if needed)
   - AP compares its code version with the WLC's.
   - If mismatched, it downloads the image from the WLC, reboots, and restarts the join process.
6. Configuration Download
   - Once the image matches, AP re-establishes CAPWAP, rejoins, and downloads:
     1 - AP Group settings
     2 - Radio configs
     3 - SSID/VLAN mappings
7. Run State
   - AP enters "RUN" state and begins:
   - Broadcasting SSIDs
   - Accepting client associations
   - Reporting to the WLC



WLC Discovery

An AP can discover WLCs using the following methods:

Via LAN Broadcast/Multicast

- Sends a Layer 2 CAPWAP Discovery Request to the local subnet.
  Only functional if the WLC is in the same L2 domain (not routed).

Via Statically Configured Controllers *(Preferred)*

- APs can be configured with a Primary, Secondary, and Tertiary WLC IP.
- These are always tried first, in order of priority.

Via DHCP Option 43

- The DHCP server provides WLC IP addresses in Option 43 during the IP lease process.

Via DNS Lookup

- The AP performs a DNS query for CISCO-CAPWAP-CONTROLLER.localdomain.
  If resolved, the returned IP(s) are added to the WLC candidate list.

WLC Selection

Once discovery is complete, the AP selects a WLC using the following criteria:

Statically Configured WLC

- If a Primary/Secondary/Tertiary WLC is configured, the AP attempts those first, in order.

Previously Joined WLC

- If the AP has successfully joined a WLC in the past, it may prefer that WLC again.

Lowest-Loaded WLC (Dynamic Selection)

- Each WLC includes its current AP load in the Discovery Response.
- The AP evaluates this, along with other factors (e.g., RTT, capacity), to choose the least-loaded viable WLC.

WLCs can reject CAPWAP Join requests if they've reached their maximum AP capacity. To manage this, APs can be assigned **join priorities**. When oversubscribed, a WLC will **prefer higher-priority APs**, even dropping lower-priority ones to make room.

WLC High Availability

Lightweight APs depend on WLCs for operation, making **controller redundancy critical**. Without it, a single WLC failure (e.g., managing 3000 APs) can lead to:

- All APs attempting to join other WLCs from their candidate list
- **Flooding and overloading** of neighboring WLCs
- **Unpredictable, non-deterministic behavior**
- Clients being disconnected or unable to associate

While configuring **Primary, Secondary, and Tertiary WLCs** improves determinism, it's not scalable for large environments.

Recommended: WLC High Availability with SSO

- Two WLCs form an **HA pair** (active/standby) using **SSO (Stateful Switchover)**
- APs maintain **CAPWAP tunnels to the active WLC**
- Active WLC continuously synchronizes:
    - AP join state
    - Client session information
    - Configuration and image data
- Upon failure, standby **instantly takes over**, maintaining AP and client sessions with **no reboot or reassociation**

This model offers the most **scalable, deterministic, and seamless** HA behavior for enterprise WLANs.

By default, APs send "Heartbeat" message (Keepalives) every 30 seconds to the WLC to ensure its up, its recommend to decrease this number to ensure faster reconvergance.

Wireless Configuration Management – IOS XE WLC (Tag-Based Architecture)

Once an AP **joins a WLC** and the **software image is validated/matched**, it enters the **configuration download phase**.

Modern IOS XE-based WLCs (e.g., Catalyst 9800 series) introduce a **modular, object-oriented** approach to configuration, replacing the rigid structure of **AireOS**.

In IOS XE-based WLCs, the configuration is modular and driven by **tags**, which act as **binding layers** between APs and profiles. There are **three main tag types**, each determining a different functional aspect of an AP's behavior.

Each tag **maps to one or more profiles**, and the **APs are assigned tags** — either manually or dynamically — to determine their behavior.

TAG TYPE	PROFILES MAPPED	PURPOSE
SITE TAG	AP Join Profile + Flex Profile	Determines AP mode (e.g., Local or FlexConnect), timers, image mgmt, logging
RF TAG	RF Profiles (per band)	Controls per-band settings like Tx power, channel, data rates, modulation
POLICY TAG	WLAN Profile + Policy Profile	Defines SSID properties (WLAN) and associated policies (VLAN, ACL, QoS, etc.)

Profile Summary

- **AP Join Profile**: Heartbeat timers, image options, NTP, syslog, etc.
- **Flex Profile**: VLAN mappings, local switching/NAT, WAN resilience, DHCP
- **RF Profiles (2.4/5/6 GHz)**: Band-specific RF behavior (Tx power, DCA, TPC, etc.)
- **WLAN Profile**: Defines SSID, security (WPA2/3, PSK, 802.1X), QoS settings
- **Policy Profile**: Maps WLAN to VLAN, applies ACLs, multicast, AVC, URL filters

Configuration Workflow

1. **Create AP Join Profile** and (optionally) **Flex Profile**, map them to a **Site Tag**
2. **Create RF Profiles** for 2.4, 5, and 6 GHz; map them to an **RF Tag**
3. **Create WLAN Profile** and **Policy Profile**; bind them together under a **Policy Tag**
4. **Assign Site Tag**, **RF Tag**, and **Policy Tag** to each AP (manually or via policy rules)

Tag-Based AP Configuration Example

Site Tag: Flex-Site

- Mode: FlexConnect
- Native VLAN: 10
- Local switching enabled

RF Tag: 2.4GHz-RF

- Enabled: 2.4 GHz only
- Tx Power: Low
- Min data rate: 6 Mbps

Policy Tag: Corp-WiFi

- WLAN Profile: Corp-SSID
- Policy Profile: VLAN 20, ACL: Allow_Internet

AP01 receives: Flex-Site, 2.4GHz-RF, Corp-WiFi
AP02 (same site, different band) receives: Flex-Site, 5GHz-RF, Corp-WiFi

Both APs receive the same Site Tag and Policy Tag but due to the flexibility of IOS-XE, A different RF Tag can be used on both, each would serve different clients whilst receiving the same configurations of the Site & Policy.

Wireless Antennas

Antennas vary in design to suit specific wireless deployment needs. Choosing the right antenna is critical for effective coverage and performance.

A key factor is **client density** — the number of users per AP. High-density areas (e.g., stadiums) require antennas and AP placement strategies that optimize airtime and distribute client load efficiently. Proper design ensures balanced coverage and capacity across all users.

Antenna Gain

Antenna gain doesn't increase signal power — it **shapes** the signal to focus energy in a specific direction. Higher gain means **better directionality**, allowing the antenna to send or receive signals more effectively toward the target area.

Antenna Beamwidth

While **antenna gain** reflects how focused a signal is, **beamwidth** indicates the **angular spread** of that signal — measured in degrees. Narrower beamwidth = more focused direction = higher gain. Vendors specify beamwidth to help predict signal propagation and identify where clients should be positioned for **optimal signal strength and integrity**. Beamwidth is essential for **coverage planning** and **directional antenna placement**.

Antenna Types

Omnidirectional Antennas

- Radiate signals **equally in all horizontal directions** (like a donut shape).
- Common in general-purpose indoor deployments.
- Not suitable for long-range or focused coverage.
- Must be **upright** for proper signal propagation — improper alignment (e.g., bent) can cause signal to direct downward, reducing client performance.

Directional Antennas

- Focus signals in a **specific direction**, increasing gain and range.
- Used for **point-to-point links**, **long-range**, or **targeted coverage**.

Common Directional Types:

- **Yagi Antenna**: Egg-shaped directional beam, typically provides **10–15 dBi gain**.
- **Parabolic Dish**: Highly focused beam, used for long-distance links; can provide up to **30 dBi gain**.

Directional antennas offer **higher gain and range**, but require **precise alignment** toward the intended coverage area.

Wireless Roaming

Roaming Basics:
Wireless clients should maintain seamless connectivity as they move between APs. The decision to roam is made by the client, based on signal metrics like RSSI or SNR. When the signal degrades, the client scans for better APs and initiates a reassociation if needed.

Roaming with Different AP Modes

Autonomous APs:

- Each AP independently manages its BSS and client association table.
- When a client roams, the old AP may forward buffered frames over the wired network to the new AP.
- Roaming is typically constrained to a **Layer 2 domain**, as VLAN/subnet consistency must be preserved to avoid IP changes and DHCP renewal

Lightweight APs (Split-MAC):

- APs forward all client data and control traffic to a WLC via **CAPWAP tunnels**.
- The **WLC centrally maintains client state** and handles roaming events.
- Two types of roaming:
    - **Intracontroller Roam:** Between APs on the same WLC – reassociation and context switch takes <10 ms.
    - **Intercontroller Roam:** Between APs on different WLCs – more complex; can be Layer 2 or Layer 3

Reassociation Delays

Two main processes can add latency during roaming:

1. **DHCP Renewal:** May occur if the client believes the subnet has changed.
2. **Authentication:** Particularly if 802.1X/EAP is used, re-authentication with RADIUS can introduce delays.

Cisco Fast Secure Roaming Techniques

CCKM (Cisco Centralized Key Management)

- The WLC securely caches Pairwise Master Keys (PMKs).
- When a CCX-compatible client roams, the WLC sends the cached key to the target AP for validation.
- This skips full 802.1X re-authentication.
- **Client must support CCX/CCKM.**

PMK Caching

- The client retains keys from recently visited APs (~8 max).
- If it rejoins one of those APs, it reuses the cached PMK.
- Works best when APs are revisited frequently, such as in small/mid-size WLANs.

802.11r (Fast BSS Transition)

- A standards-based method.
- During initial authentication with RADIUS, the client and server derive a **PMK-R0**.
- The client then derives **PMK-R1** keys for each AP it might roam to.
- Enables **pre-authentication** before roaming.
- Supported even on non-Cisco clients (as long as they support 802.11r).

All fast roaming methods **require client support**. If the client lacks support for CCKM, 802.11r, or key caching, it will fall back to full authentication and experience higher roaming latency.

Cisco documentation refers to the overall goal of fast roaming as achieving **<50 ms** delay to support VoIP/real-time traffic — ideal is **<10 ms**.

METHOD	DESCRIPTION	NOTES
OPEN AUTHENTICATION	No real authentication — simply checks if the client is 802.11-capable.	No encryption; often used with captive portals (e.g., WebAuth).
WPA (1, 2, 3)	**Wi-Fi Protected Access** standards that define encryption and authentication mechanisms. All support two modes: • **PSK (Pre-Shared Key)** • **802.1X (Enterprise)**	- PSK: Single shared key, not scalable, prone to compromise. - 802.1X: Uses RADIUS for dynamic key generation.
WPA SECURITY NOTES	- **WPA1**: Uses TKIP; obsolete and insecure. - **WPA2**: Uses AES-CCMP; secure if using 802.1X. - **WPA3**: Introduces **SAE** (Simultaneous Authentication of Equals) for PSK, improving resistance to brute force and eavesdropping.	WPA3 is the most secure option currently available.
EAP (EXTENSIBLE AUTHENTICATION PROTOCOL)	A flexible framework used in **802.1X** authentication. Allows integration with RADIUS and many identity methods (certificates, passwords, etc.).	Supports features like **Pre-Authentication** and **Port-Based Access Control**.
WEBAUTH	Browser-based captive portal used for user acknowledgment or login (e.g., splash pages, guest Wi-Fi). Can be **local** or integrated with **RADIUS/LDAP**.	Not inherently secure; should be combined with HTTPS and optionally 802.1X.

Enterprise Network Architectures

Enterprise network architecture is the structured framework that defines how networking hardware and software components interconnect to support business services. The choice of architecture must always align with the organization's specific use case, scalability goals, physical constraints, and budgetary limitations.

It’s critical to understand: **you adapt the architecture to fit your operational needs—not the other way around.** Each architectural model offers clear advantages, but they also introduce trade-offs. Selecting the right model requires evaluating how those disadvantages may impact your users, applications, and future growth.

Hierarchical LAN Design

The Hierarchical LAN design model segments the enterprise network into structured layers to improve scalability, fault isolation, and administrative simplicity. By organizing devices and functions into distinct tiers, network performance and manageability are optimized.

This model is composed of three logical layers:

Access Layer

- Connects end devices (hosts, printers, IP phones) to the network.
- Typically implemented using switches such as Cisco Catalyst 9200 or 2960XR.
- Provides port security, QoS markings, and VLAN segmentation.

Distribution Layer

- Serves as the aggregation point for access layer switches.
- Enforces policy boundaries (routing, filtering, QoS, redundancy).
- Provides link and device redundancy and summarization to optimize the core.

Core Layer

- Focused on high-speed backbone transport between distribution blocks.
- Should be fast, highly available, and minimal in policy enforcement.
- Connects different parts of the enterprise, including data centers and WAN edges.

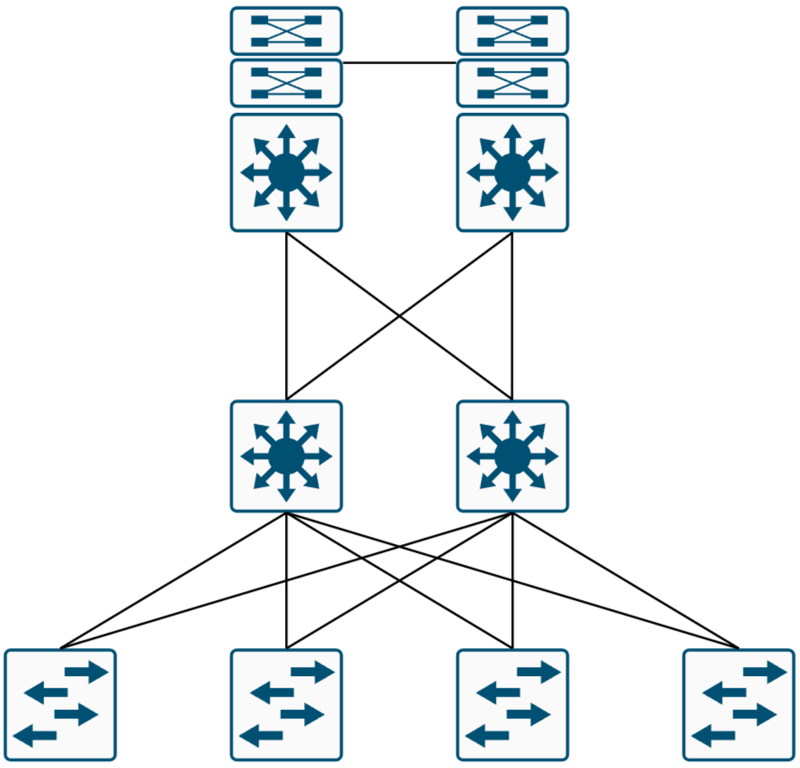
By logically separating these layers, enterprises can design networks that are modular, scalable, and easier to troubleshoot or expand. This layered architecture also supports the evolution to technologies like SD-Access and fabric-based networks.

Advantages

- **Modular and Easy to Manage**: Each layer has a clear function, allowing independent design and control.
- **Fault Isolation**: Failures can be contained within a single layer, improving resiliency and simplifying troubleshooting.
- **Scalability**: Access layers can be added in modular blocks, allowing growth without redesigning the entire network.

Disadvantages

- **Potential for Increased Latency**: Traffic traverses more hops (Access → Distribution → Core), which may introduce higher end-to-end latency.
- **Layer Interdependency**: Scaling the Access layer requires proportional scaling in the Distribution and Core layers (e.g., more uplinks, routing capacity).
- **Suboptimal ECMP and Path Selection**: Depending on how Layer 2/Layer 3 boundaries are defined, Equal-Cost Multipath (ECMP) and traffic flow efficiency may be limited.



Collapsed Core

The **Collapsed Core** design combines the **core** and **distribution layers** into a single functional layer. This is ideal for **small to medium-sized campuses** where full three-tier separation is not needed. It reduces both hardware and operational overhead while maintaining key services like routing, ACL enforcement, and high-speed inter-VLAN forwarding.

This model is composed of two logical layers:

Access Layer

- Connects endpoints (hosts, phones, printers)
- Handles **port security**, **VLAN tagging**, and **QoS marking**
- Uplinks to collapsed core via **L2 or L3 port-channels**, depending on deployment model

Collapsed Core (Distribution + Core)

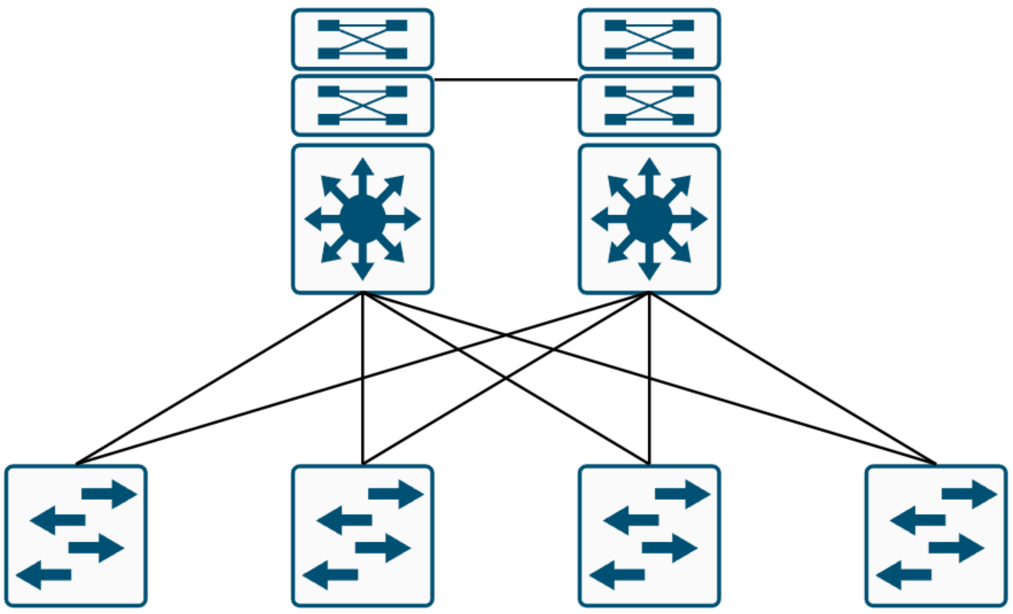
- Performs **routing**, **ACL enforcement**, and **inter-VLAN communication**
- Aggregates all access-layer switches
- Provides **redundancy**, **load balancing**, and **northbound connectivity** (to Internet, WAN, or data centers)

Advantages

- **Cost Effective –** Fewer switches and less hardware / software required.
- **Simpler to Manage -** Fewer devices to configure, maintain and monitor.
- **Lower Latency –** No core-to-distribution hops, direct routing.

Disadvantages

- **Reduced Scalability –** Limits growth due to lack of core abstraction.
- **Less Modularity –** Single point of policy and routing enforcment.
- **Weaker Fault Isolation –** No seperation of control planes like in 3-tier.



High Availability Network Design

High Availability (HA) in networking refers to implementing methods that maximize the probability of continuous network operation, even in the face of hardware, software, or link failures. HA can be broadly divided into two categories:

Network-Level High Availability, Focuses on the topology and architectural design to avoid single points of failure:

- Deploy redundant devices and links at each layer (Access, Distribution, Core).
- Avoid single points of failure (e.g., dual ISPs, redundant paths).
- Use virtualization technologies (e.g., VSS, StackWise Virtual) to simplify design while maintaining redundancy.

System-Level High Availability, Focuses on the resiliency of individual devices:

- Use modular devices with redundant components: PSUs, fans, supervisors, line cards, ASICs.
- Ensure hot-swappable capabilities for key modules.
- Implement Stateful Switchover (SSO) and Non-Stop Forwarding (NSF).
- Deploy protocol-level redundancy (e.g., UDLD for unidirectional detection, BFD for rapid failure detection).
- Use FHRPs (HSRP, VRRP, GLBP) with object tracking for intelligent failover.

High Availability Technologies

Stateful Switchover (SSO)

SSO allows seamless failover between two Route Processors (RPs) inside a redundant-capable router or switch:

- One RP is *active*, the other is in *hot-standby* mode.
- The active RP continuously checkpoints critical L2 control plane data (MAC tables, STP state, interface status, etc.) to the standby RP.
- Upon failure of the active RP, the standby takes over instantly, maintaining Layer 2 forwarding and minimizing impact.

However, SSO alone does not maintain Layer 3 protocol state, which can lead to routing protocol neighbor resets.

Non-Stop Forwarding (NSF)

NSF addresses the L3 limitations of SSO:

- Integrated with SSO, it preserves Layer 3 forwarding state (FIB and adjacency table).
- During an RP failover, the data plane continues forwarding packets using previously learned routes.
- Routing protocol neighbors (OSPF, EIGRP, BGP) are not immediately reset — instead, they continue forwarding while reconverging in the background.
- NSF allows non-disruptive switchover, but does not allow for learning new routes during the transition.

SSO/NSF with Graceful Restart (GR)

Graceful Restart (GR) is a standards-based protocol enhancement that complements **SSO/NSF** by coordinating with neighboring routers to **preserve routing protocol adjacencies** during a control plane switchover.

When a router with **NSF/SSO undergoes a Route Processor (RP) switchover**, **GR informs neighboring routers not to drop their adjacencies**, It ensures **no control-plane reset occurs**, maintaining L3 stability while the failing router recovers.

Key Behaviors

- **GR-aware neighbors** detect the NSF/SSO event and **enter a "helper mode"**, keeping the adjacency in a *temporarily valid state*.
- These neighbors continue to **forward traffic based on previously exchanged routing information**.
- If the restarting router **fails to reestablish the session** within a specific timeframe, the neighbor **tears down the adjacency** as a fail-safe.

Requirements for GR to Work

REQUIREMENT	EXPLANATION
NSF/SSO-CAPABLE ROUTER	Must support and be configured for stateful switchover
GR-AWARE NEIGHBORS	Must support GR (OSPF, BGP, EIGRP all support it)
PROTOCOL-SPECIFIC GR SUPPORT	GR must be enabled per protocol (e.g., bgp graceful-

SSO handles **internal hardware switchover**

NSF maintains **forwarding state (FIB/adjacency)**

GR ensures **external neighbor relationships remain intact**

Non-Stop Routing (NSR)

NSR is a **Cisco proprietary** high availability feature that extends the benefits of SSO by ensuring that **Layer 3 control plane protocols (OSPF, BGP, EIGRP)** also survive a Route Processor (RP) switchover — *without relying on neighbors*.

Because it is self contained, neighbors are not even aware of the event occuring, meaning it doesn’t need to rely on neighbor’s ability like with GR.

The main disadvantages of NSR are –

- Increased CPU/Memory usage, continuous L3 state checkpointing creates overhead, especially with large routing tables.
- Feature-specific support, must be enabled per protocol, and not all cisco platforms support this feature.

Generally speaking, for most deployment SSO/NSF + GR should suffice, it is important to configure them accordingly as they’re crucial for high availability.

SD-Access (Software-Defined Access)

What is Software-defined?

As networking progresses, the amount of devices that must be configured, deployed and monitor increase drastically, the amount of protocols, technologies and applications grow and change in faster pace and the policy flexibility for large enterprises gets more complex.

This results in more human error in configuration, more complex large scale infrastracture to maintain with application needs changing often at increasing pace and the need to track every policy, ACL and QoS alteration for every specific subnet or group.

To prevent these problems, "Software-Defined Networking" methods appeared, they aim to streamline deployments in large enterprises and datacenters, improve mobility, smarter automation, identity services, policy enforcements and more via a centrelized application.

SD-Access is a solution for large enterprises aimed at making the life of network engineers easier, It has 2 main components –

- Cisco Campus Fabric Solution
- Cisco DNA Center

The Campus Fabric includes technologies like VXLAN and LISP. When managed and automated through DNA Center's workflows (Design, Policy, Provision, Assurance), it becomes **SD-Access**.

The Cisco SD-Access is based on existing hardware and software, what makes it special is how the technologies are integrated and managed together (essentially, the application that streamlines it).

The SD-Access fabric can be divided into 4 parts –

Management Layer – Consists of Base-automation, Design, Policy, Provision and Assurance (and the GUI).

Controller Layer – NCP (network control platform), NDP (network data platform) and the ISE (Identity service engine).

Network Layer – SD-access Overlay network (LISP, VXLAN, CTS) and the underlay (Settings & Protocols such as IGPs).

Physical Layer – Hardware such as Switches, routers, wireless APs and WLCs, the Cisco DNA Center appliance/application and the ISE.

Physical Layer

The physical layer consists of all the devices that make up the overlay, underlay and the controller & Network layers, every device that participates in this orchestration must support the SD-Access fabric technologies, For example, since LISP is required for the control-plane, all devices that operate within the domain of control-plane must also understand LISP.

The "Domain" consists of the following –

- Switches – switches provide wired LAN connectivity to fabric and connect the edge hosts, multiple Catalysts are supported in addition to several Nexuses.
- Routers – Routers provide WAN connectivity and branch access to the fabric.
- Wireless – Cisco WLCs and APs provide connectivity to the fabric.
- Cisco controller appliances – Cisco DNA Center and Cisco ISE are required appliances / applications needed for the SD-access to operate.

Network Layer

The network layer is responsible for the underlay and the overlay, generally speaking, the underlay is responsible for actual data transfer, neighbor relationship and routing databases whereas the overlay is responsible for the abstraction of the underlay, which in layman terms means the overlay enables features such as host mobility or stretched VLANs using the underlay, overlays achieve it via tunnels that are invisible to the clients.

Underlay Network – the underlay network is responsible for the actual routing of traffic, its sole purpose is to get traffic from 1 point to another, which as a side-effect enables the Overlay to properly work.

Analogy – the underlay is essentially the road that the cars use to drive, the road doesn't care about the cars or who's in them, they just take them wherever they need to go.

Cisco recommends using IS-IS as the IGP for the underlay, whilst possible to use L2, its not recommended.

Generally speaking, any IGP that supports ECMP, fast convergence, IPv6 and IPv4 etc… is a solid candidate (such as OSPF).

There are 2 models of underlays, the manual model that is created via CLI & API (static configurations) and the automated underlay that also utilizes the Cisco DNA Center LAN automation feature.

Overlay Network – the "SD-Access" is the overlay fabric, it provides the aformentioned benefits such as host mobility, network segmentation and flexible policying, in SD-Access the fabric overlay is automated regardless of the method used for the underlay (Static vs automatic).

There are 3 basic planes of operation within the SD-Access –

- LISP For control-plane
- VXLAN for Data-plane
- Policy plane based on Cisco TrustSec

SD-Access Control Plane

LISP is used for the control-plane operations of SD-Access, this basically means that for routers that participate in SD-Access, to receive routes they must reach the MS/MR, which significantly reduces their routing tables, there are other benefits to SD-access with new iterations of LISP such as Anycast-Gateways, Fabric wireless and more jumbo mumbo.

SD-Access Data Plane

VXLAN is used for the data-plane operations of SD-Access, this is primarily because of several reasons, VXLAN is widely adopated and more acceptable, it allows for L2 VLAN stretching (whereas LISP doesn't) and generally speaking it supports better Host mobility via that stretching capability.

(LISP supports IP-in-IP whereas VXLAN supports MAC-in-IP (UDP)).

The VXLAN specification was also enhanced to support Scalable Group Tags (another feature of SD-Access) by adding 4 more Bytes in the VXLAN header, it can now carry up to 64,000 SGT Tags.

The new fields in the VXLAN GPO Packet include the following – Group Policy ID (16Bit), Don't Learn Bit (D) when set to 0 indicates the Egress VTEP shouldn't learn the source address of the encapsulated traffic and more.

SD-Access Policy Plane

Based on Cisco's TrustSec, SGT tags are assigned to authenticated groups of users or end devices, network policy such as QoS or ACLs are then applied universally based on the SGT Tags instead of IP Addresses, MACs etc… making policy far more flexible.

SD-Access Fabric Roles and Components

Control Plane Node

Performs LISP control-plane operations:

- Maintains the EID-to-RLOC mapping database
- Responds to mapping requests from ITRs
- Acts as Map-Server (MS) and Map-Resolver (MR)

Has no data-plane forwarding role

Fabric Border Node

- Connects the SD-Access fabric to external networks (non-fabric)
- Performs RLOC-to-non-fabric IP translation
- May act as a Proxy xTR (PxTR) to allow LISP ↔ non-LISP communication
- Enforces policy and segmentation at the fabric edge

Fabric Edge Node

- Connects wired endpoints to the SD-Access fabric
- cts as Anycast Gateway for connected hosts
- Performs LISP xTR functions (ITR/ETR)
- Handles authentication and SGT assignment via ISE

Fabric Wireless LAN Controller (WLC)

- Connects Lightweight APs to the fabric
- Extends SD-Access to wireless endpoints
- Participates in policy enforcement, SGT tagging, and host mobility
- Works in tandem with fabric-mode APs to encapsulate traffic in VXLAN

Intermediate Nodes

- Act as transit devices (e.g., distribution/core layer switches)
- Participate in the underlay routing (e.g., IS-IS or OSPF)
- Ensure IGP reachability between fabric nodes
- No overlay or LISP functionality — purely data-plane forwarding

Fabric Edge Node

The Fabric Edge Node is the network device that directly connects wired or wireless endpoints to the SD-Access fabric. It serves three major roles:

Default Gateway (Anycast Gateway)

- Hosts connect to the edge node using a shared SVI IP address configured identically on all edge nodes.
- Enables host mobility across the fabric — the gateway never changes even as the host moves.
- Uses the same virtual MAC/IP, ensuring seamless routing for mobile endpoints.

LISP xTR (ITR/ETR)

- Acts as an Ingress Tunnel Router (ITR):
- Queries the LISP Map-Resolver (MR) to find the destination RLOC for an endpoint.
- Acts as an Egress Tunnel Router (ETR):
- Registers endpoint EID-to-RLOC mappings with the Map-Server (MS).
- Handles LISP control-plane functions, enabling location-independent routing across the fabric.

Policy Enforcement & Identity Assignment

- Performs user/device authentication via 802.1X (EAP) or MAB.
- Communicates with Cisco ISE to:
- Assign Security Group Tags (SGTs) for TrustSec policy
- Associate the endpoint with the correct host pool (VRF/SVI)
- Enforces SGT-based policies (e.g., ACLs, QoS) at ingress.

Fabric Border Node

A Fabric Border Node is the gateway between the SD-Access fabric and external L3 networks that are not part of the fabric (e.g., Internet, data centers, legacy networks, WAN, or cloud environments).

LISP PxTR Functionality

- PETR (Proxy Egress Tunnel Router):
  - For fabric-to-external traffic
  - Encapsulates traffic from inside the fabric → sends it out to the external network
- PITR (Proxy Ingress Tunnel Router):
  - For external-to-fabric traffic
  - Accepts non-LISP traffic from outside → performs LISP mapping lookup → encapsulates in VXLAN to send into the fabric

RLOC Advertisement

- Border nodes advertise external IP space (e.g., default route or 10.0.0.0/8) into the fabric control plane, so edge nodes can route traffic out of the fabric.

SGT Enforcement / Propagation

- If integrated with ISE and TrustSec, the border node can:
- Propagate SGTs to external networks that support TrustSec (e.g., ASA, Catalyst, ISE-enabled sites)
- Or, apply SGACLs at the border to enforce policy locally before forwarding to external destinations

Fabric Wireless LAN Controller (WLC)

The **Fabric WLC** is a wireless controller that integrates **wireless endpoints into the SD-Access fabric**, enabling **SGT-based policy**, **mobility**, and **VXLAN-based encapsulation**. However, it is **not an xTR or Anycast Gateway**.

- **Terminates CAPWAP** tunnels from fabric-mode APs
- **Authenticates clients** (via ISE), and applies **SGT**
- **Encapsulates wireless client traffic in VXLAN** for forwarding into the fabric
- **Sends client traffic to a Fabric Edge Node** (which acts as the xTR and gateway)
- Maintains **wireless mobility state** (roaming, session handoff, etc.)

Control Plane with Wireless Traffic (SD-Access)

In SD-Access, control plane traffic flows from the AP to the WLC over a standard CAPWAP tunnel, just like in traditional deployments. This includes management and client association data, such as MAC addresses, client-to-AP mappings, and session information.
In addition, the WLC participates in LISP control plane registration, sending EID-to-RLOC mappings to the Control Plane Node (MS/MR) on behalf of wireless clients.

Data Plane with Wireless Traffic (SD-Access)

Unlike in a traditional centralized WLC model, SD-Access offloads the data plane from the WLC.
Instead, the fabric AP establishes a VXLAN tunnel directly to its local Fabric Edge Node (which also acts as the client's Anycast Gateway). From there, the Edge Node encapsulates the traffic again and forwards it via VXLAN to the destination VTEP (the destination Fabric Edge Node).

As a result, two VXLAN segments are used:

- AP → Local Edge Node (VTEP)
- Local Edge Node → Destination VTEP

This model bypasses the WLC for data, reducing latency and optimizing the wireless data path through the fabric.

SD-Access Fabric Concepts

Virtual Network (VN)

The term “Virtual Network” is a bit misleading — in reality, it's just a **VRF instance** used to separate routing domains.

- Each VN maintains a **separate routing table** to isolate overlapping subnets.
- When routes are registered or queried via **LISP**, the **Instance ID** (LISP I-ID) is included to differentiate VN contexts.
- **Edge nodes attach a VXLAN VNID** to encapsulate traffic, which correlates to the LISP instance.

Use case: If two buildings both use 192.168.1.0/24 but belong to different VNs (e.g., Guest vs Secure), the **VNID and LISP Instance ID** ensure correct segmentation and route resolution.

Host Pool

A **Host Pool** is essentially a **range of IPs assigned to a specific VRF/SVI combo** — it defines where authenticated endpoints are logically placed.

- Think of it as **"who lives behind which gateway (SVI) in which VRF."**
- Each Fabric Edge Node (xTR) hosts one or more SVIs, acting as the **default gateway** for connected endpoints.
- These SVIs can be **Anycast** or **unique per node**, depending on the design.

TL;DR: **Host Pools = IP-to-Gateway (SVI) mapping context** within the SD-Access fabric.

Scalable Groups

Scalable Groups are **collections of endpoints** that share the **same policy behavior**, defined and identified by **SGTs (Security Group Tags)**.

- Assigned **dynamically by Cisco ISE** during authentication
- Used for **policy enforcement** (e.g., SGACLs, QoS, access control)
- Examples: SGT-VOIP, SGT-IPCAM, SGT-Guest

Devices don’t need to be in the same subnet or VRF — **SGTs unify policy across the fabric**, regardless of location.

Anycast Gateway

An **Anycast Gateway** is a **default gateway IP address that is identically configured on all fabric edge nodes** within a host pool.

- The **same IP/MAC is shared** across all nodes
- Hosts see a **consistent gateway IP** even when roaming
- Eliminates the need to change gateway config as devices move

TL;DR: **Anycast GW = mobility-friendly default gateway** for seamless host movement.

Example - Onboarding an IP Camera into Cisco SD-Access Fabric

A new IP camera is introduced into the enterprise network. The onboarding process is secure and dynamic, with minimal manual steps. Only the initial MAC address registration is performed manually for assurance.

Step-by-Step Workflow –

1. MAC Registration in ISE (Manual Step)
   - The camera's MAC address is preloaded into Cisco ISE as a known device.
   - ISE is configured to automatically assign a specific Security Group Tag (SGT) upon recognition.
2. Network Join and Authentication
   - The camera powers on and connects to a switch port.
   - The Fabric Edge Node triggers MAC Authentication Bypass (MAB).
   - ISE authenticates the MAC address and returns the assigned SGT (e.g., SGT: IPCAM).
3. SGT Policy Application
   The assigned SGT drives the following:
   - Placement in a dedicated VRF (e.g., VRF-IPCAM)
   - **Association with a VLAN/Subnet (SVI)** scoped to camera operations
   - **QoS class marking** for prioritized handling of real-time video streams (like VOIP/Video)
   - **Access limited** to specific VLANs (e.g., only Security Operations and Video Archive Server VLANs) using **SGACLs**
4. **LISP Control Plane Registration**
   - The **Fabric Edge Node (xTR / Anycast Gateway)** registers the camera's **EID (IP)** to its **RLOC (local loopback)** with the **Control Plane Node (MS/MR)**.
   - This enables **fabric-wide routing and identity-based reachability**.
5. **SGACL Enforcement at the Fabric Border**
   - At the **Fabric Border Node**, SGACLs are configured to **deny the IPCAM SGT** from reaching any **external networks or the Internet**.
   - This ensures the camera remains **contained within the fabric**.

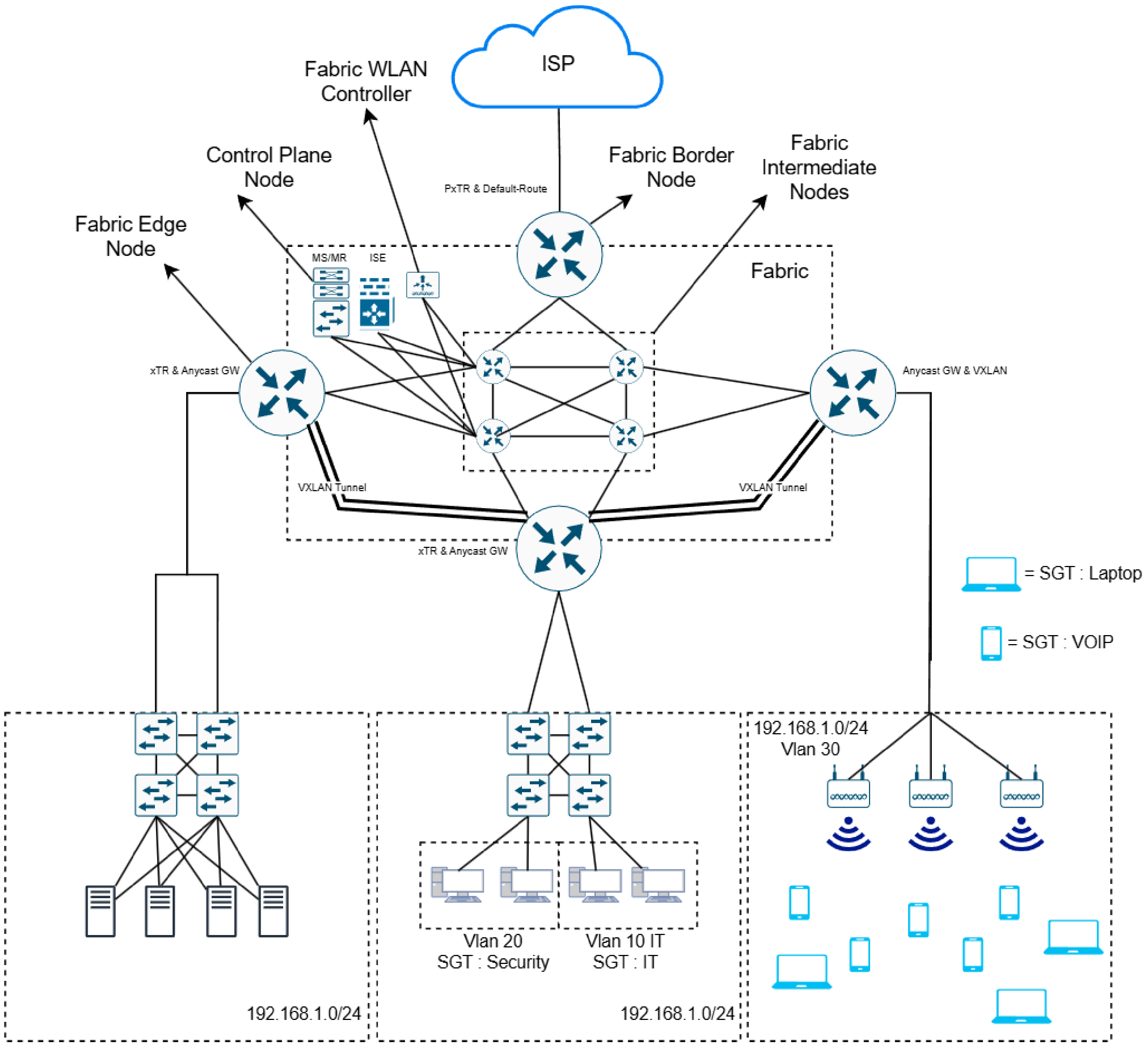
Result

The IP camera is now fully operational within the fabric:

- It is **automatically assigned the correct permissions** based on its identity.
- It receives **high-priority QoS treatment**.
- It is **allowed to communicate only with the exact endpoints required**.
- Its **network access and segmentation are consistent** regardless of physical location.

This process leverages:

- **SGTs for identity and policy**
- **VRFs for macro segmentation**
- **LISP for control-plane reachability**
- **VXLAN for data-plane encapsulation**
- **SGACLs at the border for security enforcement**



The example above shows an SD-Access use case:

- Device joins the network (Wired / Wireless)
- Authenticated via ISE
- Assigned SGT & SVI/VRF
- xTR registers with MS/MR (RLOC-to-EID)

Controller Layer

The **Controller Layer** is the intelligence plane of Cisco SD-Access, responsible for automation, analytics, and identity enforcement. It consists of three interdependent systems that collaborate to manage the network based on **intent**, **identity**, and **contextual telemetry**

NCP – Network Control Platform

Embedded in Cisco DNA Center (DNAC), Responsible for orchestration and provisioning of:

- Underlay (e.g., IS-IS)
- Overlay (LISP, VXLAN)
- Interfaces with devices using NETCONF, SNMP, and APIs.
- Pushes intent-based configurations automatically to fabric devices.

NDP – Network Data Platform

Acts as the telemetry and analytics engine, Collects real-time data from:

- NetFlow, Syslog, SNMP, SPAN or telemetry streams

Performs:

- Pattern analysis
- Anomaly detection
- Traffic forecasting
- Shares context with NCP and ISE for decision-making.

ISE – Identity Services Engine

Provides authentication and identity-based policy, Supports 802.1X, MAB, and WebAuth for endpoint auth.

Assigns:

- SGTs (Security Group Tags) for TrustSec enforcement
- Scalable Group membership
- Host Pool placement (VRF/SVI context)
- Consumes NDP context to enhance policy decisions.

Together, they enable an "**intent-based", self-adapting network** that configures and secures itself based on user identity and real-time network behavior.

Management Layer

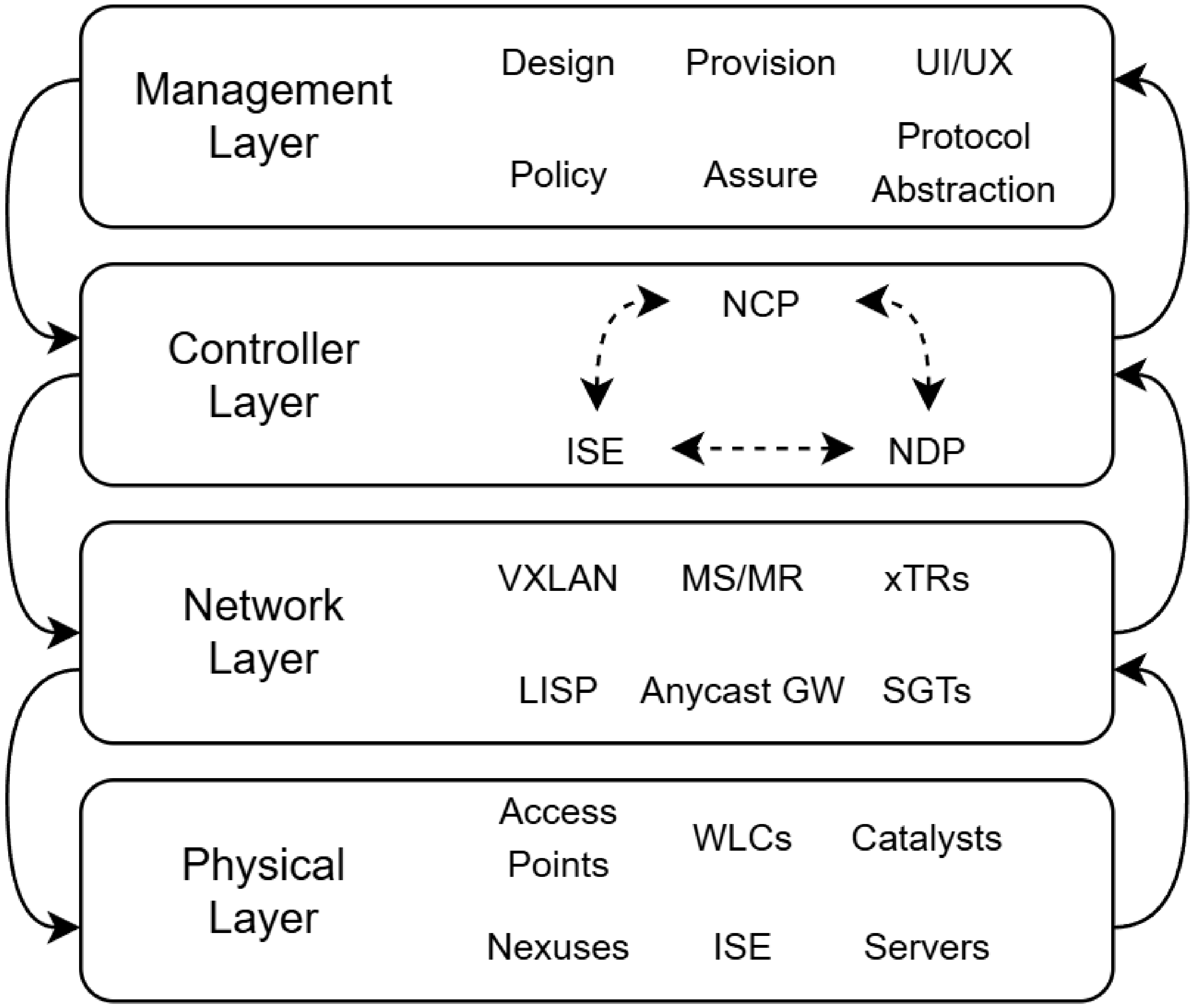
The **Management Layer** is the **user interface of Cisco DNA Center**, designed to abstract complex technologies like LISP, VXLAN, and TrustSec. It enables **intent-based network management** without requiring protocol-level expertise.

It operates through four core functions:

- **Design** – Define sites, IP pools, and network templates
- **Policy** – Create and manage segmentation and SGT-based policies
- **Provision** – Deploy configurations and assign fabric roles to devices
- **Assurance** – Monitor network health, performance, and telemetry data

This layer allows engineers to manage the SD-Access fabric using **intuitive workflows**, simplifying deployment, policy enforcement, and ongoing operations.

Cisco DNA Center Full-Stack Architecture



SD-WAN

SD-WAN is a software-defined networking solution for WAN connectivity that provides:

- **End-to-end encryption** over any transport (MPLS, LTE, broadband)
- **Centralized management** via GUI (vManage or Meraki Dashboard)
- **Application-aware routing** with dynamic path selection
- **Policy abstraction**, simplifying complex routing and SLA configs

There are two main versions:

- **Cisco SD-WAN (Viptela)** – Enterprise-grade, granular control, supports segmentation
- **Meraki SD-WAN** – Cloud-based, plug-and-play, simplified management with built-in security features

Just like SD-Access, SD-WAN uses controllers and overlays to automate and secure traffic across multiple sites.

Why SD-WAN?

Modern enterprises are increasingly offloading **SaaS and IaaS workloads to the cloud**, such as **AWS, Azure, and Microsoft 365**. However, traditional WAN architectures were designed for **data center-centric traffic**, and they struggle to support this shift in flow patterns.

As a result, enterprises must prioritize how to **optimize user experience** for **cloud-based applications**.

SD-WAN addresses this by providing tools to enhance **Quality of Experience (QoE)** for remote services. It enables the network to:

- **Select the best path per application**, based on real-time metrics like **latency, jitter, and packet loss**
- **Dynamically reroute traffic** if link conditions degrade

One key feature is **Application-Aware Routing (AAR)**, which uses **BFD (Bidirectional Forwarding Detection)** to continuously monitor link health. If performance drops below defined thresholds, SD-WAN can switch to a better-performing path automatically.

Combined with its **centralized control plane**, SD-WAN not only improves performance but also increases **resilience and high availability**, ensuring better end-user experience across distributed environments.

Cisco SD-WAN Roles

vBond (Orchestrator)

- Acts as the initial authentication and control-plane coordinator.
- Authenticates all SD-WAN components (Edge, vSmart, vManage) using certificates and RSA.
- Is reachable via FQDN or IP — FQDN is preferred for horizontal scaling and load balancing across multiple vBond instances.
- Performs NAT detection using NAT traversal (NAT-Stun) techniques to help establish tunnels behind NATed devices.
- Maintains permanent DTLS/TLS control sessions with vSmart.

Once authenticated, devices are handed off to vSmart and vManage for full control-plane and management-plane integration.

vManage (Management Plane)

The centralized GUI and API interface for operating the SD-WAN fabric, Manages:

- Device templates and onboarding
- Configuration of control and data plane policies
- Firmware/software updates
- Real-time monitoring and alarms

Interfaces with all components via secure API calls and facilitates policy distribution and visualization.

vSmart (Control Plane)

- Establishes DTLS/TLS sessions with all Edge devices.
- Runs OMP (Overlay Management Protocol) — Cisco's proprietary protocol similar to BGP — to:
  - Distribute routing information
  - Push data policies (e.g., traffic steering, QoS, segmentation, security zones)

Acts as the central policy decision point, ensuring consistent forwarding behavior across the SD-WAN overlay.

SD-WAN Edge Routers (Data Plane)

- Physical or virtual routers deployed at branch sites, DCs, cloud, or SOHOs.
- Establish DTLS/TLS connections to vSmart, forming OMP neighbor relationships.

Receive:

- Route advertisements
- Policies (QoS, security, segmentation) from vSmart

Support multiple transports (MPLS, LTE, DIA) and form the encrypted overlay fabric.

vAnalytics (Optional Insight Layer)

Integrated with vManage to provide:

- Traffic visibility
- Historical trends
- Link performance stats (latency, jitter, packet loss)
- Forecasting and health alerts
- Helps identify policy inefficiencies, link saturation, or outages.

Offers optimization suggestions based on observed performance.

Network Assurance

Network Assurance refers to the continuous monitoring, verification, and validation of network health, performance, and behavior to ensure that the network operates as intended. It encompasses:

- **Fault Detection & Diagnosis**: Identifying issues via logs (syslog), SNMP traps, NetFlow, or telemetry.
- **Performance Monitoring**: Measuring delay, jitter, loss, and availability (e.g., via IP SLA).
- **Operational Verification**: Using tools like debug, traceroute, and conditional monitoring to validate control/data plane operation.
- **Automated Remediation**: Integrating fallback mechanisms like floating static routes or event-based scripts (EEM) to maintain service continuity.
- **Telemetry & Analytics**: Tools like **Cisco DNA Center Assurance** use real-time telemetry, AI/ML, and intent-based validation to proactively spot and resolve issues.

Ping

ping is an ICMP-based tool used to verify Layer 3 (IP) reachability and basic Layer 2 functionality. It's one of the simplest and most universally available network diagnostic commands.

Use cases

Connectivity Testing:
Confirms that the destination IP is reachable. Quickly detects issues like:

- Down interfaces
- Unplugged/faulty cables
- Misconfigured IP addresses or gateways

MTU/MSS Troubleshooting:
By using extended ping options (size + DF-bit), you can test path MTU to discover:

- Black hole scenarios in tunnels (e.g., GRE, IPsec)
- Application-specific failures due to fragmentation

Response Timing:
Measures round-trip time (RTT) for latency analysis, though basic.

Limitations

- Doesn't show the **entire path** (unlike traceroute)
- Can be **rate-limited or blocked** by ACLs, firewalls, or control plane policing
- Lacks granularity: Cannot isolate L2 vs L3 issues explicitly

Role in Network Assurance

- Forms the basis of **reactive troubleshooting**
- Often used within automated monitoring systems to verify **device reachability**
- Used in **IP SLA operations** for proactive path and performance testing

Best practice: Always use ping as your first step before escalating to more advanced tools like debug, traceroute, or protocol-level analysis.

Traceroute

traceroute is a diagnostic tool used to map the path packets take from a source to a destination. It helps identify where delays or failures occur along the route.

How It Works

- Sends **probe packets** (typically UDP, ICMP, or TCP) with **incrementing TTL (Time to Live)** values.
- Each router that decrements the TTL to 0 responds with an **ICMP Time Exceeded** message.
- By collecting each hop's reply, the full path is mapped.
- Usually sends **3 probes per TTL** to measure RTT variation.

Use Cases

- Path discovery (routing verification)
- Identifying slow or failing hops
- Locating routing loops
- Diagnosing black holes in GRE/IPsec/MPLS

Common Failure Causes

- Router drop/no reply
- ICMP Replies disabled
- Down interface/hop
- Loop in routing
- Firewall / ACL drops

Several common Traceroute Codes / Symbols

- * - (No response (Timeout for specific probe)
- !H - Host unreachable
- !N – Network Unreachable
- !F – Fragmentation needed and DF-bit set
- !X – Communication administratively prevented

These codes may vary by platform (Cisco vs Linux vs Windows). Some implementations only show * or basic ICMP errors.

Best Practices: Use with extended options: traceroute <ip> numeric, -T for TCP-based traceroute, Combine with ping and show ip route to get full path + protocol reachability

By default, Traceroute sends 3 probes using UDP with high source and destination ports, this should be taken into account when using it in a production environment that might prohibit UDP probes or when hosts aren't confgured to respond with ICMP Type 3 Code 3 (Port unreachable), This can also be impacted by Assymetric routing (or ECMP for that matter)

Cisco Debugging

The debug command in Cisco devices provides real-time visibility into process-switched packets, making it an essential tool for troubleshooting network protocols like OSPF. However, debugging can be CPU-intensive, potentially overwhelming the CLI with excessive output and impacting network performance.

Mitigating Debugging Risks

To safely use debugging, consider these best practices:

1 - Use Access Control Lists (ACLs) – Filter debug output to specific traffic.

2 - Apply Conditional Debugging – Limit debugging to specific conditions, such as a source IP or interface.

3 - Redirect Debug Logs to Buffer – Store debug output in memory instead of the console to avoid system performance degradation.

4 - Save Debug Logs to File – Preserve logs for further analysis.

Example: Debugging OSPF Messages for Specific Hosts

To filter OSPF (port 89) debug logs between two hosts, use an extended ACL:

```
access-list 101 permit ospf host 1.1.1.5 host 1.1.1.6

debug ip packet detail 101
```

This ensures only OSPF packets between 1.1.1.5 and 1.1.1.6 are logged, reducing unnecessary output.

Example: Conditional Debugging for an Interface

To debug traffic only on interface GigabitEthernet0/1:

```
debug condition interface GigabitEthernet0/1

debug ip ospf events
```

This command ensures OSPF debug output is restricted to the specified interface.

Redirecting Debug Logs to a Buffer (Safer Approach)

Instead of flooding the console with debug messages, store them in the logging buffer:

```
logging buffered 50000 debugging

show logging
```

This allows logs to be reviewed later without impacting system performance.

Saving Debug Logs to Bootflash for Analysis

To preserve debug output for future analysis, redirect logs to a file:

```
monitor capture mydebug buffer limit 10 size 100 type buffer

monitor capture mydebug export bootflash:ospf_debug.pcap
```

This stores the captured debug logs in bootflash:/ospf_debug.pcap, making it accessible for further analysis.

Stopping Debugging

After troubleshooting, always disable debugging to avoid excessive CPU usage:

```
undebug all
```

By using ACL filtering, conditional debugging, buffered logging, and saving logs to bootflash, you can troubleshoot effectively while minimizing impact on the network. These practices help ensure debugging remains controlled and insightful without overwhelming system resources.

SNMP

SNMP is a protocol used to monitor, manage, and alert on network devices. It allows a central **SNMP Manager** to query or receive notifications from **SNMP Agents** (routers, switches, etc.) running on each device.

Primary Use Cases

- **Event Alerting**: Interface status changes, OSPF neighbor events, authentication failures, etc.
- **Polling**: The manager can periodically pull counters (e.g., CPU, memory, bandwidth).

Although SNMP supports write actions (via SET), it's almost never used for configuration in modern environments. Tools like **Ansible, RESTCONF**, or **NETCONF** are preferred

MIB (Management Information Base)

- A structured database of objects that can be **monitored or set** via SNMP.
- MIBs define **OID trees** (Object Identifiers) for various device stats (e.g., interface stats, CPU load, BGP peer status).
-

SNMPv3 is the only version recommended for secure environments, SNMP is **agent-based**, meaning each device must have SNMP configured with correct credentials and ACLs. It is **not agentless**, unlike tools like **Ansible**, which use SSH or APIs.

TL;DR: The MIB is a catalog inside the SNMP agent (device) that defines what can be monitored. The SNMP server (manager) queries the agent using that structure to retrieve readable, structured data mapped to OIDs.

There are Standard MIBs (IETF) that are defined by RFCs and supported by all compliant devices, but MIBs between different vendors vary, it depends on the application, the vendor and features.

Basic SNMP Config

```
! Restrict SNMP access to a specific NMS host

access-list 1 permit 192.168.1.1 0.0.0.0


! Define community strings with access level and bind them to the ACL

snmp-server community SNMP-Server ro 1

snmp-server community SNMP-Server-Write rw 1


! Enable only relevant traps to avoid flooding the NMS

snmp-server enable traps config

snmp-server enable traps cpu


! Define SNMP trap destination

snmp-server host 192.168.1.1 traps SNMP-Server
```

Syslog

Syslog is a protocol used for **logging system messages** on Cisco devices. It is **event-driven**, meaning logs are **generated and pushed** when specific events occur (e.g., **link flap**, **neighbor down**, **authentication failure**, etc.). It does **not support polling** like SNMP.

Syslog messages can be directed to the **console**, **buffer**, or a **remote syslog server**. These destinations help operators **monitor**, **troubleshoot**, and **audit** network behavior.

Syslog vs. SNMP

- **Syslog** = **Push model**, event-driven logs
- **SNMP** = **Pull model**, used for polling metrics (with optional trap support)
- Together, they provide **complementary monitoring and observability**

LEVEL	SEVERITY	DESCRIPTION
0	**Emergency**	System is unusable or about to crash
1	**Alert**	Immediate action needed (e.g., power failure)
2	**Critical**	Critical conditions (e.g., link down)
3	**Error**	Runtime errors (e.g., interface errors)
4	**Warning**	Warning conditions, not yet critical
5	**Notice**	Significant but normal conditions
6	**Informational**	Informational messages (e.g., OSPF up)
7	**Debug**	Debugging messages (used in troubleshooting)

Syslog Storage Locations

Console –

- Default location for syslog messages
- Message show up when connected via console or vty (if enabled)
- Default logging level: severity 7 (debug)
- Used during interactive troubleshooting with the debug command

Syslog Buffer –

- Dedicated internal buffer for storing logs
- Configurable size and severity level

Syslog Server –

- Sends logs to an external host over UDP 514
- Log persists across device reloads
- Ideal for compliance, aduiting and long term storage.

Syslog Configurations

```
!Changing Logging buffer size (Bytes)
Logging buffer {Size}

!Changing severity
Logging buffer {1-7}
!Example for sotring level 4 and below in the buffer
Logging buffered warnings
!Example of storing level 5 and lower with Buffer of 4096
Logging buffered 4096 notifications

!Show logging buffer
Show logging

!Example of moving debug messages to Syslog buffer and removing them from Console
Logging buffer 100000
Logging buffer debugging
No logging console

!Send logs to host only for severties 2 and lower
Logging host 192.168.1.1
Logging trap alerts
```

Netflow and Flexible Netflow

Netflow

NetFlow is a Cisco-developed protocol for collecting **IP traffic statistics** on network interfaces. It records details like source/destination IPs, ports, protocols, and byte/packet counts. Routers maintain a **flow cache** of these entries and periodically **export** them to a NetFlow **collector** for analysis.

NetFlow helps network engineers:

- Understand traffic patterns
- Detect anomalies or congestion
- Assist in capacity planning and policy tuning

It complements SNMP (for device metrics) and Syslog (for event tracking) by providing **flow-level visibility** into **who is talking to whom and how much**.

Flexible Netflow

Flexible NetFlow is an enhanced version of traditional NetFlow, designed to meet the needs of modern, dense, and diverse networks. It provides:

- **Custom flow definitions** based on fields like ports, protocols, and IPs.
- **Traffic sampling** to reduce overhead on high-speed interfaces.
- **Greater control** over what data is collected and exported.

While it offers significant flexibility and scalability, features like sampling can potentially obscure low-volume or short-lived flows.

A practical use case is monitoring and exporting traffic **from a specific host or destination**, enabling targeted flow analysis.

Netflow Configuration

```
! Enable NetFlow on an interface (usually ingress)
interface GigabitEthernet0/1
 ip flow ingress

Define the destination (NetFlow collector)
ip flow-export destination 192.168.1.100 2055

! Set version (v5 or v9)
ip flow-export version 9

! Optionally set source interface for exports
ip flow-export source Loopback0
```

Flexible Netflow Configuration

```
!Create a flow record (define what to match and collect)
flow record FNF-RECORD
 match ipv4 source address
 match ipv4 destination address
 match transport source-port
 match transport destination-port
 collect counter bytes
 collect counter packets

!Create a flow exporter (where to send flow data)
flow exporter FNF-EXPORTER
 destination 192.168.1.100
 source Loopback0
 transport udp 2055
 export-protocol netflow-v9

!Create a flow monitor (bind record + exporter)
flow monitor FNF-MONITOR
 exporter FNF-EXPORTER
 record FNF-RECORD
 cache timeout active 60

! Apply the monitor to the interface
interface GigabitEthernet0/1
 ip flow monitor FNF-MONITOR input
```

This Flexible NetFlow configuration captures ingress traffic on interface GigabitEthernet0/1, matching flows based on IPv4 source and destination addresses as well as source and destination ports. It collects packet and byte counters for each flow and exports the flow data from the router’s Loopback0 interface to a NetFlow collector at 192.168.1.100 using UDP port 2055.

SPAN

SPAN is a monitoring feature that allows a switch to copy traffic from one or more interfaces or VLANs and forward that traffic to a destination port for analysis. The specific SPAN method (Local SPAN, RSPAN, or ERSPAN) is selected based on the analyzer's location in the network. The analyzer receives the complete original packet, enabling full inspection.

SPAN traffic can oversubscribe the destination port if the combined mirrored bandwidth exceeds its capacity. For example, mirroring two 10G ports to a single 1G destination port will result in **packet loss** at the SPAN destination, reducing visibility and analytical accuracy.

Local SPAN

Local SPAN (LSPAN) is used when both the source interface(s) and the analyzer (destination) are located on the same switch. It allows direct copying of traffic from one or more interfaces to a local destination port without involving any VLAN tagging or encapsulation. Configuration simply requires identifying the source and destination interfaces.

```
monitor session 1 source interface Gi1/0/1 - 2

monitor session 1 destination interface Gi1/0/52
```

Remote SPAN

Remote SPAN (RSPAN) is used when the source and destination interfaces are on **different switches within the same Layer 2 domain**. The source switch mirrors traffic into a dedicated **RSPAN VLAN**, which is tagged and forwarded across trunk links to the destination switch. The destination switch is configured to monitor this VLAN and forward the mirrored traffic to a local analyzer port.

RSPAN does **not involve IP addresses** or routing. The **RSPAN VLAN must be configured on all intermediate switches** with the remote-span keyword, and it must be **allowed on all trunk links** involved in the path. Additionally, on the **destination switch**, a separate monitor session must be configured to receive mirrored traffic from the RSPAN VLAN and forward it to the analyzer interface.

Care must be taken in design, as overuse can lead to congestion or unnecessary traffic propagation across the Layer 2 domain.

```
!Create RSPAN Vlan

vlan 99

 name RSPAN-Vlan

 remote-span


!on source switch, Identify the Analyzer.

monitor session 1 source interface Gi1/0/1

monitor session 1 destination remote vlan 99


!on destination switch, identify the target of replication.

monitor session 1 source remote vlan 99

monitor session 1 destination interface Gi1/0/52
```

Encapsulated Remote SPAN

Encapsulated Remote SPAN (ERSPAN) is used when the analyzer is located outside the local Layer 2 domain. It enables full frame replication by encapsulating mirrored packets inside an IP packet using a **GRE header format**. This allows the traffic to traverse Layer 3 networks without needing traditional GRE tunnel interfaces. The GRE terminology refers to the **packet structure**, not a manually configured GRE tunnel. IP connectivity must exist between the source and destination. Configuration includes specifying the source interface, ERSPAN destination IP, ERSPAN ID, origin IP, and optional TTL.

```
!Create ERSPAN session on Source switch
monitor session 1 type erspan-source
 source interface Gi1/0/1
 destination
  ip address 192.168.1.250
  erspan-id 1
 origin ip address 192.168.2.13
 erspan ttl 32
```

IP SLA

A built-in Cisco IOS feature that allows the router to proactively **simulate traffic** and measure **performance metrics** across the network like a built-in automation probe engine.

SLA OPERATION	USE CASE	PROTOCOL
ICMP-ECHO	Track reachability/latency	ICMP
UDP-JITTER	Measure jitter and delay	UDP (VoIP)
HTTP	Verify HTTP service availability	HTTP
TCP-CONNECT	Check TCP port reachability	TCP
DNS	Measure DNS resolution time	UDP

IP SLA Configuration

```
ip sla 1
 icmp-echo 8.8.8.8 source-interface Loopback0
 frequency 60
ip sla schedule 1 start-time now life forever
```

Other Uses:

- **Track objects** for FHRP decisions (e.g. HSRP failover)
- **SNMP integration** for monitoring metrics (latency, loss, jitter)
- **Time-based testing** for links or services (e.g., backup line checks at night)

Secure Network Access Control

Cisco SAFE is a **security reference architecture and framework** designed to help organizations **visualize, design, and implement** consistent security across the entire enterprise network.

PIN EXAMPLE	TYPICAL RISKS	COMMON PROTECTIONS
CAMPUS	BYOD, lateral movement	NAC (ISE), segmentation, posture checks
BRANCH	Untrusted edge access	SD-WAN firewalls, secure tunneling
DATA CENTER	East-west threats, data exfil	NGFW, segmentation, IPS, DDoS protection
EDGE (INTERNET)	DDoS, scanning, external actors	Firewall, DNS-layer security (Umbrella)
CLOUD (SAAS/IAAS)	API abuse, misconfig	Cloud-native controls, CASB, AMP

Each PIN is protected with **appropriate controls based on risk, visibility, and cost**.

Cisco SAFE organizes its protections around the **security lifecycle –**

PHASE	FOCUS	EXAMPLE TOOLS/TECHNOLOGIES
BEFORE	Prevent known threats	NGFW, ISE, segmentation, access control
DURING	Detect/respond in real time	AMP, Firepower IPS, NetFlow/SNMP analytics
AFTER	Analyze and recover	Stealthwatch, DNA Center, forensic logs

Cisco SAFE doesn't specify just one product, it includes multiple **security layers and tools**:

- Identity and Access Control (ISE, 802.1X, MAB)
- Firewalls (NGFW, ZBFW, Cloud Firewalls)
- Intrusion Detection and Prevention (IDS/IPS)
- Anti-malware and sandboxing (AMP, Threat Grid)
- Network telemetry and analytics (NetFlow, Stealthwatch)
- Cloud security (Umbrella, Duo, CASB)

Endpoint Security

Endpoints — such as laptops, mobile devices, and IoT nodes — are among the **most vulnerable assets** in the enterprise. Modern trends like **BYOD**, **remote work**, and **wireless expansion** have increased exposure at the **Branch** and **Campus** level.

Traditional antivirus is not enough:

- Static signature-based AV can't handle **zero-day exploits**, **fileless malware**, or **polymorphic attacks**.
- Attackers now operate with higher **speed and sophistication** than traditional defenses.

Cisco Talos: Global Threat Intelligence Engine

Cisco Talos is the threat intelligence group at Cisco that:

- Aggregates telemetry from Cisco products (Firepower, AMP, Umbrella, etc.)
- Collaborates with security research communities and law enforcement
- Publishes real-time threat data, malware signatures, IOC feeds, and vulnerability research

It’s easier to see it as a real-time global threat database that identifies, classifies, and shares malicious activity across Cisco’s entire security ecosystem.

Cisco Secure Malware Analytics (Threat Grid)

Cisco Secure Malware Analytics—commonly known as Threat Grid—is a cloud-based or on-premises malware analysis platform that performs both static and dynamic analysis of files to detect malicious behavior.

Key Features:

- Static analysis: Examines file metadata like filenames, hashes (e.g., MD5), and structure without executing the file.
- Dynamic (behavioral) analysis: Executes files in a controlled sandbox environment to monitor and analyze their behavior in real-time.
- Threat Intelligence Integration: Leverages Cisco Talos intelligence and correlates findings with billions of known malware artifacts.
- Anti-evasion: Many malware types attempt to detect sandbox environments to evade execution; Threat Grid is designed to remain undetectable to such malware.
- Manual and Automatic Submission: Suspicious files can be uploaded manually or automatically from integrated Cisco or third-party security tools.
- Glovebox: A secure interactive sandbox interface that allows direct observation and interaction with potentially malicious files.

This tool plays a critical role in advanced malware detection and response workflows by identifying not just whether a file is malicious, but also what it does, when it does it, and how it behaves in various execution stages.

Cisco Advanced Malware Protection (AMP)

Cisco AMP is a centralized, cloud-based advanced threat detection and prevention solution that protects against malware before, during, and after an attack. It integrates multiple technologies and intelligence sources to provide comprehensive threat visibility and control across various vectors.

Key Components:

- Cisco Talos: Provides real-time, global threat intelligence (file hashes, IPs, domains, TTPs).
- Cisco Threat Grid: Performs dynamic behavioral analysis of files in a sandbox to understand what a file does, not just what it is.
- Cisco AMP Cloud Database: Maintains retrospective security data, allowing analysis and alerting even *after* malware has initially bypassed defenses.

Deployment Points:

AMP can be integrated across multiple attack vectors:

- AMP for Endpoints (Windows, macOS, Linux)
- AMP for Networks
- AMP for Email Security Appliances (ESA)
- AMP for Web Security Appliances (WSA)
- AMP for Firepower (NGFW, NGIPS)

Capabilities:

- Prevention: Blocks known threats using signatures from Talos and cloud indicators.
- Detection: Identifies malicious behavior using Threat Grid analytics and file trajectory.
- Retrospective Security: If a previously unknown file is later classified as malicious, AMP can retroactively detect and contain it based on historical telemetry.

Cisco Secure Anyconnect

Cisco AnyConnect Secure Mobility Client is a remote access VPN solution that provides secure encrypted connectivity from endpoints (laptops, mobile devices) to the enterprise network.

FUNCTION	DESCRIPTION
VPN ENCRYPTION	Supports both **IPsec (IKEv2)** and **TLS (SSL VPN)** for encrypted tunnels
APPLICATION LAYER TLS	If TLS VPN is used, traffic is encrypted using HTTPS-like TLS encapsulation
NETWORK LAYER IPSEC	Alternatively, **IPsec-based IKEv2 VPN** provides full network-layer encryption
MODULAR DESIGN	Includes optional **modules** like posture checks and telemetry
CISCO ISE INTEGRATION	Works with **Cisco Identity Services Engine** to enforce **posture checks**, ensuring the endpoint meets security policies (e.g., up-to-date AV, no malware) before permitting access

Cisco Umbrella

Cisco Umbrella is a cloud-delivered security platform that primarily acts as a DNS-layer protection system to block access to malicious, phishing, and command-and-control domains before a connection is ever established.

FUNCTION	DESCRIPTION
DNS-LAYER SECURITY	Intercepts and filters DNS queries to block access to known malicious domains
PHISHING & MALWARE PROTECTION	Prevents users from resolving or accessing domains associated with phishing, malware, ransomware, and botnets
ANYCAST INFRASTRUCTURE	Uses globally distributed data centers with Anycast routing to ensure low-latency, high-availability resolution
VISIBILITY & LOGGING	Logs DNS requests per user/device/IP, allowing visibility into internet-bound traffic
POLICY ENFORCEMENT	Allows customized block/allow lists, content filtering, and identity-based rules (per IP, user, group)

Deployment Simplicity:

Can be deployed by **changing DNS server settings** on:

- DHCP scope (e.g., 208.67.222.222, 208.67.220.220)
- Client OS settings
- Router/gateway configuration

Optional **Umbrella roaming client** for identity-based enforcement outside the network (e.g., for laptops on public Wi-Fi).

Cisco Secure Web Appliance (WSA)

Cisco Secure Web Appliance (WSA) is a dedicated web security gateway that protects users from web-based threats. It inspects and filters HTTP/HTTPS traffic, blocks malicious sites, enforces URL filtering, prevents data leaks, and analyzes downloads using Cisco AMP and Threat Grid. It leverages Talos intelligence and can detect threats both in real-time and retrospectively, quarantining malicious content even after initial access.

Cisco Secure Email (ESA)

Cisco Secure Email (formerly ESA/IronPort) is an **email security gateway** designed to **prevent phishing, malware, spam, and malicious URLs** in inbound and outbound email traffic.

It leverages **Cisco Talos** for real-time threat intelligence and **Cisco Secure Malware Analytics (Threat Grid)** for behavioral file analysis. ESA ensures **malicious or suspicious emails are blocked before reaching users**.

It also offers:

- **URL filtering and domain reputation checks**
- **Spam and spoofed domain protection**
- **Advanced analytics** to track which users received threats, clicked links, or are at highest risk

This makes ESA a **critical first layer** of defense for **email-based attacks**, one of the most exploited vectors in enterprise environments.

Cisco Secure IPS (Firepower NGIPS)

Cisco Secure IPS (formerly FirePOWER, from the 2013 Sourcefire acquisition) is an inline threat prevention system that can detect, alert, log, and block threats in real time.

Unlike an IDS, which is passive, IPS is active — it stops malicious traffic.

It integrates:

- Cisco Secure Malware Analytics (Threat Grid) for deep file behavior inspection
- Cisco Talos (indirectly) for signature and threat intelligence
- Cisco AMP for file reputation and retrospective detection

It provides real-time, signature- and behavior-based threat prevention at the network level and is commonly deployed at critical ingress/egress points for proactive intrusion defense.

Cisco Secure Firewall (NGFW)

Cisco Secure Firewall (Next-Generation Firewall) inspects and controls inbound and outbound traffic using both traditional firewall functions and advanced, application-aware capabilities.

It supports:

- Stateful packet inspection
- Layer 7 (application-layer) inspection
- Granular access control policies
- Deep packet inspection (DPI) for detecting evasive threats

Cisco NGFW integrates with:

- Talos for threat intelligence
- Cisco Secure Malware Analytics (Threat Grid) for file sandboxing
- Cisco AMP Cloud for real-time malware detection and retrospective analysis

It is available in multiple form factors:

- Physical appliances (e.g., Firepower 1000/2100/4100 series)
- Virtual firewalls (e.g., Secure Firewall Threat Defense Virtual)
- Cloud-delivered firewall capabilities
- Legacy ASA platforms with Firepower services

This makes Cisco NGFW a core perimeter defense tool, combining classic firewall control with modern threat prevention.

Cisco Secure Network Analytics (Stealthwatch Enterprise)

Cisco Stealthwatch Enterprise is a centralized network visibility and threat detection platform that aggregates and analyzes telemetry data from across the network to detect anomalies, lateral movement, and insider threats.

Key Components:

- Flow Collectors: Capture NetFlow, IPFIX, sFlow, syslog, SNMP, and telemetry from routers, switches, firewalls, and integrations like ISE or AMP.
- Cisco Flow Analytics Manager (FMC): Correlates, organizes, and analyzes data across collectors, creating a unified threat and traffic dashboard.
- Flow Rate Licensing: Required to operate based on traffic volume (measured in flows per second).

Optional components include:

- Stealthwatch Cloud (for cloud telemetry)
- Integration with SecureX, Threat Grid, and other Cisco security platforms

Stealthwatch provides rich behavioral analytics using machine learning and can detect threats even without signature-based detection, making it critical for east-west traffic visibility and zero-trust enforcement.

Stealthwatch Cloud (now branded as Cisco Secure Cloud Analytics) is a cloud-native threat detection and visibility platform that monitors network activity in public cloud environments (like AWS, Azure, GCP) and remote endpoints.

It collects and analyzes cloud flow logs and telemetry using machine learning to detect:

- Misconfigurations
- Lateral movement
- Data exfiltration
- Anomalous or unauthorized behavior

As a SaaS solution, it requires no on-prem infrastructure and provides centralized visibility across multi-cloud and hybrid environments.

Cisco ISE (Identity Services Engine)

Cisco ISE is a centralized policy engine that controls who and what gets access to the network, based on identity, device posture, and context — not just IP or MAC addresses.

ISE integrates with authentication protocols (e.g., 802.1X, RADIUS, EAP) and works with Active Directory to authenticate users and devices. It interfaces with Cisco security products like AMP, Umbrella, and Secure Firewall, and with DNA Center for policy-based automation.

Key Capabilities:

- Identity-Based Access Control: Assigns access based on user, device type, or role, not just IP/MAC
- Cisco TrustSec & SGTs: Uses Security Group Tags (SGTs) for logical, scalable segmentation and access enforcement
- Posture Assessment: Verifies device compliance (AV status, patches, etc.) before granting access (via AnyConnect)
- Guest/BYOD Portals: Manages secure access for unmanaged or guest devices
- Centralized Policy Engine: Granular control of network access across wired, wireless, VPN

ISE shifts the model from IP-based enforcement to identity-driven access, improving security, visibility, and policy flexibility across the enterprise.

pxGrid

pxGrid is a secure, scalable framework built into Cisco ISE that allows different security systems to exchange contextual data (like user identity, device posture, threat status). It uses protocols like XMPP over TLS and a publish/subscribe model.

pxGrid enables Cisco and third-party tools (e.g., AMP, Stealthwatch, firewalls, SIEMs) to share and act on real-time security information, allowing for automated threat response such as quarantining endpoints or adjusting access policies dynamically.

Example –

A user’s laptop connects to the corporate network through VPN using Cisco AnyConnect. The laptop appears clean during initial ISE posture check, so access is granted.

Later, Cisco AMP for Endpoints detects suspicious behavior on the laptop (e.g., a known malware hash execution).
AMP publishes a threat event to pxGrid.

Cisco ISE, subscribed to AMP's threat intelligence feed via pxGrid, receives the alert and automatically updates the user’s Security Group Tag (SGT) to “High Risk.”

As a result, the Cisco NGFW enforces a policy that limits the infected host to a quarantine VLAN or blocks all outbound internet access — all without manual intervention.

Product	Category	Function	Deployment Location
Talos	Global Threat Intelligence	Feeds signatures and threat data to other systems	Cloud-based
Threat Grid	Malware Behavioral Analysis	Sandbox for static and dynamic file inspection	Cloud / On-Prem
AMP for Endpoints	Endpoint Detection & Response	Real-time malware detection, file tracking, retrospective alerts	Deployed on endpoints
AnyConnect	VPN with Posture Check	Secure access with pre/post connects posture enforcement	Client device / VPN Gateway
Umbrella	DNS-layer Security	Blocks malicious domains before IP connection is made	DNS Forwarder / Internet Edge
Secure Email (ESA)	Email Security Gateway	Blocks phishing, malware, spam, malicious URLs in emails	Mail Gateway / Perimeter
Secure Web (WSA)	Web Proxy & Threat Filtering	URL filtering, malware scanning, SSL decryption	Proxy Gateway / Perimeter
NGFW (Firepower)	Next-Gen Firewall	Stateful inspection, app control, integrations AMP/Talos	Network Edge / Internal segmentation
NGIPS (Firepower)	Intrusion Prevention System	Inline threat detection and prevention	Network Inline / Behind NGFW
ISE	Identity & Access Control	Controls network access via 802.1X, posture, SGTs	Core Policy Engine
Stealthwatch Enterprise	Network Flow Analytics	Analyzes NetFlow/sFlow for anomaly detection	On-Prem / Core Network
Stealthwatch Cloud	Cloud-native Flow Monitoring	Monitors public cloud traffic and remote endpoints	SaaS / Cloud Networks
pxGrid	Context-Sharing Framework	Exchanges threat, posture, identity data across systems	Embedded in ISE / Network-wide

Network Access Control (NAC)

NAC is a concept and set of technologies used to control who and what can access the network, Cisco's ISE for example is the platform that utilizes these technologies for authentication –

- 802.1X – Authentication at port-level via EAP methods
- MAB (MAC Auth Bypass) – Authenticates non .1x users via MAC addresses
- WebAuth – Portal login (Guest access)
- Cisco TrustSec

802.1X

802.1X is a port-based Network Access Control mechanism defined by IEEE, designed to authenticate devices before they are allowed to send normal traffic into the LAN. It relies on the EAP (Extensible Authentication Protocol) framework to relay authentication credentials.

802.1X Roles –

- Supplicant - The end host (e.g., PC, phone, IP camera) requesting access. It runs an 802.1X client to handle EAP exchanges.
- Authenticator - Typically a switch or WLC, which enforces port control and acts as a proxy to forward EAP messages.
- Authentication Server - Usually a RADIUS server (e.g., Cisco ISE) that validates credentials and returns an authorization decision.

Communication between Authentication server & Host over the Authenticator is done via EAPoL (EAP over LAN) encapsulated within RADIUS messages over UDP.

Authentication Flow –

1. Link Detection: A device connects to a switch port. Either:
    - The supplicant sends EAPoL-Start
    - The switch proactively sends EAP-Request/Identity.
2. EAP Exchange:
    - The switch forwards EAP messages between the supplicant and the RADIUS server.
    - The switch does not inspect credentials — it simply acts as a relay.
    - The RADIUS server processes the credentials and applies the defined authentication method (e.g., EAP-TLS, PEAP, EAP-FAST).
3. Authorization Result:
    - If accepted: The switch opens the port.
    - Optionally, the server returns:
        - Downloadable ACLs (dACLs) to restrict traffic.
        - SGT (Security Group Tags) used in Cisco TrustSec.
        - VLAN assignment to place the device in a specific access zone.
4. Port State:
    - If successful: Port becomes authorized, and user data can flow.
    - If failed: Port remains unauthorized (can fall back to MAB or WebAuth if configured).

EAP Authentication Methods

- EAP Challenge-based Authentication Methods –
    - Extensible Authentication Protocol- Message digest 5 (EAP-MD5)
- EAP TLS Authentication methods –
    - Extensible Authentication Protocol – Transport layer Security (EAP-TLS)
- EAP Tunneled TLS Authentation Methods –
    - Extensible Authentication Protocol – Flexible Authentication via Secure Tunneling (EAP-FAST)
    - Extensible Authentication Protocol – Tunneled Transport Layer Security (EAP-TTLS)
    - Protected Extensible Authentication Protocol – PEAP
- EAP Inner Authentication Methods –
    - EAP Generic Token Card (EAP-GTC)
    - EAP-MSCHAPv2 (Microsoft Handshake)
    - EAP TLS

EAP-FAST with Chaining

- Purpose: Authenticates both machine and user in a single EAP session.
- Mechanism: Uses a single TLS tunnel (outer EAP) established via PACs (not certificates).
- No separate inner EAP methods — machine and user credentials are both exchanged within the same tunnel.
- Advantage: Avoids redundant TLS tunnel setups seen in PEAP (which uses separate tunnels), resulting in faster authentication, especially during wireless roaming.
- Use Case: Cisco environments needing seamless, secure roaming with unified policy control.

MAB

MAB is a fallback mechanism to 802.1X, used when a device cannot support 802.1X (e.g., IP phones, printers, cameras). Instead of using EAP-based credentials, the switch uses the device's MAC address as its identity.

The switch (authenticator) tries to initiate 802.1X authentication, If no response is received within a timeout (typically ~90 seconds), the switch assumes the device does not support 802.1X. The switch opens the port briefly, captures the source MAC address, and then closes the port.

It sends a RADIUS request using the MAC address as the username and password. The RADIUS server (e.g., Cisco ISE) decides to accept, reject, or apply specific policy (e.g., VLAN assignment, dACL).

MAB Decision Methods:

Static MAC entries (manually defined as authorized).

Profiling/analytics via Cisco ISE, which can validate devices based on:

- Vendor OUI
- Known behavior patterns
- Device fingerprinting

MAB is a last-resort method, MACs are easily spoofed and scaling large Databases of MAC addresses is inefficient, the only use-case for MAB would be devices such as Printers or IP-Cameras that are crucial but usually don't support Dot1x

WebAuth (Web Authentication)

WebAuth is a fallback access control method used when a device cannot support 802.1X but does have a web browser, such as user laptops or guest devices. It works by intercepting HTTP traffic and redirecting the user to a login portal for credential entry.

Like MAB, it is considered a last-resort mechanism, often used for guest access or non-managed user endpoints.

Types of WebAuth:

Local WebAuth

- Portal is hosted on the switch or WLC.
- Credentials are sent to a local RADIUS server.
- Basic features; limited customization and policy enforcement.

Best suited for small or static environments.

Central WebAuth (with Cisco ISE)

- The switch/WLC redirects to a centralized Cisco ISE portal.
- ISE handles the authentication and policy decision.
- Offers: Full customization (branding, flows), Post-login policy enforcement (VLAN/dACL/SGT), Guest management workflows

Common in enterprise guest or BYOD deployments, ISE may allow limited pre-auth access (e.g., DHCP, DNS) to reach the portal.

Enhanced Flexible Authentication (FlexAuth)

Cisco Enhanced FlexAuth, also referred to as *Access Session Manager*, is a mechanism that improves the efficiency of Network Access Control (NAC) by allowing concurrent use of multiple authentication methods—such as 802.1X, MAB, and WebAuth. This parallel approach eliminates the delay caused by sequential fallback (e.g., waiting for 802.1X to timeout before trying MAB or WebAuth), which is especially important for endpoints that don't support 802.1X (e.g., printers, phones).

Cisco Identity-Based Networking Services (IBNS)

IBNS is Cisco’s framework for enabling identity-aware network access control. It allows network services and policies to be dynamically applied based on the identity of the user or device, not just on traditional identifiers like IP or MAC addresses.

Core Concepts:

- Identity Sources: Users and devices are authenticated using 802.1X, MAB, or WebAuth. Cisco ISE is often the identity authority.
- SGTs (Security Group Tags): Once identity is known, devices/users are assigned an SGT, which acts as a scalable, tag-based identifier.
- Policy Enforcement: Network policies (e.g., ACLs, QoS, VLAN assignment) are enforced based on SGTs and other identity attributes, not IPs.
- TrustSec Integration: IBNS leverages Cisco TrustSec for scalable SGT-based policy enforcement across Layer 2 and Layer 3.
- FlexAuth (Access Session Manager): Allows concurrent authentication methods (802.1X, MAB, WebAuth) to reduce onboarding delays and ensure endpoint coverage.

IBNS essentially decouples network access control from IP addresses, allowing for dynamic, identity-driven security and policy in both wired and wireless environments

Cisco TrustSec

TrustSec is Cisco's identity-based access control framework that enforces security based on Security Group Tags (SGTs) rather than traditional IP/MAC filtering.

Core Components

- **SGTs**: Tags assigned by **Cisco ISE** during authentication to represent a user/device identity.
- **Enforcement**: Network devices (switches, firewalls) act on SGTs using **SGACLs** or **SGFW** policies.
- **Infrastructure**: Devices must support **SGT awareness** (via inline tagging or SXP propagation).

Three TrustSec Phases

1. **Ingress Classification**:
   - **Dynamic**: SGT assigned by ISE after 802.1X/MAB/WebAuth.
   - **Static**: Manually map SGTs to IP, VLAN, port (e.g., in data centers).
2. **Propagation**:
   - **Inline Tagging**: SGT is embedded in the Ethernet frame (Cisco-only).
   - **SXP**: IP-to-SGT mappings exchanged over TCP between devices.
3. **Egress Enforcement**:
   - SGTs are evaluated by policy (SGACLs or firewalls) at the network edge.

Notes on Flexibility

- A session/device can have **only one SGT at a time**.
- **VLAN and SGTs** are assigned independently but can be coordinated.
- TrustSec offers **scalable identity-based segmentation**, but not per-flow granularity.

MACsec Summary

MACsec is a Layer 2 encryption protocol that provides **on-the-wire encryption and authentication** of Ethernet frames.

It operates **hop-by-hop**: each switch decrypts and re-encrypts, so **every hop must support MACsec** for full-path protection.

Two keying methods:

- **MKA (MACsec Key Agreement)** – standards-based, enables **host-to-switch (downlink)** encryption.
- **SAP (Secure Association Protocol)** – Cisco proprietary, commonly used for **switch-to-switch (uplink)** encryption.

Use Cases:

- Protects against L2 threats in **untrusted physical environments** (e.g., colocation, exposed cabling).
- Often used to meet **compliance mandates** (FIPS, DoD, etc.).

Terminology:

- **Downlink MACsec** = Host ↔ Switch (requires 802.1X + MKA)
- **Uplink MACsec** = Switch ↔ Switch (SAP or MKA)

Network Device Access Control and Infrastructure Security

Access Control Lists (ACLs)

Access Control Lists (ACLs) are sequential rule sets composed of Access Control Entries (ACEs) that permit or deny traffic based on specified criteria. ACLs operate much like firewall rules — they are processed top-down, and the first matching entry determines the action taken. Any traffic that doesn't match an explicit permit statement is implicitly denied at the end.

Matching Criteria (Based on ACL Type):

ACLs can match on:

- Source IP address
- Destination IP address
- Wildcard mask
- Protocol (IP, TCP, UDP, ICMP, etc.)
- Source/destination ports (for TCP/UDP)
- MAC addresses (in MAC ACLs only)
- DSCP or precedence (in some extended ACLs)

Types of ACLs:

ACL TYPE	MATCH CRITERIA	NUMBER RANGE / FORM	USAGE CONTEXT
STANDARD (NUMBERED)	Source IP only	1–99, 1300–1999	Basic filtering, near destination
EXTENDED (NUMBERED)	Source/dest IP, port, protocol	100–199, 2000–2699	Detailed filtering, near source
NAMED ACL	Same as numbered, but human-readable	ip access-list ...	More scalable & readable
PORT ACL (PACL)	Any of the above, applied to **Layer 2 switchports**	—	Ingress filtering only
VLAN ACL (VACL)	Uses vlan access-map, matches IP/MAC	—	Filters bridged and routed VLAN traffic

ACLs are evaluated sequentially (top-down), First match wins; remaining entries are ignored, Implicit deny any is at the end if not explicitly matched.

ACLs use wildcard masks (inverted subnet masks):
For example, subnet 255.255.255.0 → wildcard 0.0.0.255

Direction of Application:

- **Ingress**: Traffic entering the interface
- **Egress**: Traffic leaving the interface
- Only **one ACL per direction per interface**

Standard, Extended and Named ACL Configuration

```
! --- Standard Numbered ACL: Deny all traffic from 192.168.1.0/24 ---
access-list 1 deny 192.168.1.0 0.0.0.255
access-list 1 permit any
! Applied typically to a routed interface (SVI or Layer 3 interface)

! --- Extended Numbered ACL: Deny traffic from 192.168.1.0/24 to 192.168.2.0/24 ---
access-list 101 deny ip 192.168.1.0 0.0.0.255 192.168.2.0 0.0.0.255
access-list 101 permit ip any any
! Applied near the source, to drop only specific traffic flows

! --- Named Standard ACL: Same as above but with a name ---
ip access-list standard DenyLAN1
 deny 192.168.1.0 0.0.0.255
 permit any
```

Extended, Named Port-ACL Configuration

```
! --- Named Extended ACL for use as PACL on L2 port ---
ip access-list extended Block_Host
 deny ip host 192.168.1.100 192.168.2.0 0.0.0.255
 permit ip any any

interface FastEthernet0/10
 switchport
 ip access-group Block_Host in
! This is a PACL: Applied to a Layer 2 port to filter IP traffic based on IP source/destination
```

Extended, Named Vlan-ACL Configuration

```
! --- VACL: Deny ICMP for VLAN 10, allow all else ---
ip access-list extended NoICMP
 deny icmp any any

ip access-list extended AllowRest
 permit ip any any

vlan access-map VACL-NoICMP 10
 match ip address NoICMP
 action drop

vlan access-map VACL-NoICMP 20
 match ip address AllowRest
 action forward

vlan filter VACL-NoICMP vlan-list 10
! This applies the VACL to all traffic in VLAN 10 (bridged or routed), drops ICMP, allows everything else
```

Important to know –

- Port-ACLs are only supported for ingress traffic (No outbound filtering support).
- Port-ACLs cannot filter L2 control packets such as CDP, STP etc.
- Port-ACLs do not support IPv6 or MPLS.
- When several types of ACLs are configured, Depending on “Bridged” vs “Routed”, this is how they’ll be processed –
    - Bridged
        - Inbound PACL
        - Inbound VACL
        - Outbound VACL
    - Routed
        - Inbound PACL
        - Inbound VACL
        - Inbound ACL
        - Outbound ACL
        - Outbound VACL
- “dACLs” or downloadable ACLs are another form of ACLs, they’re assigned dynamically via RADIUS after authentication (Used in ISE at SD-Access for exmaple).
- VACLs are the only ACL type where traffic that doesn't match any access-map clause is implicitly forwarded, not denied.

Terminal Lines and Password Protection

In Cisco IOS-XE, securing terminal access is the first line of defense against unauthorized configuration changes. CLI access must be protected using usernames, passwords, and optionally AAA frameworks. Unrestricted terminal access allows untrusted users full control of the device, potentially compromising the network.

Cisco IOS-XE provides three primary methods to access the CLI:

Console Access

- CLI: line con 0
- This is the physical console port, typically used for local, direct access via a serial cable, It's commonly used during initial setup, recovery, or out-of-band (OOB) access in case the network is down.
- Always available, even without network connectivity.

Auxiliary (AUX) Access

- CLI: line aux 0
- AUX is another physical port, functionally similar to console, but designed for remote out-of-band access using a modem over a telephone line (PPP/serial).
- Historically used for dial-in remote management, especially before broadband was common.
- Not used as often today but still supported.

Virtual Terminal Lines (VTY)

- CLI: line vty 0 15 (can vary by platform)
- VTY lines are logical interfaces used for remote CLI access over the network (e.g., via Telnet or SSH)
- These lines represent concurrent session slots (0 to 15 = 16 sessions).
- Must be secured with a password or AAA to prevent unauthorized remote logins.

All these different access points need to be properly secured via authentication to prevent unauthorized access, Cisco has 3 methods authentication ranked from least secure and centrelized to most –

- Configuring password directly over the line, For example –

  Line con 0

  Password !Q2w3e4r

  login

- Configuring Local Username & Password, for example –

  Username admin password !Q2w3e4r

  Line vty 0 4

  Login local

- Using an AAA server (Such as RADIUS/TACACS+) for authorization.

Password Types

TYPE	ALGORITHM	HASHED?	ENCRYPTED?	SECURITY LEVEL	NOTES
0	None	No	No	Not secure	Plaintext in config
5	MD5 + salt	Yes	No	Weak	Legacy, easily cracked
7	Vigenère cipher	No	Weakly	Not secure	Obfuscation only
8	PBKDF2 + SHA-256	Yes	No	Strong	Recommended
9	scrypt	Yes	No	Strong	Recommended, memory-hard

Configure Passwords at different strengths

```
! Type 9 (recommended)

enable algorithm-type scrypt secret MySecurePass


! Type 8 (PBKDF2 + SHA-256)

enable algorithm-type sha256 secret MySecurePass


! Type 5 (MD5, deprecated)

enable algorithm-type md5 secret MyOldPass
```

Service Password-encryption

- **Purpose**: Obfuscates all **Type 0 plaintext passwords**.
- **Method**: Converts them to **Type 7** (weak Vigenère cipher).
- **Scope**: Affects BGP keys, VTY/AUX passwords, SNMP strings, etc
- **Security**: **Not secure**, only deters casual viewing.

Username and password Authentication

A simple, local method to authenticate users, Offers accountability by identifying which user logged in (unlike shared enable secret).

Only the password is hashed or encrypted, the username is stored in plaintext.

Password Storage Methods:

Type 0 (Plaintext) –

```
username admin password mypass
```

Type 5 (MD5) –

```
username admin secret mypass
```

Note: On modern IOS-XE, this may auto-convert to Type 9.

Type 5 / 8 / 9 (Explicit)

```
username admin algorithm-type scrypt secret mypass  ! Type 9
username admin algorithm-type sha256 secret mypass  ! Type 8
username admin algorithm-type md5 secret mypass     ! Type 5 (some platforms may hide this option)
```

Configuring Line Local Password Authentication –

To enable password authentication on the line itself, the following commands are used –

```
Line con 0
Password xyz1234
Login
```

To ensure terminal access is secure, it's critical to know where authentication is applied. Securing only vty 0 4 leaves higher-numbered VTY lines unprotected. Similarly, leaving the console without a password allows unrestricted physical access.

Always verify all terminal lines are protected—console and the full VTY range. This prevents gaps in security and ensures all access points require proper authentication.

Privilege levels and Role based access control

In Cisco IOS-XE, privilege levels determine a user's access to configuration and operational commands.

Key Levels:

- Level 0: Minimal access (e.g., disable, enable, exit, logout, help)
- Level 1: *User EXEC mode* — allows basic monitoring (show, ping, etc.), but no configuration
- Level 15: *Privileged EXEC mode* — full access to all commands and configuration

Custom Levels (2–14):

Levels 2–14 can be used to assign specific commands for tailored roles. This adds flexibility but also **complexity** to manage.

Example, NOC Technician with Limited Show Access –

```
username NOC privilege 5 secret NoCEnG123
! Allow limited show commands at privilege level 5
privilege exec level 5 show
privilege exec level 5 show interfaces
privilege exec level 5 show ip interface brief
privilege exec level 5 show version
privilege exec level 5 show running-config
```

Access to the CLI (via VTY, AUX, or Console) can be secured using ACLs. This improves security by restricting access to specific IP addresses or subnets. All line types VTY, AUX, and even Console supports the access-class command. On modern platforms like CSR1000v running IOS-XE, access-class applies even to direct console access, not just remote sessions.

Enabling SSH And VTY Access

Telnet was once the standard for remote CLI access, but due to its lack of encryption, it poses a security risk by allowing credentials and session data to be sniffed.

To address this, SSH was introduced. SSH Version 1 provided basic encryption but suffered from multiple vulnerabilities. Modern networks should use SSH Version 2, which offers robust security, is NIST/FIPS certified, and suitable for current standards.

To enable SSH access on a Cisco device, follow these steps:

```
hostname R1                  ! Set the device hostname (required for RSA key generation)

ip domain-name example.com        ! Set the domain name (used in key generation for SSH)


crypto key generate rsa ! Generate RSA keys (2048-bit) for SSH encryption
```

You'll be prompted to choose the modulus size (range: 360–4096 bits). Higher values offer greater security but require more computational resources.

```
username admin privilege 15 password adminpass  ! Create a local admin user with privilege 15


line vty 0 15                ! Enter VTY line configuration (for remote access)

 login local                 ! Use local username/password for login

 transport input ssh             ! Allow only SSH connections on VTY lines


ip ssh version 2                ! Enforce use of SSH version 2 (more secure than v1)
```

Example of SSH Key

```
do show crypto key mypubkey rsa

% Key pair was generated at: 21:47:08 UTC May 23 2025

Key name: CSR1.roei

Key type: RSA KEYS

 Storage Device: not specified

 Usage: General Purpose Key

 Key is not exportable. Redundancy enabled.

 Key Data:

  305C300D 06092A86 4886F70D 01010105 00034B00 30480241 00A772F3 F3D8B87E

  EA36192A 400A243C 76645784 1BCF22EE 3195E198 2F32B635 B610077C 6B95C50B

  C5AE3922 115167CC BADBF843 E7A4D517 8154F5D2 643BC250 15020301 0001
```

Authentication, Authorization and Accounting

AAA (Authentication, Authorization, and Accounting) is a security framework composed of three independent components, used to enhance access control and monitoring—both within and beyond networking environments.

- **Authentication** verifies the identity of users or devices through credentials such as usernames/passwords, certificates, or keys.
- **Authorization** grants authenticated users specific access rights to resources, devices, or network segments.
- **Accounting** tracks and logs user actions for auditing and analysis, recording what was done and where.

In networking, AAA is primarily used for two purposes:

1. Authenticating and authorizing administrative access to network devices (e.g., routers, switches).
2. Authenticating users when they connect to the network (e.g., through wireless APs or switch ports).

The ENCOR exam emphasizes the distinction between the two main AAA protocols:

- **TACACS+** is a Cisco proprietary protocol that operates over TCP port 49. It separates authentication, authorization, and accounting into discrete processes, making it the preferred choice for managing access to network devices.
- **RADIUS** is an open standard developed by the IETF. It combines authentication and authorization, and uses UDP (typically ports 1812/1813). Because it supports EAP (Extensible Authentication Protocol), it is commonly used for user authentication, particularly in wireless and 802.1X environments, and plays a central role in Software-Defined Access (SD-Access).

FEATURE	RADIUS	TACACS+
PROTOCOL	UDP	TCP
DEFAULT PORTS	1812 (Authentication), 1813 (Accounting)	49
ENCRYPTION SCOPE	Encrypts only the password	Encrypts the entire payload
AUTHENTICATION & AUTHORIZATION	Combined	Separated
ACCOUNTING SUPPORT	Yes	Yes
PRIMARY USE CASE	User authentication (e.g., 802.1X, VPNs)	Administrative device access (e.g., SSH)

The roles for each AAA protocol is best explained via examples –

RADIUS Role – Network Access Authentication

Imagine a new employee connecting their laptop to the corporate network. For maximum security and flexibility, the network uses 802.1X (dot1x) on access switches and wireless LAN controllers.

- RADIUS supports 802.1X and acts as the backend protocol between the access device (authenticator) and the authentication server (typically Cisco ISE).
- It enables the use of EAP methods such as EAP-TLS, PEAP, or EAP-FAST, allowing secure credential exchange, certificate validation, and dynamic policy enforcement.
- TACACS+ does not support EAP and therefore cannot be used for 802.1X.

Conclusion: RADIUS is inherently more flexible for network access authentication, making it the standard for onboarding users to the network.

TACACS+ Role – Device Access Control

Now consider two users: a NOC engineer and a network administrator, both accessing a Cisco router's CLI via SSH.

- With TACACS+, authentication (who they are) and authorization (what they can do) are handled separately.
- This allows the router to send every CLI command entered by each user to the TACACS+ server for real-time authorization decisions.
    - For example, the engineer may be allowed only show and ping commands.
    - The administrator may have full configuration privileges.
- RADIUS cannot authorize at the per-command level, since it combines authentication and authorization and lacks the granularity required for CLI command control.

Conclusion: TACACS+ provides superior control and visibility for administrative access to network devices.

Configuration of AAA

```
! Step 1: (Optional) Configure a fallback local user with privilege 15

username backup-admin privilege 15 algorithm-type sha256 secret 1234

! Used if TACACS+ is unreachable. Prevents lockout.

! Step 2: Enable AAA on the device

aaa new-model

! Activates AAA feature set, required before using any AAA commands.

! Step 3: Configure the TACACS+ server (choose one style based on IOS version)

! --- For IOS versions prior to 16.12.2 ---

tacacs-server host 10.1.1.100 key MySecretKey

! Defines the TACACS+ server IP and shared secret used for authentication and encryption.

! --- For IOS-XE 16.12.2 and newer ---

tacacs server TACACS1

 address ipv4 10.1.1.100

 key MySecretKey

! Server name "TACACS1" is just a label. "address" can be an IP or DNS name. "key" is the shared secret.

! Step 4: Configure AAA authentication for login

aaa authentication login default group tacacs+ local if-authenticated

! Tries TACACS+ first, falls back to local database if unreachable. "if-authenticated" permits access if all methods fail.

! Step 5: Enable AAA authorization for EXEC shell (privileged access)

aaa authorization exec default group tacacs+ local if-authenticated

! Controls whether a user can enter EXEC mode (e.g., after login). Without this, users may enter CLI regardless of role.

! Optional: Authorization for console access

aaa authorization console

! Explicitly enables authorization on the console line.

! Step 6: Enable AAA command authorization

aaa authorization commands 15 default group tacacs+ local if-authenticated

! Authorizes every command at privilege level 15. Without this, user can run any command after login.

! Step 7: Enable command accounting

aaa accounting commands 15 default start-stop group tacacs+

! Logs each command run by users at privilege level 15. Useful for auditing and tracking.

! Step 8: Enable session/accounting logging for login/logout

aaa accounting exec default start-stop group tacacs+

! Tracks user sessions (when a user logs in and exits).
```

Zone-Based Firewall (ZBFW)

Properties of ZBFW –

- A stateful firewall feature in IOS/IOS XE routers.
- Built-in security that mirrors ASA-like zone control.
- Uses zones and policies to inspect and allow traffic between different interfaces.
- Traffic between interfaces in the same zone is always allowed.
- Traffic between different zones is denied by default unless allowed via policy.

Built-in Zones

self: Represents the router's control/management plane.

default: All interfaces not assigned to a zone (traffic dropped unless explicitly permitted).

Understanding inspect, pass, and drop

- inspect: Bidirectional and stateful. Return traffic is automatically allowed based on the established session.
- pass: One-way only. Return traffic is blocked unless a separate reverse zone-pair with pass or inspect is configured.
- drop: Traffic is explicitly denied. Can optionally generate logs depending on the platform and configuration.

Configuring ZBFW –

The goal in this example is to enable bidirectional communication between the OUTSIDE zone and the DMZ zone for DNS, DHCP, HTTP, and HTTPS traffic. All other traffic will be dropped. This demonstrates how to build a ZBFW policy that inspects and allows specific service flows using class maps and policy maps.

```
! Step 1: Define zones

zone security INSIDE

zone security OUTSIDE

zone security DMZ


! Step 2: Create ACLs

ip access-list extended ACL-DNS-DHCP

 permit udp any any eq domain

 permit udp any any eq bootps


ip access-list extended ACL-WEB

 permit tcp any any eq 80

 permit tcp any any eq 443
```

```
! Step 3: Create class maps
class-map type inspect match-any CLASS-DNS-DHCP
 match access-group name ACL-DNS-DHCP

class-map type inspect match-any CLASS-WEB
 match access-group name ACL-WEB

! Step 4: Define policy maps
policy-map type inspect POLICY-OUTSIDE-TO-DMZ
 class type inspect CLASS-DNS-DHCP
  inspect
 class type inspect CLASS-WEB
  inspect
 class class-default
  drop

! Step 5: Apply policy to zone pair
zone-pair security OUTSIDE-TO-DMZ source OUTSIDE destination DMZ
 service-policy type inspect POLICY-OUTSIDE-TO-DMZ

! Step 6: Assign interfaces to zones
interface GigabitEthernet0/0
 zone-member security OUTSIDE

interface GigabitEthernet0/1
 zone-member security DMZ
```

📎 *Verification Commands*

```
show zone security
show zone-pair security
show policy-map type inspect
show policy-map type inspect zone-pair
show class-map type inspect
```

CoPP

Purpose:
CoPP protects the router's **CPU (control plane)** from being overwhelmed by excessive or malicious traffic, using QoS policing techniques applied directly to the **control-plane interface**. Control-plane traffic (e.g., routing updates, ICMP, SSH) bypasses ASIC/CEF and is punted to the CPU — making it more sensitive and attack-prone.

Design Considerations:

- Misconfigured CoPP may drop critical control traffic like OSPF Hellos or BGP sessions.
- Traffic is grouped by function via **ACLs**, **class-maps**, and a **policy-map** that defines bandwidth and actions.
- A class-default catch-all is used to drop or police unknown traffic.
- CoPP is applied to the logical control-plane interface (not physical interfaces).

The benefits of a good CoPP configuration can bring visibility to control-plane traffic, ensure CPU usage is kept to a minimum and only for the needed protocols whilst keeping the control-plane intact.

Some cisco platforms have default CoPP policies pre-configured (Such as the 9000 series).

CoPP Configuration Example

```
! Step 1: Define ACLs to match control-plane traffic types
ip access-list extended ACL-CoPP-Routing
 permit ospf any host 224.0.0.5
 permit ospf any host 224.0.0.6
 permit tcp any eq bgp any
 permit eigrp any host 224.0.0.10

ip access-list extended ACL-CoPP-ICMP
 permit icmp any any echo
 permit icmp any any echo-reply
 permit icmp any any unreachable
 permit icmp any any ttl-exceeded

! Step 2: Match traffic with class-maps
class-map match-all CLASS-CoPP-Routing
 match access-group name ACL-CoPP-Routing

class-map match-all CLASS-CoPP-ICMP
 match access-group name ACL-CoPP-ICMP

! Step 3: Define policy-map actions per class
policy-map POLICY-CoPP
 class CLASS-CoPP-Routing
  police 64000 conform-action transmit exceed-action transmit violate-action drop

 class CLASS-CoPP-ICMP
  police 8000 conform-action transmit exceed-action transmit violate-action drop

 class class-default
  police 4000 conform-action transmit exceed-action drop violate-action drop

! Step 4: Apply to control-plane
control-plane
 service-policy input POLICY-CoPP
```

Device Hardening

Device Hardening refers to manually securing network devices by disabling unnecessary features, especially on outbound interfaces or high-risk systems. Key measures include:

Disable topology discovery tools (LLDP/CDP)

Disable globally or per interface:

```
! Disable CDP globally
no cdp run


! Disable CDP on an interface
interface <interface>
 no cdp enable


! Disable LLDP globally
no lldp run


! Disable LLDP on an interface
interface <interface>
 no lldp transmit
 no lldp receive
```

Enable TCP keepalives

```
! Enable incoming TCP keepalives
service tcp-keepalives-in


! Enable outgoing TCP keepalives
service tcp-keepalives-out
```

Disable IP redirects

Recommended on Layer 3 interfaces:

```
interface <interface>
 no ip redirects
```

Disable Proxy ARP

```
interface <interface>
 no ip proxy-arp
```

Disable service configuration (for remote config loading)

```
no service config
```

Disable MOP (Maintenance Operation Protocol)

Commonly disabled on serial or Ethernet interfaces:

```
interface <interface>
 no mop enabled
```

Disable PAD (Packet Assembler/Disassembler)

Legacy protocol, disable globally:

```
no service pad
```

These commands help harden a device by reducing attack surfaces and limiting unneeded protocol exposure.

Virtualization

Virtualization allows multiple virtual machines (VMs) to run on a single physical host, improving resource utilization and operational flexibility. It emerged due to inefficiencies in deploying separate physical servers for each workload, where hardware was often underutilized.

For example**,** A standalone Windows Server may consume high memory/CPU due to OS overhead but underutilize application resources. Virtualization mitigates this by consolidating workloads onto fewer physical devices.

Hypervisors

A **Hypervisor** is the software layer that enables virtualization. Two types exist:

Type 1 – Runs directly on harware "Bare metal", such as ESXi or Hyper-V

Type 2 – Runs on host OS as an application, such as VMware or VirtualBox

Benefits of Virtual Machines

1. **Rapid Provisioning** – VMs can be instantiated programmatically and on demand.
2. **Flexible Resource Allocation** – CPU, memory, and other resources can be adjusted dynamically (subject to hypervisor type and storage limits).
3. **Mobility** – VMs can be live-migrated between hosts, enhancing **availability**.
4. **Improved Resource Utilization** – Consolidation reduces hardware footprint and allows efficient scaling.

Containers – Application Virtualization

Containers virtualize at the **application level**, not the hardware level. Unlike Virtual Machines (VMs), which emulate entire hardware stacks, containers isolate and run individual applications with all their dependencies in a lightweight, consistent environment.

ASPECT	CONTAINERS	VIRTUAL MACHINES (VMS)
ABSTRACTION LEVEL	**Application layer**	**Hardware layer**
GUEST OS	Shared host kernel	Full OS per VM
OVERHEAD	Low (no full OS per container)	Higher (OS + hypervisor per VM)
ISOLATION	Process-level (via namespaces, cgroups)	Full system-level (via hypervisor)
BOOT TIME	Seconds	Minutes

How Containers Work

- Each container runs as an **isolated process** on the host OS.
- Containers **bundle code and dependencies** (e.g., libraries, binaries) inside an image.
- The host OS **kernel is shared** across all containers.
- A **container engine** is required to manage containers.

Without the engine, a container image is inert — it's the engine that makes it executable.

Container Engine

A **container engine** starts, stops, and manages containers. Popular engines:

- **Docker** – Most common in development and production.
- **containerd** – Low-level container runtime (used by Docker internally).
- **CRI-O** – Kubernetes-native runtime.
- **rkt** (CoreOS) – Now deprecated but used in some legacy systems.

Use Case Benefit

Containers provide:

- **Fast deployment** and startup.
- **Consistency** across environments (dev, test, prod).
- **Isolation** without the overhead of full VMs.
- **Better scalability** for microservice-based architectures.

Virtual Switch

A **Virtual Switch** is like a software-based Ethernet switch that operates **inside the hypervisor** or **container engine**. It connects **VMs or containers** on the same host so they can communicate with each other or with external networks.

Think of it as:

"If VMs are virtual servers, the virtual switch is their virtual network rack."

Key Points:

- **Built into hypervisors**: Both Type 1 (like ESXi) and Type 2 (like VMware Workstation) hypervisors include virtual switch functionality.
- **Docker too**: Docker sets up a **bridge network** by default (usually called docker0), acting like a simple virtual switch for containers.
- **Same host = same broadcast domain** by default — so intra-node communication just works.
- **You can segment them**:
    - VLAN tagging
    - Virtual port groups
    - Isolated bridge networks in Docker
- **You can apply policies**:
    - **ACLs** or **firewall rules**
    - **QoS** (on hypervisor-managed vSwitches)
    - **MAC filtering**, **traffic shaping**

Real Examples:

- **ESXi** has vSwitch0, vDS (distributed switches)
- **Hyper-V** uses “External,” “Internal,” and “Private” virtual switches
- **Docker** uses bridge, host, and overlay networks (with iptables rules behind the scenes)

NFV in Simple Terms

NFV is about turning what used to be dedicated network appliances (firewalls, load balancers, routers, WAN optimizers, etc.) into software, and running that software on standard x86 servers — no need for special hardware or ASICs anymore.

It's a major shift from:

"One box = one function = one vendor" to, "Any function = any server = vendor-neutral"

COMPONENT	DESCRIPTION
VNF (VIRTUAL NETWORK FUNCTION)	The actual software version of a network function (e.g., virtual ASA, pfSense)
NFVI (NFV INFRASTRUCTURE)	The physical servers, storage, and networking + hypervisor/virtualization layer
MANO (MANAGEMENT & ORCHESTRATION)	Tools to deploy, manage, and scale VNFs (e.g., OpenStack, Cisco ESC)

Why NFV Matters

- **No vendor lock-in** – You're free to use pfSense + Suricata + HAProxy instead of buying Cisco/Juniper/F5 hardware.
- **Fast rollouts** – Spin up a new firewall instance in seconds.
- **DevOps-friendly** – Treat network infrastructure as code.
- **Cloud-native fit** – Ties well into containerized environments and SDN.

Inbound Packet Flow (Heavily simplified):

1. **pNIC → RAM (via DMA)**
    - NIC uses DMA to place the packet in a ring buffer in host memory.
2. **CPU → vSwitch logic**
    - CPU is triggered (via interrupt/polling) to process the packet and run the vSwitch lookup logic.
3. **vSwitch → guest vNIC (copy or map to guest memory)**
    - Packet is delivered to the VM's vNIC receive queue.
4. **vCPU → processes packet in guest**
    - Hypervisor schedules the VM, and the guest OS's vCPU reads the packet from the vNIC buffer.

This process illustrates how the VNF receives traffic within a server from Physical NICs (pNIC) and how the CPU processes it to send it correctly.

OVS-DPDK in Simple Terms

Normally, packets go through the kernel, which is safe but slow due to checks, interrupts, and system calls. DPDK moves packet processing to user space, skipping the kernel to make things much faster. You give dedicated CPU cores to do this work — they constantly poll the NIC, so there's no waiting, but it uses more CPU

Why It's Fast

- **No context switches**
- **No syscalls**
- **No interrupt delays**
- **No kernel overhead**

It's like bypassing the security line at an airport and walking straight to your plane.

The Catch

- Less secure (bypasses OS protections)
- Uses more CPU (dedicated cores)
- Needs manual tuning (core pinning, hugepages, etc.)

Just like RDMA or GPU DMA — it's all about direct access, no middlemen, and maximum speed.

OVS-DPDK is a software that can only be installed on user-space in Linux, meaning It can either work on VMWare (Linux .iso) or Bare-metal Linux, making it use-case specific.

PCI Passthrough

PCI Pass-through is hardware-level NIC handoff to a VNF — best for max speed, but at the cost of flexibility and scalability.
It’s like giving a VNF its own physical NIC — no sharing, no switching, just raw access.

Benefits:

- **Bypasses vSwitch** and hypervisor
- **No CPU overhead** from shared switching
- **Ultra-low latency**
- **Max throughput** (near line-rate)

Tradeoffs:

- You **lose flexibility**:
  - That NIC is **no longer shared** with other VMs/VNFs
  - No vSwitch = **no ACLs, QoS, VLAN tagging**, etc. unless done inside the VNF
- Requires:
  - **IOMMU / VT-d** (Intel) or **AMD-Vi** support
  - BIOS + hypervisor configuration
- Limited by how many NICs you physically have

SR-IOV

SR-IOV (Single Root I/O Virtualization) is a PCIe feature that allows a single physical NIC to expose multiple lightweight virtual functions (VFs), which can be directly assigned to VMs or VNFs.

Key Points:

- **Enables direct NIC access** per VM, bypassing the hypervisor
- **Each VF acts like a mini NIC**
- **High performance**, low CPU overhead (like passthrough)
- **Still shareable** — one physical NIC can serve many VMs

Use Case:

Perfect for VNFs needing:

- Dedicated traffic paths
- Low-latency, high-throughput NIC access
- Hardware isolation without dedicating the entire NIC

Think of it as **"passthrough, but shareable."**

Cisco ENFV – Enterprise Network Function Virtualization

Cisco ENFV is Cisco's solution for deploying **VNFs at scale** across **branch offices**, focused on:

- **Flexibility**
- **Reduced hardware footprint**
- **Centralized management**
- **High availability**

Why ENFV?

- Branches are **resource-constrained** (space, power, no on-site IT)
- VNFs run on **x86 hardware**, replacing physical routers, firewalls, LBs, etc.
- Managed centrally via **Cisco DNA Center**
- Reduces:
  - Initial deployment complexity
  - Hardware cost (no need for purpose-built boxes)
  - Operational overhead

Architecture (based on ETSI NFV)

LAYER	ROLE
1. MANO (MANAGEMENT & ORCHESTRATION)	Deploys, automates, and monitors VNFs — integrated with **Cisco DNA Center**
2. VNFS	The virtualized services (e.g., router, firewall, DHCP, DNS, load balancer)
3. NFVIS	Cisco's hypervisor-like OS that enables virtualization (similar to ESXi)
4. HARDWARE	x86 server platform (CSP-5000, UCS, or custom white box)

Key Benefits

- Centralized provisioning via **DNA Center**
- Supports **SD-WAN** and **SD-Access**
- Redundancy via multiple x86 nodes
- Hardware agnostic (custom-spec x86 servers)
- VNFs are portable, scalable, and replace rigid hardware appliances

Network Programmability Concepts

The Command Line Interface (CLI) is the traditional and most recognized method of configuring devices—whether in Linux, Windows CMD, or Cisco IOS-XE.

A major cause of network outages is human error—typically due to misconfiguration or misunderstanding of how protocols and architectures interact. Unlike hardware or software faults, human error is preventable.

Network programmability aims to reduce these errors by automating repetitive tasks, improving consistency, and accelerating provisioning. This approach leads to faster, more efficient, and less error-prone operations.

While CLI remains useful for its transparency and documentation, it struggles with scalability, complexity, and intuitiveness. Programmability helps address these limitations by introducing templates, automation tools, and streamlined workflows.

Northbound API: Interfaces used by external applications or automation tools to communicate **with a controller** (e.g., Cisco DNA Center) for retrieving data, pushing policies, or initiating actions.

Southbound API: Interfaces used **by the controller** to interact with **network devices** (routers, switches, access points) using protocols such as RESTCONF, NETCONF, or CLI over SSH.

The terms *northbound* and *southbound* are only relevant in architectures that include a controller. Without a controller, direct automation (e.g., Ansible or Python scripts targeting devices) does not involve these distinctions.

API (Application Programming Interface): A structured, standardized way for software components to exchange data and commands. Typically implemented over HTTP/S using ports like TCP 443, APIs define the available operations (e.g., GET, POST) and the expected input/output format (commonly JSON or XML). APIs are not network ports themselves—they operate *through* them.

RESTful APIs are a type of API that follow the REST architectural style and use **HTTP or HTTPS** for communication. They leverage standard HTTP methods to perform operations on resources, typically in a stateless way.

The main HTTP methods used in RESTful APIs are:

- **GET** – Retrieve data from a specific resource (e.g., fetching a user profile)
- **POST** – Submit new data to a resource (e.g., creating a new user or sending credentials)
- **PUT** – Replace an existing resource entirely (e.g., updating an address)
- **PATCH** – Modify part of an existing resource (e.g., changing a single field)
- **DELETE** – Remove a resource (e.g., deleting an account)

These align closely with **CRUD** operations commonly used in databases:

- **Create** – Corresponds to POST
- **Read** – Corresponds to GET
- **Update** – Corresponds to PUT or PATCH
- **Delete** – Corresponds to DELETE

While PUT and PATCH are both used to update data, PUT usually **replaces the entire object**, whereas PATCH **makes partial changes**.

RESTful APIs are Stateless, but its important to note they're stateless in the sense that servers do not remember anything between requests, and that each request must carry all necessary data for the server to understand and process it.

Common HTTP/S Response Codes

While we often expect API interactions to work seamlessly, reality is often disappoint, Understanding HTTP response codes is essential for diagnosing issues when working with RESTful APIs (or any HTTP-based protocol).

These status codes are returned by the server in response to HTTP methods like GET, POST, PUT, etc., and help classify success, client-side errors, and server-side errors.

CODE	MEANING	DESCRIPTION
200	**OK**	**The request was successful (common for GET, POST, etc.).**
201	**Created**	**A new resource was successfully created (commonly returned by POST).**
400	**Bad Request**	**The client sent malformed or invalid data (e.g., wrong syntax or missing fields).**
401	**Unauthorized**	**Authentication is missing or invalid (e.g., no API key or token expired).**
403	**Forbidden**	**Credentials are valid, but access to the resource is not allowed.**
404	**Not Found**	**The URL or endpoint doesn't exist — often a typo or wrong route.**

Little tip, 4xx Codes usually reflect a client related issues whereas 500 codes reflect a server side issue.

Postman

Postman is a desktop and web-based tool used to test, send, and manage HTTP/S API requests. It’s widely used for learning, debugging, and developing RESTful APIs.

Key Uses

- **Send API requests** using standard HTTP methods: GET, POST, PUT, PATCH, DELETE
- **Set custom headers**, authentication, query parameters, and request bodies
- **View and analyze responses** in real time (status code, headers, JSON body)
- **Automate tests** for API behavior and correctness (basic scripting)
- **Organize API calls** into **collections** for reuse and documentation

Data Formats in API Communication

When using APIs (RESTful or not), the structure of the data being exchanged is critical. Below are the two most common data formats used today in network programmability and automation:

XML (eXtensible Markup Language)

XML is an older, tag-based formatting language — visually similar to HTML. It's highly structured and verbose, and still widely used in protocols like NETCONF.

- Each element (tag) must be properly **opened and closed**
- Data is **wrapped inside relevant tags**
- Indentation is optional but recommended for readability

Example of XML –

```
<users>
  <user>
    <name>Roei</name>
  </user>
  <user>
    <name>Idan</name>
  </user>
</users>
```

JSON (JavaScript Object Notation)

JSON is a more modern data format and is the default for most **RESTful APIs** (including RESTCONF). It's favored for being lighter, cleaner, and easier to parse both manually and programmatically.

- Structured as **key-value pairs**
- Formatted as **objects ({})** and **arrays ([])**
- Strict on syntax: uses double quotes for keys and strings

Example of JSON –

```
{
  "admin": "Roei",
  "client": "Idan"
}
```

Cisco DNA Center APIs – Overview and Usage

Cisco DNA Center (DNAC) provides a rich set of RESTful APIs designed to simplify network programmability and automation. All communication is performed over HTTP/S, and data is exchanged in JSON format.

Authentication

Before making any API calls, authentication is required. DNAC uses **token-based authentication**, meaning:

- You authenticate **once** using a POST request with your username and password.
- The server responds with an **access token**.
- You reuse this token in the headers of all subsequent requests.

Authentication Endpoint (POST)

POST https://sandboxdnac.cisco.com/api/system/v1/auth/token

Auth Method: Basic Auth

- **Username**: devnetuser
- **Password**: Cisco123!

Response: A token like:

```
{
  "Token": "eyJhbGciOiJFUzI1NiIsInR5cCI6IkpXVCJ9..."
}
```

Use the Token in All Future Requests

For all API requests, include this in the headers:

Key: X-Auth-Token

Value: <your_token>

Example Request (GET)

GET https://sandboxdnac.cisco.com/api/v1/network-device

Headers:

X-Auth-Token: <your_token>

```
{
  "hostname": "sw1",
  "managementIpAddress": "10.10.20.175",
  "softwareVersion": "17.12.1prd9",
  "type": "Cisco Catalyst 9000 UADP 8 Port Virtual Switch",
  "upTime": "71 days, 0:02:30.00",
  "reachabilityStatus": "Reachable"
}
```

Data Models

What is a Data Model?

A data model defines the structure, fields, and data types required when sending or receiving data via APIs.

It ensures data is organized, consistent, and machine-readable, providing a standard way to represent both configuration and operational data.

Why It Matters

When interacting with APIs (e.g., RESTCONF or NETCONF), you must:

- Use the appropriate data model (e.g., YANG, OpenConfig, native)
- Match the defined structure exactly
- Populate fields with correct data types

Example Data Model –

```
{
  "interface": {
    "name": "string",
    "enabled": "boolean"
  }
}
```

Valid Configuration Data –

```
{
  "interface": {
    "name": "GigabitEthernet2",
    "enabled": true
  }
}
```

If the structure or data types are incorrect, the request will fail due to schema validation.

Data Models

- **YANG** (data model)
- **NETCONF** (protocol using YANG over SSH/XML)
- **RESTCONF** (RESTful protocol using YANG over HTTP/JSON or XML

YANG defines the schema like a class or form. Your config or query results are the instantiated object, each interface is a branch, and name, enabled, speed, etc., are leaves.

```
Branch: interface (container/list)
 └─ Leaf: name        (string)
 └─ Leaf: description  (string)
 └─ Leaf: enabled      (boolean)
 └─ Leaf: mtu         (uint16)
 └─ Leaf: mac-address  (read-only)
 └─ Leaf: speed       (enum)
```

You can consider each interface (e.g., GigabitEthernet1/0/1) as a ***branch****, and each of its attributes (name, enabled, mac-address, etc.) as* ***leaves****.*

Pulled Config as the Instance (Data) –

When you pull configuration or operational state, the output (in JSON/XML) is **just the data filled into that structure**, like:

```
{
  "interface": {
    "name": "GigabitEthernet1/0/1",
    "description": "Uplink",
    "enabled": true,
    "mtu": 1500,
    "mac-address": "00:11:22:33:44:55",
    "speed": "1gbps"
  }
}
```

This JSON matches the YANG blueprint exactly — if it doesn't, the device will reject the config or the query.

NETCONF

- **Transport**: SSH (port 830)
- **Encoding**: XML
- **Data Model**: YANG-based
- **API Style**: RPC (stateful, uses <rpc>, <edit-config>, etc.)
- **Key Features**:
    - Full config retrieval (<get-config>)
    - Transaction support (lock, commit, rollback)
    - Schema-driven, ideal for deep programmatic config
- **Use Case**: Precise, transaction-safe configuration (e.g., SD-WAN)

RESTCONF

- **Transport**: HTTP/HTTPS (port 443)
- **Encoding**: JSON or XML
- **Data Model**: YANG-based
- **API Style**: RESTful (GET, POST, PUT, PATCH, DELETE)
- **Key Features**:
    - Simpler than NETCONF
    - Easily scriptable (curl, Postman, Python)
    - Aligned with web standards
- **Use Case**: Lightweight, human-readable automation (e.g., DNAC to IOS-XE)

Example: GET Request to 192.168.1.23 (Catalyst Switch)

NETCONF

***Transport**: SSH (port 830)*

***Protocol**: RPC-based (not REST)*

Data Model**: Based on **YANG

Encoding**: **XML

***Session**: Stateful (requires opening a persistent NETCONF session)*

RESTCONF

***Transport**: HTTP/HTTPS (port 443)*

***Protocol**: RESTful (standard verbs like GET, POST, PUT, DELETE)*

Data Model**: Based on **YANG

Encoding**: **JSON** or **XML

***Session**: Stateless (standard HTTP request/response)*

GitHub (ENCOR Context)

GitHub, specifically in the context of ENCOR, is primarily used for **version control**. It's a common platform for developers and network engineers to store code, collaborate, review changes, and track project history over time.

A **Git repository** is used to store a project. Files can be **created, removed, or updated** within it, and the repository tracks every change.

There are several key git commands used to manage this workflow:

COMMAND	DESCRIPTION
GIT INIT	Initializes a new Git repository locally
GIT CLONE	Copies an existing remote repository to your local machine
GIT STATUS	Shows the current state of the working directory and staged changes
GIT ADD	Stages a file or set of files to be committed
GIT COMMIT	Saves the staged changes as a new version (commit) in the local repo
GIT PUSH	Sends committed changes to the remote repository (e.g., GitHub)
GIT PULL	Fetches and merges updates from the remote repository into the local repo
GIT FETCH	Retrieves updates from the remote without merging
GIT MERGE	Combines changes from another branch or source into the current branch
GIT BRANCH	Views or creates branches for isolated development
GIT CHECKOUT	Switches to a different branch or commit
GIT LOG	Views the commit history
GIT DIFF	Shows the differences between files or commits

These commands form the foundation of collaborative version control, aligning with ENCOR automation and DevOps topics where Git is often used for configuration-as-code and infrastructure versioning.

Automation Tools

EEM – Embedded Event Manager

EEM is a flexible, on-device automation framework built into Cisco IOS/IOS-XE. It enables network engineers to write event-driven scripts (applets) that proactively monitor and respond to system events.

Unlike external agent-based or polling systems, EEM runs natively on the device, offering faster and more granular responses to specific conditions — such as interface flaps, CPU spikes, or syslog messages.

Conceptual Summary

- Think of EEM as an "if/then" automation system, similar to conditional scripting in Python.
- The key elements are:
  - Events – triggers that initiate the applet (e.g., syslog messages, SNMP traps, timers).
  - Actions – responses executed by the device (e.g., send syslog, run CLI commands, send email).

Example: Loopback Interface Recovery

```
event manager session cli username poyke


event manager applet Loop0
 event syslog pattern "Loopback0.*down"
 action 1.0 syslog msg "EEM TRIGGERED: Loopback0 went down"
 action 2.0 cli command "enable"
 action 3.0 cli command "conf t"
 action 4.0 cli command "interface Loopback0"
 action 5.0 cli command "no shut"
 action 6.0 cli command "do show interface Loopback0"
 action 7.0 syslog msg "Loopback0 has been brought back up"
```

Key Operational Notes

- event manager session cli username is mandatory when using cli command actions.
  This user must have privilege level 15.
- EEM applets execute in exec/config mode as if typed manually — hence enable and conf t are required.
- Latency: EEM is reactive but not instantaneous. In testing, ~3 packets may be dropped before the interface is restored. This makes it faster than protocol Hello timers, but not as fast as BFD.

EEM + Tcl: High CPU Auto-Logging

Use Case:

In resource-constrained environments, a CPU spike can indicate serious issues — such as route flapping, DoS attempts, or control-plane overload. This script automatically monitors CPU usage every 60 seconds and, if a threshold is crossed, logs the top CPU-consuming processes to flash.

Concept:

- **EEM (Embedded Event Manager)** runs every 60 seconds via a watchdog timer.
- If CPU utilization ≥ 80%, it triggers a Tcl script stored in flash.
- The Tcl script saves a timestamped snapshot of CPU state to a local file.

Script: flash:/cpu-monitor.tcl

```
set fh [open "flash:/cpu-alert-[clock format [clock seconds] -format %Y%m%d-%H%M].log" w]

puts $fh "==== CPU Alert Triggered at [clock format [clock seconds]] ===="

puts $fh [exec "show processes cpu sorted | include CPU utilization"]

puts $fh [exec "show processes cpu sorted | include PID"]

close $fh
```

EEM Applet: cpu-monitor

```
event manager applet cpu-monitor

 event timer watchdog time 60 maxrun 30

 action 1.0 cli command "enable"

 action 2.0 cli command "show processes cpu | include one minute"

 action 3.0 regexp {(\d+)\%} "$_cli_result" match cpu

 action 4.0 if $cpu ge 80

 action 4.1  cli command "tclsh flash:/cpu-monitor.tcl"

 action 4.2  syslog msg "High CPU detected snapshot saved to flash"

 action 4.3 end
```

Notes:

- Threshold (80%) can be adjusted to suit operational policies
- Logs can be retrieved later for auditing or RCA (Root Cause Analysis)
- You can enhance this by adding scp: or email alert capabilities if needed

Agent Based Automation Tools

Puppet – Agent-Based Orchestration for Infrastructure Automation

Puppet is an **agent-based automation and orchestration tool** used to manage the entire lifecycle of infrastructure devices and services — from provisioning to configuration and compliance enforcement.

It is designed for **large-scale deployments**, and can manage thousands of nodes with consistent, repeatable configuration logic.

Architecture

- **Puppet Server (Master)**: Central control node that stores configuration files (manifests), modules, and templates. Compiles them into catalogs.
- **Puppet Agent**: Installed on managed nodes. Pulls configuration (catalog) from the Puppet Server at regular intervals (usually every 30 minutes).
- Communication is done over **SSL-encrypted channels**, ensuring authenticity and confidentiality.

Puppet on Cisco

Puppet is supported on several Cisco platforms, particularly those with **Linux-based environments** or **Puppet agent support**, such as:

- **Cisco UCS**
- **Cisco Catalyst 9000**
- **Cisco Nexus (NX-OS with Bash access)**
- Generic **Linux-based appliances or VMs**

For Cisco IOS, direct native support is limited — but **Puppet Labs provides the cisco_ios module** to simulate Puppet behavior over SSH or APIs. These use declarative DSL to configure IOS/XE devices.

Puppet Components

Manifests (*.pp)

- The primary configuration files that define the desired state using Puppet's DSL.
- Example:

```
Banner
 {
 'motd':
  motd => 'Illegal Entry results in execution',
}
```

Modules

- Collections of manifests, templates, files, and metadata bundled together to manage a specific service or device type.
- Cisco uses the **puppetlabs-cisco_ios** module for managing IOS routers/switches declaratively.

Templates (.erb)

- Embedded Ruby files used to dynamically generate configuration content based on variables or logic.
- Example use case: Creating different SNMP config blocks based on region or site code.

Files

- Static files (e.g., banners, scripts, certs) that can be transferred to the agent device using file resource declarations.
- Example:

```
file {
 '/etc/motd':
  source => 'puppet:///modules/cisco/motd.txt',
}
```

Scalability

Depending on the deployment mode:

- Single server: ~4,000 nodes
- Compiled master with load-balanced CA: Up to 20,000+ nodes

Redundancy is achieved via **multi-master clustering**, **load balancers**, and **shared backends** (e.g., PostgreSQL, PuppetDB).

Chef Overview

Chef is an agent-based configuration management and orchestration tool, similar in design and function to Puppet. It operates in a client/server model where the Chef Client (agent) pulls configuration policies from the Chef Server. Chef is primarily written in Ruby and Erlang, with configuration logic (recipes) authored using Ruby DSL.

Unlike Puppet's strictly pull-based model, Chef supports both pull and limited push capabilities (via Knife or Push Jobs).

Chef vs. Puppet: Key Comparisons

FUNCTION	CHEF TERM	PUPPET EQUIVALENT
SERVER	Chef Server	Puppet Master
AGENT	Chef Client	Puppet Agent
CONFIG FILE	Recipe	Manifest
CONFIG PACKAGE	Cookbook	Module
ADMIN INTERFACE	Workstation	Puppet CLI (not Console)

Both tools offer:

- Open-source and enterprise versions
- Code-based configuration stored and versioned
- Declarative, idempotent configuration models
- Support for managing thousands of nodes
- Communication over HTTP using TCP

Chef Deployment Models

1. **Chef Solo** – All components run locally on the same system (no server).
2. **Chef Client/Server** – Standard distributed architecture.
3. **Hosted Chef** – Cloud-hosted Chef Server provided by Progress.
4. **Private Chef** – Fully on-premise Chef Infra Server deployment.

Ansible

Ansible is an agentless, open-source automation tool designed for configuration management, application deployment, and orchestration. It communicates over SSH (or WinRM for Windows) and requires no software agents on managed devices. Ansible is written in Python and defines automation tasks using YAML-based Playbooks.

Architecture & Components:

- **Control Node**: Executes playbooks, manages inventory, and connects to remote nodes.
- **Managed Nodes**: Devices or systems (e.g., routers, switches, servers) accessible via SSH/WinRM.
- **Inventory**: Static or dynamic list of target hosts (can group devices by function or location).
- **Modules**: Executable units (e.g., ios_config, nxos_command, eos_facts) that perform configuration or state checks.
- **Playbooks**: Declarative YAML files defining tasks, handlers, variables, roles, and conditionals.
- **Templates**: Jinja2-based templates for dynamic configuration generation.

Networking Use Cases:

- Configuration deployment and rollback
- Compliance auditing and drift detection
- VLAN, interface, routing protocol automation
- Scheduled backup and restore of device configurations
- Integration with CI/CD pipelines and Git

Supported Platforms:

- Cisco IOS, IOS-XE, NX-OS, ASA, Meraki (via API)
- Juniper, Arista, F5, Palo Alto, and others via respective modules or APIs

Key Advantages:

- **Idempotency**: Ensures repeatable results; avoids redundant changes.
- **Scalability**: Handles thousands of nodes via dynamic inventory and parallel SSH.
- **Extensibility**: Supports custom modules, plugins, and REST APIs.
- **Security**: SSH-based with support for Ansible Vault to encrypt sensitive data.

Comparison of orchestration tools

TOOL	AGENT MODEL	LANGUAGE	PUSH/PULL	NETWORK SUPPORT	NOTES
ANSIBLE	Agentless	YAML/Python	Push	Extensive	Simplest to deploy; widely adopted
CHEF	Agent-based	Ruby	Push/Pull	Limited (via plugins)	Rich DSL; steeper learning curve
PUPPET	Agent-based	DSL (Ruby)	Pull	Limited	Strong state enforcement
SALTSTACK	Optional Agent	YAML/Python	Push/Pull	Moderate	Fast execution via ZeroMQ

Additional Topics

System Log Message Elements Summary

Cisco system logs contain several key elements that provide details about network events:

- **Sequence Number**: Optional; enabled with service sequence-numbers.
- **Timestamp**: Displays event time in different formats (real-time or uptime).
- **Facility**: Identifies the source of the message (e.g., SNMP, SYS).
- **Severity**: A single-digit code (0-7) indicating the message's importance.
- **Mnemonic**: A unique identifier describing the event.
- **Description**: Detailed event information.

Example Log Messages

1. **Interface Status Changes**
    a. %LINK-3-UPDOWN: Interface state change (e.g., Port-channel1 and GigabitEthernet0/1 moved to "up").
    b. %LINEPROTO-5-UPDOWN: Line protocol changes (e.g., Vlan1 and GigabitEthernet0/1 went down).
2. **Configuration Changes**
    a. %SYS-5-CONFIG_I: Configuration modified from vty2 session with a specific IP.

These logs help track network activity, troubleshoot issues, and monitor configuration changes.

BFD & UDLD

BFD and UDLD are protocols used to detect failures at their respective layer.

Configuring BFD and UDLD on an Interface

Enabling UDLD in Normal Mode (Detects issues but does not shut down the port):

```
interface GigabitEthernet0/1
  udld port
```

Logs unidirectional failures but does **not** disable the interface.

Enabling UDLD in Aggressive Mode (Shuts down the port on failure):

```
interface GigabitEthernet0/1
  udld port aggressive
```

BFD is typically used in conjunction with routing protocols like OSPF, BGP, or EIGRP.

Steps to configure BFD on an interface:

```
interface GigabitEthernet0/1
  bfd interval 50 min_rx 50 multiplier 3
```

interval 50 → BFD hello packet interval (in milliseconds).
min_rx 50 → Minimum BFD receive interval (in milliseconds).
multiplier 3 → Multiplier for BFD timeout (i.e., session is considered down after 3 missed packets).

Enabling BFD for OSPF:

```
router ospf 1
  bfd all-interfaces
```

Enables BFD for all OSPF interfaces.

Enabling BFD for BGP Neighbor:

```
router bgp 65001
  neighbor 192.168.1.2 remote-as 65002
  neighbor 192.168.1.2 bfd
```

Enables BFD for fast failure detection on the BGP session.

FEATURE	BFD (BIDIRECTIONAL FORWARDING DETECTION)	UDLD (UNIDIRECTIONAL LINK DETECTION)
PURPOSE	Detects failures in **Layer 3 paths**	Detects **unidirectional fiber link** failures
DETECTION METHOD	Sends **periodic hello packets**	Sends **hello packets to verify one-way faults**
FAILURE ACTION	Triggers **routing protocol reconvergence**	Can **shut down the affected interface**
SUPPORTED PROTOCOLS	OSPF, EIGRP, BGP	Ethernet only
IMPLEMENTATION	Software-based, uses **lightweight timers**	Software-based, used primarily on **fiber links**

ARP

Arp is used to determine the MAC address of devices in the LAN using their IP Address.

The following process illustrates how ARP operates.

ARP Request & Reply Composition

ARP Request to Target

- Host A (e.g., PC1) constructs an ARP request that populates four key ARP fields:
    - **Source Hardware Address (SHA):** Host A's MAC address
    - **Source Protocol Address (SPA):** Host A's IP address
    - **Target Hardware Address (THA):** Set to 00:00:00:00:00:00 (unknown)
    - **Target Protocol Address (TPA):** Host B's (PC2) IP address
- **Layer 2 Encapsulation**
- This ARP message is encapsulated in an Ethernet frame where:
    - **Ethernet Source MAC:** Host A's MAC address
    - **Ethernet Destination MAC:** FF:FF:FF:FF:FF:FF (broadcast)
    - **EtherType:** 0x0806 (indicating ARP)
- **Broadcast and Switch Forwarding**
- The switch, upon receiving the broadcast frame, forwards it out all ports within the same VLAN/broadcast domain.

ARP Reply from Target

- Only the host matching the **TPA** (Host B) processes the ARP Request.
- Host B replies with a unicast ARP Reply containing:
    - **SHA:** Host B's MAC
    - **SPA:** Host B's IP
    - **THA:** Host A's MAC
    - **TPA:** Host A's IP
- The Ethernet header is now addressed specifically back to Host A (unicast).
- **ARP Cache Updates**
- Both Host A and Host B update their ARP caches with the Source Hardware/Protocol Address mappings of the other device.
- This eliminates the need for broadcast ARP queries when sending subsequent IP packets to each other.

Maximum Transmission Unit (MTU)

- **Definition**: The largest size of an IP packet (including IP headers) that can travel across a network path without fragmentation.
- **Default Size**: Often 1500 bytes for Ethernet networks.
- **Configuration**: Typically set on network interfaces (e.g., ip mtu 1500 on Cisco).
- **Min RFC Standard**: 576 bytes (some references state 576, minus 40 bytes for overhead), though Cisco devices commonly reference 567 bytes + 40 for headers.
- **Fragmentation**: Occurs if a packet exceeds the path or interface MTU; routers split packets into multiple fragments (which is often less efficient).
- **Jumbo Frames**: Packets larger than 1500 bytes (e.g., 9000 bytes) used in some high-throughput networks.

Path MTU Discovery (PMTUD)

- **Purpose**: Determines the smallest MTU along a path to avoid fragmentation.
- **Method**: Sends packets with the DF (Don't Fragment) bit set. If a router needs to fragment and can't, it returns an ICMP "Fragmentation Needed" message. This allows hosts to dynamically adjust their packet sizes.

Maximum Segment Size (MSS)

- **Definition**: The largest amount of TCP payload data a host is willing to receive in one segment. Generally, MSS = MTU - IP/TCP header size.
- **Default**: 1460 bytes on a standard Ethernet network (1500 MTU minus 20-byte IP header and 20-byte TCP header).
- **Negotiation**: Exchanged during the TCP three-way handshake. Each host declares its MSS; the **lower** of the two is used.
- **Impact of Overhead**: Additional headers (e.g., GRE, IPsec) reduce the effective MSS to avoid fragmentation. For example, if GRE adds 28 bytes, MSS might be set to 1432.

MSS Clamping / MSS Adjust

- **Use Case**: Allows an intermediate device (router) to modify the MSS option in TCP SYN and SYN-ACK packets (e.g., ip tcp adjust-mss 1432 on a tunnel interface).
- **Why?**: Ensures packets do not exceed the reduced MTU on links with additional overhead (like VPN, GRE) or in networks that require uniform smaller segments.
- **Scope**: Affects transit traffic only (not traffic terminating on the device).

Common Troubleshooting Steps

1. **Ping with DF Bit**:
    a. ping <destination> size <bytes> df-bit
    b. Incrementally increase packet size to find the path's actual MTU.
2. **Capture ICMP Messages**: Look for "Fragmentation Needed" responses or dropped packets.
3. **Check Interface MTU**:
    a. Cisco: show interface <iface>
    b. Linux: ifconfig or ip addr show
    c. Windows: netsh interface ipv4 show interfaces
4. **Review MSS Adjustments**:
    a. Cisco: ip tcp adjust-mss <size> on interface.
5. **Look for Errors/Counters**:
    a. show ip traffic / debug ip icmp / debug ip tcp (in a lab or controlled setting).

Key Takeaways

- **MTU & MSS Relationship**: MSS is always derived from MTU by subtracting overhead (IP + TCP headers and any additional encapsulation).
- **Goal**: Maximize payload size (MSS) without exceeding the path MTU, thereby reducing fragmentation and increasing efficiency.
- **PMTUD & MSS Clamping**: Two complementary strategies for dynamically or manually optimizing segment size.

IP SLA , Tracking and Syslog using EEM

In this setup, we used **IP SLA** not just to passively measure reachability, but to **actively detect service loss** and **notify the network operator** via syslog. This mimics how SLA is used in production environments for link or service health monitoring.

1 - IP SLA Monitor - Probes a remote IP using ICMP with configurable frequency and timeout.

2 - Track Object - Ties SLA reachability to a logical.

3 - EEM Applets (Embedded Event Manager) - EEM applets are small event-driven scripts native to IOS that let the device react to internal events (e.g., syslog messages, track state changes, CLI input).

Configuration –

```
! Step 1: Create the IP SLA operation
ip sla 2
 icmp-echo 192.168.1.1
 timeout 1000
 frequency 5
exit

! Step 2: Schedule the SLA
ip sla schedule 2 life forever start-time now

! Step 3: Track SLA reachability
track 1 ip sla 2 reachability

! Step 4: EEM applet to log when SLA goes DOWN
event manager applet SLA_Down
 event track 1 state down
 action 1 syslog msg "SLA_DOWN: Target unreachable"

! Step 5: EEM applet to log when SLA goes UP
event manager applet SLA_Up
 event track 1 state up
 action 1 syslog msg "SLA_UP: Target reachable"
```

Result of IP going down and up.

```
%TRACK-6-STATE: 1 ip sla 2 reachability Up -> Down
%HA_EM-6-LOG: SLA_DOWN: Target unreachable
%TRACK-6-STATE: 1 ip sla 2 reachability Down -> Up
%HA_EM-6-LOG: SLA_UP: Target reachable
```