Device IPs

- Default device IPs
- Create an IP subnet
- Assign device IPs
  - Prerequisites
  - Create an IP profile
  - Selectors
    - User email
    - User group emails
    - User group IDs
    - User group names
    - User name
    - SAML attributes
  - Comparison operators
  - Value
  - Logical operators
  - Order of precedence
- Verify device IPs
  - Via the dashboard
  - Via the CLI
- View subnet usage
- Edit an IP subnet

import { Render, Details } from "~/components";

Client modes	Zero Trust plans

• Traffic and DNS mode • Traffic only mode	All plans

System	Availability	Minimum client version
Windows	✅	2025.10.186.0
macOS	✅	2025.10.186.0
Linux	✅	2025.10.186.0
iOS	✅	
Android	✅	
ChromeOS	✅	

A device IP identifies and routes traffic to a specific device in your Zero Trust organization. When a user registers the Cloudflare One Client (formerly WARP), Cloudflare assigns a virtual IPv4 and IPv6 address to the device registration. The Cloudflare One Client uses these IP addresses to create a virtual network interface on the device, which allows your private network to reach the device via Cloudflare Mesh or Cloudflare WAN on-ramps.

You can verify device IPs and, if needed, reconfigure address pools to avoid overlapping IPs with existing internal resources.

Default device IPs

By default, Cloudflare assigns device IPs from the following address space:

- Default IPv4: `100.96.0.0/12`
- Default IPv6: `2606:4700:0cf1:1000::/64`

If your organization already uses the default IPv4 range for internal networking, or if you require more granular IP assignments for firewall policy management, you can configure custom device IPv4 subnets. You can assign different IPv4 subnets to devices based on the user's identity.

The default IPv6 range is owned by Cloudflare and therefore should not conflict with services on your private network. The device IPv6 range is not configurable.

Create an IP subnet

:::note If your account uses Cloudflare WAN, custom device IP subnets require Unified Routing. If your account uses Legacy Routing, follow How to upgrade to Unified Routing. :::

Create a custom IP subnet when the default IPv4 range conflicts with services on your private network.

To define a custom IPv4 subnet for device IPs:

1. In the Cloudflare dashboard, go to **Zero Trust** > **Team & Resources** > **Devices** > **Device profiles**.
2. Under **Device IP subnets**, select **Add new IP subnet**.
3. Enter any name for the subnet.
4. In **CIDR**, enter a valid IPv4 CIDR block from the supported private ranges:
   - `10.0.0.0/8`
   - `172.16.0.0/12`
   - `192.168.0.0/16`
   - `100.64.0.0/10`

   The configured CIDR block must be at least size `/24`.
   :::note[Avoid IP conflicts] Ensure that the chosen CIDR block does not overlap with existing internal routes, such as local office subnets or virtual private clouds (VPCs) in AWS or GCP. :::
5. Select **Add subnet** to save.

Next, assign this subnet to a group of devices.

Assign device IPs

Assign custom IP subnets to ensure devices are provisioned within a predictable address space based on specific user identity criteria.

Prerequisites

- **Assign a unique IP address to each device** is enabled in your general device profiles.

Create an IP profile

To assign IP subnets to your devices:

1. In the Cloudflare dashboard, go to **Zero Trust** > **Team & Resources** > **Devices** > **Device profiles**.
2. Under **Device IP profiles**, select **Add new IP profile**.
3. Enter a name for this group of devices (for example, `IT department`).
4. Create rules to define the users or devices that will receive these IPs. Learn more about the available Selectors, Operators, and Values.
5. Choose an existing IPv4 subnet from the dropdown menu, or create a new subnet.
6. Select **Assign IP address**.
7. (Optional) In the **Device IP profiles** table, change the order of precedence of IP profiles.

Devices that match your rules are assigned a random IP from this address space upon registration. Only newly registered devices will receive a new IP; existing devices will not see any impact to connectivity. To assign a new IP to an existing device, you must delete its registration and then re-enroll the device in your Zero Trust organization.

Organizations are currently limited to 30 custom device IP profiles per account.

Selectors

You can configure IP profiles to match against the following selectors or criteria. Identity-based selectors are only available if the user enrolled the device by logging in to an identity provider (IdP).

User email

Apply a device profile based on the user's email.

<Render file="gateway/selectors/user-email" product="cloudflare-one" params={{ UIname: "User email" }} />

User group emails

Apply a device IP profile based on an IdP group email address of which the user is configured as a member in the IdP.

<Render file="gateway/selectors/user-group-email" product="cloudflare-one" params={{ UIname: "User group emails" }} />

User group IDs

Apply a device IP profile based on an IdP group ID of which the user is configured as a member in the IdP.

<Render file="gateway/selectors/user-group-ids" product="cloudflare-one" params={{ UIname: "User group IDs" }} />

User group names

Apply a device IP profile based on an IdP group name of which the user is configured as a member in the IdP.

<Render file="gateway/selectors/user-group-names" product="cloudflare-one" params={{ UIname: "User group names" }} />

User name

<Render file="gateway/selectors/user-name" product="cloudflare-one" params={{ UIname: "User Name" }} />

SAML attributes

Apply a device IP profile based on an attribute name and value from a SAML IdP.

Comparison operators

Comparison operators determine how device IP profiles match a selector.

Operator	Meaning
in	matches at least one of the defined values
not in	does not match any of the defined values
is	equals the defined value
matches	regular expression (regex) evaluates to true

Value

<Render file="gateway/value" product="cloudflare-one" params={{ selector: "User email", selectordescription: "user's email domain", operator: "matches", valueOne: "@acme.com", valueTwo: "@widgets.com", }} />

Logical operators

To evaluate multiple conditions in an expression, select a logical operator:

Operator	Meaning
And	match all of the conditions in the expression
Or	match any of the conditions in the expression

Order of precedence

The Cloudflare One Client checks the IP profiles from top to bottom as they appear in the Cloudflare One dashboard (lowest precedence number is checked first). The client follows the first match principle — once a device matches an IP profile, the client stops evaluating and no subsequent IP profiles can override the decision. You can rearrange the IP profiles in the Cloudflare One dashboard according to your desired order of precedence.

Verify device IPs

Via the dashboard

To check the virtual IP addresses assigned to a specific device registration:

1. In the Cloudflare dashboard, go to **Zero Trust** > **Team & Resources** > **Devices**.
2. Select your device > **View details**. :::tip[Device filters] The Cloudflare One dashboard defaults to showing devices that were last seen within the past year. You can select **Show filters** to change the date range or filter by the last active user. :::
3. Scroll down to **Users**. You will see the registrations associated with this device along with their assigned IPv4 and IPv6 addresses.

Via the CLI

To check the device IP used by the device client's virtual network interface:

View subnet usage

Monitor the consumption of your IPv4 subnets to ensure you have enough addresses for new device registrations. Devices will be unable to register if they match a subnet with no available IPs.

Use the Cloudflare One dashboard to view a high-level overview of assigned and available IPs:

1. In the Cloudflare dashboard, go to **Zero Trust** > **Team & Resources** > **Devices** > **Device profiles**.
2. Locate the **Device IP subnets** table.
3. The **IPs assigned** column displays the total number of IPs currently assigned to active device registrations versus the total capacity of the CIDR block.

If your subnet is approaching capacity, you can expand your subnet to increase the number of available IPs. Alternatively, you can free up IPs by deleting existing device registrations, particularly revoked registrations that may be consuming IP space despite the device no longer being in use.

:::tip[Delete device registrations instead of revoking] Revoking a device registration does not release the virtual IPs that are assigned to the registration. Because virtual IPs are a finite resource, Cloudflare strongly advises deleting a registration rather than revoking it. :::

To get a list of all device registrations in a subnet (including revoked registrations), use the Cloudflare API. For example, the following script fetches all device registrations and their device IPs, and outputs all registrations within the specified CIDR block.

1. Create a new file called `filter-device-ips.py` that contains the following code: ```python import requests import ipaddress import json

```
# --- Configuration ---
AUTH_EMAIL = "<CLOUDFLARE_ACCOUNT_EMAIL>"
AUTH_KEY = "<CLOUDFLARE_API_KEY>"  # Refer to
https://developers.cloudflare.com/fundamentals/api/get-started/keys/. API token
authentication is not currently supported for fetching device IPs.
ACCOUNT_ID = "<CLOUDLFARE_ACCOUNT_ID"  # Refer to
https://developers.cloudflare.com/fundamentals/account/find-account-and-zone-ids/.
TARGET_CIDR = "100.64.0.0/10"

# --- API request headers ---
headers = {
□□"X-Auth-Email": AUTH_EMAIL,
□□"X-Auth-Key": AUTH_KEY,
```

```
    "Content-Type": "application/json"
}

def get_all_registrations():
    """Fetches all device registrations including revoked registrations. """
    devices = {}
    url = f"https://api.cloudflare.com/client/v4/accounts/{ACCOUNT_ID}/devices/registrations"
    params = {"per_page": 50, "status": "all"}

    while True:
        response = requests.get(url, headers=headers, params=params).json()
        if not response.get('success'):
            print(f"Error fetching registrations: {response.get('errors')}")
            break

        for d in response.get('result', []):
            # We use the ID as the key to link with IP data later
            devices[d['id']] = d

        cursor = response.get('result_info', {}).get('cursor')
        if not cursor:
            break
        params['cursor'] = cursor

    return devices

def filter_by_cidr(device_map, network):
    """Fetch device IPs and return devices that fall within the target CIDR block."""
    matches = []
    device_ids = list(device_map.keys())

    # API limits IP correlation to batches of 20
    for i in range(0, len(device_ids), 20):
        batch = device_ids[i:i+20]
        # Construct parameters for the IP endpoint
        params = {f"device_ids[{idx}]": d_id for idx, d_id in enumerate(batch)}
        url = f"https://api.cloudflare.com/client/v4/accounts/{ACCOUNT_ID}/teamnet/devices/ips"
```

```
        res = requests.get(url, headers=headers, params=params).json()

        if not res.get('success'):
            print(f"Error fetching IPs: {res.get('errors')}")
            continue

        for item in res.get('result', []):
            d_id = item.get('device_id')
            ip_data = item.get('device_ips', {})
            ipv4_str = ip_data.get('ipv4')

            if ipv4_str:
                try:
                    if ipaddress.IPv4Address(ipv4_str) in network:
                        if d_id in device_map:
                            full_data = device_map[d_id]
                            full_data['device_ips'] = ip_data
                            matches.append(full_data)
                except ValueError:
                    continue

    return matches

if __name__ == "__main__":
    try:
        net = ipaddress.IPv4Network(TARGET_CIDR, strict=False)

        print(f"[*] Fetching registrations (status=all)...")
        all_devices = get_all_registrations()
        print(f"[*] Found {len(all_devices)} total registrations.")

        print(f"[*] Checking IP ranges for match...")
        filtered_list = filter_by_cidr(all_devices, net)

        if filtered_list:
            print(f"\n--- Found {len(filtered_list)} Device(s) in {TARGET_CIDR} ---\n")
            for dev in filtered_list:
                print(json.dumps(dev, indent=2))
```

```
□□□□□□□□print("-" * 50)
□□□□else:
□□□□□□print(f"\nNo devices found within the {TARGET_CIDR} range.")

□□except Exception as e:
□□□□print(f"Script Error: {e}")
```

2. In the script configuration section, input your Cloudflare API credentials and your IP subnet range.
3. Open a terminal and navigate to the script directory. To run the script, type:

   ```
   python3 filter-device-ips.py
   ```

Edit an IP subnet

Cloudflare does not support editing an existing IPv4 subnet definition. To assign a different IPv4 subnet to your devices:

1. In the Cloudflare dashboard, go to **Zero Trust** > **Team & Resources** > **Devices** > **Device profiles**.
2. Under **Device IP profiles**, find the device group associated with the old subnet and select **Edit**.
3. Select **Create new subnet IP range** to define a new subnet.
4. Select **Save**.

The new subnet will appear in the **Device IP subnets** table. You can now delete the old subnet. Devices will only get an IP from the new subnet when they re-register; existing registrations will retain their current IP.

□□□ #2
□ □□□ □□□ 2026-11-01 01:16:33 UTC
□ □□□ □□□ 2026-11-01 01:16:35 UTC