Set IPv6 on Router

This is a copy of a conversation between ChatGPT & Anonymous.

how to set ipv6 address in router in cisco packet tracer

To set an IPv6 address on a router in **Cisco Packet Tracer**, follow these steps:

Step 1: Enable IPv6 Routing

1. Open Cisco Packet Tracer and select a **router**.
2. Enter the **CLI (Command Line Interface)**.
3. Enable privileged EXEC mode:

```
shell

Router> enable
```

4. Enter global configuration mode:

```
arduino

Router# configure terminal
```

5. Enable IPv6 routing (if not already enabled):

```
scss

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

Step 2: Assign an IPv6 Address to an Interface

1. Enter the interface configuration mode (e.g., for **GigabitEthernet0/0**):

```
arduino

Router(config)# interface GigabitEthernet0/0
```

2. Assign an IPv6 address:

```
scss

Router(config-if)# ipv6 address 2001:db8:1::1/64
```

   - **2001:db8:1::1** → Example IPv6 address

- **/64** → Subnet prefix

3. Enable the interface:

```
arduino

Router(config-if)# no shutdown
```

4. Exit the interface configuration:

```
arduino

Router(config-if)# exit
```

Step 3: Verify Configuration

1. Check the assigned IPv6 address:

```
kotlin

Router# show ipv6 interface brief
```

2. To check IPv6 routing:

```
arduino

Router# show ipv6 route
```

Step 4: Assign a Link-Local Address (Optional)

If needed, you can manually set a **link-local** IPv6 address:

```
arduino

Router(config-if)# ipv6 address fe80::1 link-local
```

Now your router is configured with an IPv6 address. Let me know if you need further details! 🚀

how to do ipv6 routing

How to Configure IPv6 Routing in Cisco Packet Tracer

IPv6 routing in Cisco routers requires enabling **IPv6 unicast routing** and configuring **routing protocols** like **Static Routing**, **RIPng**, or **OSPFv3**.

🔹 Step 1: Enable IPv6 Routing

1. Open the router CLI.
2. Enter privileged EXEC mode:

```
bash

Router> enable
```

3. Enter global configuration mode:

```
bash

Router# configure terminal
```

4. Enable IPv6 routing:

```
bash

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

🔹 Step 2: Assign IPv6 Addresses to Interfaces

1. Select the interface (e.g., GigabitEthernet0/0):

```
bash

Router(config)# interface GigabitEthernet0/0
```

2. Assign an IPv6 address:

```
bash

Router(config-if)# ipv6 address 2001:db8:1::1/64
```

3. Enable the interface:

```
bash

Router(config-if)# no shutdown
```

4. Repeat for other interfaces as needed.

🔹 Step 3: Configure IPv6 Routing

You can use **static routing** or **dynamic routing protocols**.

📌 Option 1: Static Route

To configure a static route to another network:

```
bash

Router(config)# ipv6 route 2001:db8:2::/64 2001:db8:1::2
```

- **2001:db8:2::/64** → Destination network
- **2001:db8:1::2** → Next-hop IPv6 address

📌 Option 2: RIPng (RIP for IPv6)

1. Enable RIPng:

```
bash

Router(config)# ipv6 router rip MYRIP
```

2. Activate RIPng on interfaces:

```
bash

Router(config-if)# ipv6 rip MYRIP enable
```

📌 Option 3: OSPFv3 (OSPF for IPv6)

1. Enable OSPFv3:

```
bash

Router(config)# ipv6 router ospf 1
```

2. Assign a Router ID (if needed):

```
bash

Router(config-rtr)# router-id 1.1.1.1
```

3. Enable OSPFv3 on interfaces:

```
bash

Router(config)# interface GigabitEthernet0/0
Router(config-if)# ipv6 ospf 1 area 0
```

🔹 Step 4: Verify Configuration

1. Check IPv6 interfaces:

```
bash

Router# show ipv6 interface brief
```

2. Check IPv6 routes:

```
bash

Router# show ipv6 route
```

3. Check IPv6 neighbors:

```
bash

Router# show ipv6 neighbors
```

Now your **IPv6 routing** is set up! 🚀 Let me know if you need further details. 😊

which option is easy for routing

Which IPv6 Routing Option is Easiest?

The **easiest IPv6 routing method** depends on your network size and requirements:

Routing Method	Best For	Complexity
Static Routing	Small networks (few routers)	**Easiest** ✅
RIPng (RIP for IPv6)	Simple dynamic routing	**Easy** ✅
OSPFv3 (OSPF for IPv6)	Large networks	**Complex** ❌

🔹 Best Option for Beginners: Static Routing

- Simple to configure 🚀
- No extra settings needed
- Works well for **small networks**

👉 **Example:**

```
bash

Router(config)# ipv6 route 2001:db8:2::/64 2001:db8:1::2
```

💡 **Best when you have only a few routers** to manage.

🔹 Best Option for Larger Networks: RIPng

- **Automatically updates routes** 🛠️

- Easier than OSPFv3
- Works well for **medium-sized networks**

👉 **Example:**

```
bash

Router(config)# ipv6 router rip MYRIP
Router(config-if)# ipv6 rip MYRIP enable
```

💡 **Best for networks with 5-10 routers.**

🔹 When to Use OSPFv3?

- Best for **large networks** 🌍
- More **control & efficiency**
- **Harder to configure** than RIPng

👉 **Example:**

```
bash

Router(config)# ipv6 router ospf 1
Router(config-if)# ipv6 ospf 1 area 0
```

💡 **Use OSPFv3 only if you have 10+ routers.**

✅ Final Recommendation

✔️ **Use Static Routing** if you have 2-3 routers.
✔️ **Use RIPng** if you have 5-10 routers.
✔️ **Use OSPFv3** only if you have a large network.

For **easy IPv6 routing**, **static routing** is the best choice! 🚀 Let me know if you need help setting it up. 😊

in static i can understand the Destination network and Next-hop IPv6 address

Yes! Let me explain **Destination Network** and **Next-Hop IPv6 Address** in **Static Routing** in a simple way. 🚀

🔹 Destination Network

This is the network you want to **reach** from your router.

- It is defined using an **IPv6 address and prefix** (like a subnet).

👉 **Example:**

```
bash

Router(config)# ipv6 route 2001:db8:2::/64 2001:db8:1::2
```

Here, `2001:db8:2::/64` is the **destination network** (the network you want to send traffic to).

🔹 Next-Hop IPv6 Address

This is the **IPv6 address of the next router** (gateway) that will forward your packet towards the **destination network**.

In the example:

```
bash

Router(config)# ipv6 route 2001:db8:2::/64 2001:db8:1::2
```

- `2001:db8:1::2` is the **next-hop IPv6 address** (the next router your data will go through).

💡 **Think of it as:**

- **Destination Network** = Where you want to go 🌍
- **Next-Hop Address** = Who will take you there 🚦

📌 Real Example in a Network

Network Topology:

```
css

[Router1] ----- [Router2] ----- [Network 2001:db8:2::/64]
```

- **Router1 has an interface:** `2001:db8:1::1/64`
- **Router2 has an interface:** `2001:db8:1::2/64` and is connected to `2001:db8:2::/64`

To make **Router1** reach `2001:db8:2::/64`, we set a static route:

```
bash

Router1(config)# ipv6 route 2001:db8:2::/64 2001:db8:1::2
```

- Now, Router1 knows that **to reach** `2001:db8:2::/64`, it should send traffic to **Router2 (`2001:db8:1::2`)**.

✅ Summary

Term	Meaning	Example
Destination Network	The network you want to reach	`2001:db8:2::/64`
Next-Hop IPv6 Address	The next router that will forward traffic	`2001:db8:1::2`

Let me know if you need further clarification! 😊

i have 3 router how to do ripng

Configuring RIPng on 3 Routers in Cisco Packet Tracer

RIPng (Routing Information Protocol for IPv6) is a **dynamic routing protocol** that helps routers exchange IPv6 routes automatically. Let's configure **RIPng** for a network with **3 routers**.

🔹 Step 1: Network Topology

Imagine you have 3 routers connected like this:

```
css

[Router1] --- [Router2] --- [Router3]
```

Router	Interface	IPv6 Address	Network
R1	Gig0/0	`2001:db8:1::1/64`	`2001:db8:1::/64`
R1	Gig0/1	`2001:db8:2::1/64`	`2001:db8:2::/64`
R2	Gig0/0	`2001:db8:2::2/64`	`2001:db8:2::/64`
R2	Gig0/1	`2001:db8:3::1/64`	`2001:db8:3::/64`
R3	Gig0/0	`2001:db8:3::2/64`	`2001:db8:3::/64`
R3	Gig0/1	`2001:db8:4::1/64`	`2001:db8:4::/64`

🔹 Step 2: Enable IPv6 Routing on All Routers

Run the following commands on **each router**:

```
bash

Router> enable
Router# configure terminal
Router(config)# ipv6 unicast-routing
```

🔹 Step 3: Configure RIPng on Each Router

Now, configure **RIPng** and enable it on the interfaces.

📌 On Router 1 (R1)

```
bash

Router1(config)# ipv6 router rip MYRIP
Router1(config)# interface GigabitEthernet0/0
Router1(config-if)# ipv6 rip MYRIP enable
Router1(config-if)# exit
Router1(config)# interface GigabitEthernet0/1
Router1(config-if)# ipv6 rip MYRIP enable
Router1(config-if)# exit
```

📌 On Router 2 (R2)

```
bash

Router2(config)# ipv6 router rip MYRIP
Router2(config)# interface GigabitEthernet0/0
Router2(config-if)# ipv6 rip MYRIP enable
Router2(config-if)# exit
Router2(config)# interface GigabitEthernet0/1
Router2(config-if)# ipv6 rip MYRIP enable
Router2(config-if)# exit
```

📌 On Router 3 (R3)

```
bash

Router3(config)# ipv6 router rip MYRIP
Router3(config)# interface GigabitEthernet0/0
Router3(config-if)# ipv6 rip MYRIP enable
Router3(config-if)# exit
Router3(config)# interface GigabitEthernet0/1
Router3(config-if)# ipv6 rip MYRIP enable
Router3(config-if)# exit
```

🔹 Step 4: Verify Configuration

After configuring all routers, check if they have learned routes from each other.

Check RIPng Neighbors

```
bash

Router# show ipv6 rip neighbor
```

Check the Routing Table

```
bash

Router# show ipv6 route
```

You should see RIP-learned routes in the table, marked as **"R"**.

✅ Summary

1. **Enable IPv6 Routing** on all routers.
2. **Create a RIPng process** (`ipv6 router rip MYRIP`).
3. **Enable RIPng on interfaces** (`ipv6 rip MYRIP enable`).
4. **Verify connectivity** using `show ipv6 route`.

Now, all 3 routers will **dynamically share IPv6 routes** with each other using **RIPng**! 🚀

Let me know if you have any doubts! 😊

so i have to do ripng in all up ports

Yes! You Need to Enable RIPng on All Up (Active) Ports 🚀

In **RIPng**, each router **advertises** and **learns** routes through its **active interfaces**.

So, **you must enable RIPng on all interfaces that are connected to another router**.

🔹 Why Enable RIPng on All Up Ports?

1. **Allows route exchange** between connected routers.
2. **Ensures automatic updates** for network changes.
3. **Prevents routing issues** (if RIPng is not enabled, that interface won't share routes).

🔹 Example: 3 Routers Connected

```
css

[Router1] --- [Router2] --- [Router3]
```

Router	Interface	IPv6 Address	Enable RIPng?
R1	Gig0/0	`2001:db8:1::1/64`	✅ YES
R1	Gig0/1	`2001:db8:2::1/64`	✅ YES
R2	Gig0/0	`2001:db8:2::2/64`	✅ YES
R2	Gig0/1	`2001:db8:3::1/64`	✅ YES
R3	Gig0/0	`2001:db8:3::2/64`	✅ YES
R3	Gig0/1	`2001:db8:4::1/64`	✅ YES

So, on **each router**, you must configure **RIPng on all active interfaces**:

```
bash

Router(config)# interface GigabitEthernet0/X
Router(config-if)# ipv6 rip MYRIP enable
```

🔹 If a Port is Down?

- If a port **is not connected** or **unused**, you can **skip enabling RIPng** on that port.
- Use `` `show ipv6 interface brief` `` to check active ports.

✅ Final Answer:

✔️ **Yes, enable RIPng on all up (active) ports** to ensure routing works properly. 🚀

Let me know if you need more help! 😊

Configure, apply and verify an ACL that will block HTTP and HTTPS access on Router 3

Configure, Apply, and Verify an ACL to Block HTTP and HTTPS on Router 3

We need to create an **IPv4 ACL** that blocks **HTTP (port 80)** and **HTTPS (port 443)** on **Router 3 (R3)**, apply it to the correct interface, and verify its effect.

🔹 Step 1: Configure the ACL

1. Access **R3 CLI**.
2. Create an **Extended ACL** (blocks traffic based on ports):

```
bash


Router3> enable
Router3# configure terminal
Router3(config)# access-list 110 deny tcp any any eq 80
Router3(config)# access-list 110 deny tcp any any eq 443
Router3(config)# access-list 110 permit ip any any
```

✅ **Explanation:**

- `` `deny tcp any any eq 80` `` → Blocks HTTP
- `` `deny tcp any any eq 443` `` → Blocks HTTPS
- `` `permit ip any any` `` → Allows all other traffic (to prevent blocking everything)

🔹 Step 2: Apply ACL to an Interface

Now, apply the ACL to **R3's inbound or outbound interface**.

- If **blocking incoming traffic**, apply ACL **inbound** on the **LAN-facing interface**.
- If **blocking outgoing traffic**, apply ACL **outbound** on the **WAN-facing interface**.

Example: Apply on GigabitEthernet0/0 (Inbound)

```
bash

Router3(config)# interface GigabitEthernet0/0
Router3(config-if)# ip access-group 110 in
Router3(config-if)# exit
```

🔹 Step 3: Verify ACL Configuration

1️⃣ Check if ACL is Applied

```
bash

Router3# show ip interface GigabitEthernet0/0
```

✅ Look for:

`Inbound access list is 110`

2️⃣ Check ACL Rules

```
bash

Router3# show access-lists
```

✅ Look for:

```
pgsql

Extended IP access list 110
    deny tcp any any eq www
    deny tcp any any eq 443
    permit ip any any
```

3️⃣ Test with a PC

1. **Try accessing a website** from a PC connected to R3.
2. If HTTP/HTTPS is blocked, the webpage should **not load**.
3. **Try pinging a website (e.g., `ping 8.8.8.8`)** → Ping should work since we only blocked HTTP/HTTPS, not ICMP.

✅ Summary

✔️ **Step 1:** Create ACL **(Blocks HTTP & HTTPS, Allows Other Traffic)**
✔️ **Step 2:** Apply ACL **(Inbound or Outbound on Interface)**
✔️ **Step 3:** Verify **(Check ACL Rules & Test Connection)**

Now, HTTP & HTTPS traffic is blocked on Router 3! 🚀

Let me know if you need modifications! 😊

tell for outbound

Configure, Apply, and Verify an ACL to Block HTTP & HTTPS (Outbound) on Router 3

In this case, we will **block HTTP (port 80) and HTTPS (port 443) traffic going out of Router 3**. This means users behind **R3 can send requests**, but the responses from web servers will be blocked.

🔹 Step 1: Configure the ACL

1. Open **R3 CLI**.
2. Create an **Extended ACL** to block HTTP & HTTPS **outbound**:

```
bash

Router3> enable
Router3# configure terminal
Router3(config)# access-list 110 deny tcp any any eq 80
Router3(config)# access-list 110 deny tcp any any eq 443
Router3(config)# access-list 110 permit ip any any
```

✅ **Explanation:**

- `deny tcp any any eq 80` → Blocks **HTTP** traffic
- `deny tcp any any eq 443` → Blocks **HTTPS** traffic
- `permit ip any any` → Allows all other traffic

🔹 Step 2: Apply ACL to an Outbound Interface

- Apply it on the **WAN-facing interface (GigabitEthernet0/1)** to **block outgoing HTTP/HTTPS requests**.

```
bash

Router3(config)# interface GigabitEthernet0/1
Router3(config-if)# ip access-group 110 out
Router3(config-if)# exit
```

🔹 Step 3: Verify ACL Configuration

1️⃣ Check if ACL is Applied

```
bash

Router3# show ip interface GigabitEthernet0/1
```

✅ Look for:

```
pgsql

Outbound access list is 110
```

2️⃣ Check ACL Rules

```
bash

Router3# show access-lists
```

✅ Output should include:

```
pgsql

Extended IP access list 110
    deny tcp any any eq www
    deny tcp any any eq 443
    permit ip any any
```

3️⃣ Test the ACL

1. **Try browsing a website** from a PC behind R3.
2. The request will be sent, but the response won't return → Page **won't load**.
3. **Try Pinging `8.8.8.8`** → Ping **should work** (ICMP is not blocked).

✅ Summary

✔️ **Step 1:** Create ACL (Block HTTP & HTTPS **Outbound**)
✔️ **Step 2:** Apply ACL **to outbound interface** (WAN-facing)
✔️ **Step 3:** Verify with **`show ip interface`** and **test browsing**

Now, all **outgoing HTTP & HTTPS traffic is blocked** from R3! 🚀

Let me know if you need any modifications. 😊

↪

* If **blocking incoming traffic**, apply ACL **inbound** on the **LAN-facing interface**. * If **blocking outgoing traffic**, apply ACL **outbound** on the **WAN-facing interface**.
explain

Understanding Inbound vs. Outbound ACLs in Cisco Routers

An **Access Control List (ACL)** can be applied **inbound** or **outbound** on a router interface. The choice depends on whether you want to block traffic **before it enters** the router or **before it leaves** the router.

🔹 Inbound ACL vs. Outbound ACL

ACL Type	Where It's Applied	Traffic Affected	Example Scenario
Inbound ACL	Applied on the **interface receiving traffic**	Blocks packets **before they enter** the router	Block users from accessing a website
Outbound ACL	Applied on the **interface sending traffic**	Blocks packets **before they leave** the router	Prevent the router from forwarding certain traffic

---

🔹 Explanation of Inbound ACL (LAN-facing interface)

When to Use Inbound ACL?

- **Blocks traffic before it enters the router**.
- Prevents **unwanted traffic** from **ever being processed**.
- **Efficient**, as blocked packets do not consume router resources.

Example: Blocking HTTP/HTTPS Inbound on Router 3

```
css

[User PC] -----> [Router 3] -----> [Internet]
```

1. The user **tries to access a website** (sends an HTTP/HTTPS request).
2. The **inbound ACL on Router 3's LAN-facing interface** blocks it **before reaching the router**.
3. The request **never goes out to the internet**.

✅ **Command:**

```
bash

Router3(config)# interface GigabitEthernet0/0  # LAN-facing interface
Router3(config-if)# ip access-group 110 in
```

💡 **Use Case:** Best when you want to stop users behind R3 from **sending** traffic.

🔹 Explanation of Outbound ACL (WAN-facing interface)

When to Use Outbound ACL?

- Blocks traffic **before it leaves** the router.
- Ensures the router does not **forward unwanted traffic**.
- Useful when you want to **filter traffic after processing**.

Example: Blocking HTTP/HTTPS Outbound on Router 3

```
css

[User PC] -----> [Router 3] -----> [Internet]
```

1. The user **sends an HTTP/HTTPS request** to a website.
2. Router 3 **receives the request** but **blocks it before forwarding** it out.
3. The request **never reaches the internet**.

✅ **Command:**

```
bash

Router3(config)# interface GigabitEthernet0/1  # WAN-facing interface
Router3(config-if)# ip access-group 110 out
```

💡 **Use Case:** Best when you want to **control traffic leaving the router**.

🔹 Key Differences

Feature	Inbound ACL	Outbound ACL
Blocks traffic...	Before entering the router	Before leaving the router
Processing efficiency	More efficient	Less efficient
Use Case	Restrict what enters the router	Restrict what the router sends

---

✅ Final Answer

- Use **Inbound ACL** if you **want to stop traffic before it reaches the router**.
- Use **Outbound ACL** if you **want to stop traffic before it leaves the router**.

For **blocking HTTP & HTTPS on Router 3**, you can **apply ACL outbound** on its **WAN interface (Gig0/1)** to **prevent users from accessing the internet**.

🚀 **Let me know if you need further clarification!** 😊

↪

Outbound access list is 110
this aint coming

If `Outbound access list is 110` is **not appearing** when running `show ip interface GigabitEthernet0/1`, follow these steps to troubleshoot:

🔹 Step 1: Verify ACL Configuration

Check if the ACL is properly created using:

```
bash

Router3# show access-lists
```

✅ You should see output like:

```
pgsql

Extended IP access list 110
    deny tcp any any eq www
    deny tcp any any eq 443
    permit ip any any
```

🚨 **If the ACL is missing**, reconfigure it:

```
bash

Router3(config)# access-list 110 deny tcp any any eq 80
Router3(config)# access-list 110 deny tcp any any eq 443
Router3(config)# access-list 110 permit ip any any
```

🔹 Step 2: Check Interface Name

Make sure you're applying ACL to the **correct interface**. List interfaces using:

```
bash

Router3# show ip interface brief
```

Look for the **active (up/up) interface** connected to the WAN.

If your WAN interface is `**GigabitEthernet0/1**`, apply ACL to it:

```
bash

Router3(config)# interface GigabitEthernet0/1
Router3(config-if)# ip access-group 110 out
Router3(config-if)# exit
```

🔹 Step 3: Verify If ACL is Applied

Now check if the ACL is attached to the interface:

```
bash

Router3# show ip interface GigabitEthernet0/1
```

✅ **Expected Output:**

```
pgsql

Outbound access list is 110
```

🚨 **If not showing**, reapply:

```
bash

Router3(config)# interface GigabitEthernet0/1
Router3(config-if)# no ip access-group 110 out
Router3(config-if)# ip access-group 110 out
```

🔹 Step 4: Test the ACL

1. **Try accessing a website (HTTP/HTTPS) from a PC** behind Router 3.
2. **If the page loads**, ACL is **not applied correctly**.
3. **Try pinging a website (e.g., `ping 8.8.8.8`)** → It should work since only TCP 80/443 is blocked.

🔹 Step 5: Save Configuration

After verifying the ACL works, save the configuration:

```
bash

Router3# write memory
```

✅ Final Solution

✔️ **Check if ACL 110 exists (`show access-lists`)**
✔️ **Check the correct interface (`show ip interface brief`)**
✔️ **Reapply ACL if missing (`ip access-group 110 out`)**
✔️ **Test if HTTP/HTTPS is blocked**

If it's still not working, let me know what output you are getting! 😊

bruh i have routers with ipv6 give the appropriate command

Got it! Since you're working with **IPv6**, you need to configure an **IPv6 ACL** instead of an IPv4 ACL. Here's how you can block **HTTP (port 80) and HTTPS (port 443) outbound** on **Router 3**.

🔹 Step 1: Configure the IPv6 ACL

1. **Go to Router 3 CLI**
2. **Create an IPv6 ACL to block HTTP & HTTPS outbound**

```
bash

Router3> enable
Router3# configure terminal
Router3(config)# ipv6 access-list BLOCK_HTTP_HTTPS
Router3(config-ipv6-acl)# deny tcp any any eq 80
```

```
Router3(config-ipv6-acl)# deny tcp any any eq 443
Router3(config-ipv6-acl)# permit ipv6 any any
Router3(config-ipv6-acl)# exit
```

✅ **Explanation:**

- `deny tcp any any eq 80` → Blocks HTTP
- `deny tcp any any eq 443` → Blocks HTTPS
- `permit ipv6 any any` → Allows all other traffic

🔹 Step 2: Apply the ACL to the Outbound Interface

Now, apply the **IPv6 ACL** to the **WAN-facing interface** (e.g., **GigabitEthernet0/1**):

```
bash

Router3(config)# interface GigabitEthernet0/1
Router3(config-if)# ipv6 traffic-filter BLOCK_HTTP_HTTPS out
Router3(config-if)# exit
```

✅ Why Apply It to Outbound?

- This **blocks all HTTP/HTTPS requests leaving Router 3**.
- Any device behind Router 3 **won't be able to browse websites**.

🔹 Step 3: Verify ACL Configuration

1️⃣ Check if ACL is Applied

```
bash

Router3# show ipv6 interface GigabitEthernet0/1
```

✅ Look for:

```
pgsql

Outbound access list is BLOCK_HTTP_HTTPS
```

2️⃣ Check ACL Rules

```
bash

Router3# show ipv6 access-list
```

✅ You should see:

```
pgsql

IPv6 access list BLOCK_HTTP_HTTPS
    deny tcp any any eq www
```

```
deny tcp any any eq 443
permit ipv6 any any
```

3️⃣ Test the ACL

1. **Try opening a website (HTTP/HTTPS)** from a PC behind R3.
2. The request **should be blocked** (page won't load).
3. **Try pinging `2001:4860:4860::8888` (Google DNS)**

```
bash

ping 2001:4860:4860::8888
```

✅ Ping should work since **only TCP (80/443) is blocked**, not ICMP.

🔹 Step 4: Save the Configuration

Once tested, save the ACL so it persists after a reboot:

```
bash

Router3# write memory
```

✅ Summary

✔️ **Step 1:** Create an **IPv6 ACL** (Block HTTP & HTTPS).
✔️ **Step 2:** Apply ACL to the **outbound interface**.
✔️ **Step 3:** Verify & Test (Check ACL & Try Accessing a Website).
✔️ **Step 4:** Save Configuration.

Now, HTTP & HTTPS **traffic is blocked outbound on Router 3!** 🚀

Let me know if it's still not working! 😊

how to block or permit ftp, icmp, http, https, dns, smtp for ipv4 and ipv6

How to Block or Permit FTP, ICMP, HTTP, HTTPS, DNS, SMTP for IPv4 & IPv6 in Cisco Routers

We will configure **Access Control Lists (ACLs)** to **permit or block** the following protocols for both **IPv4** and **IPv6**:

Protocol	Purpose	Port / Protocol
FTP	File Transfer	TCP **21**
ICMP	Ping & Network Diagnostics	ICMP (**No port, uses protocol 1**)
HTTP	Web Traffic	TCP **80**

Protocol	Purpose	Port / Protocol
HTTPS	Secure Web Traffic	TCP **443**
DNS	Domain Name System	UDP/TCP **53**
SMTP	Email Sending	TCP **25**

IPv4 ACL Configuration

1. **Create an Extended IPv4 ACL**

```
bash

Router(config)# access-list 110 deny tcp any any eq 21     # Block FTP
Router(config)# access-list 110 deny icmp any any          # Block ICMP
Router(config)# access-list 110 deny tcp any any eq 80     # Block HTTP
Router(config)# access-list 110 deny tcp any any eq 443    # Block HTTPS
Router(config)# access-list 110 deny udp any any eq 53     # Block DNS (UDP)
Router(config)# access-list 110 deny tcp any any eq 53     # Block DNS (TCP)
Router(config)# access-list 110 deny tcp any any eq 25     # Block SMTP
Router(config)# access-list 110 permit ip any any          # Allow everything else
```

2. **Apply the ACL to an Interface (Inbound or Outbound)**
   - **Inbound (blocks traffic coming into the router)**

```
bash

Router(config)# interface GigabitEthernet0/0
Router(config-if)# ip access-group 110 in
Router(config-if)# exit
```

   - **Outbound (blocks traffic leaving the router)**

```
bash

Router(config)# interface GigabitEthernet0/1
Router(config-if)# ip access-group 110 out
Router(config-if)# exit
```

3. **Verify IPv4 ACL Configuration**

```
bash

Router# show access-lists
```

IPv6 ACL Configuration

1. **Create an IPv6 ACL**

```
bash

Router(config)# ipv6 access-list BLOCK_SERVICES
Router(config-ipv6-acl)# deny tcp any any eq 21     # Block FTP
Router(config-ipv6-acl)# deny icmp any any          # Block ICMP
Router(config-ipv6-acl)# deny tcp any any eq 80     # Block HTTP
```

```
Router(config-ipv6-acl)# deny tcp any any eq 443    # Block HTTPS
Router(config-ipv6-acl)# deny udp any any eq 53     # Block DNS (UDP)
Router(config-ipv6-acl)# deny tcp any any eq 53     # Block DNS (TCP)
Router(config-ipv6-acl)# deny tcp any any eq 25     # Block SMTP
Router(config-ipv6-acl)# permit ipv6 any any        # Allow everything else
Router(config-ipv6-acl)# exit
```

2. **Apply the ACL to an Interface (Inbound or Outbound)**
   - **Inbound (blocks traffic coming into the router)**

```
bash

Router(config)# interface GigabitEthernet0/0
Router(config-if)# ipv6 traffic-filter BLOCK_SERVICES in
Router(config-if)# exit
```

   - **Outbound (blocks traffic leaving the router)**

```
bash

Router(config)# interface GigabitEthernet0/1
Router(config-if)# ipv6 traffic-filter BLOCK_SERVICES out
Router(config-if)# exit
```

3. **Verify IPv6 ACL Configuration**

```
bash

Router# show ipv6 access-list
```

✅ Summary

✔️ **For IPv4**, use `**access-list 110**` and apply it with `**ip access-group**` on an interface.
✔️ **For IPv6**, use `**ipv6 access-list BLOCK_SERVICES**` and apply it with `**ipv6 traffic-filter**`.
✔️ **Use "deny" to block traffic** and **"permit" to allow traffic**.
✔️ **Apply ACLs to inbound or outbound interfaces based on where traffic needs to be blocked.**

Now, your router will **block or allow** FTP, ICMP, HTTP, HTTPS, DNS, and SMTP based on your rules. 🚀 Let me know if you need modifications! 😊