Tencent Cloud EdgeOne

L4 Proxy

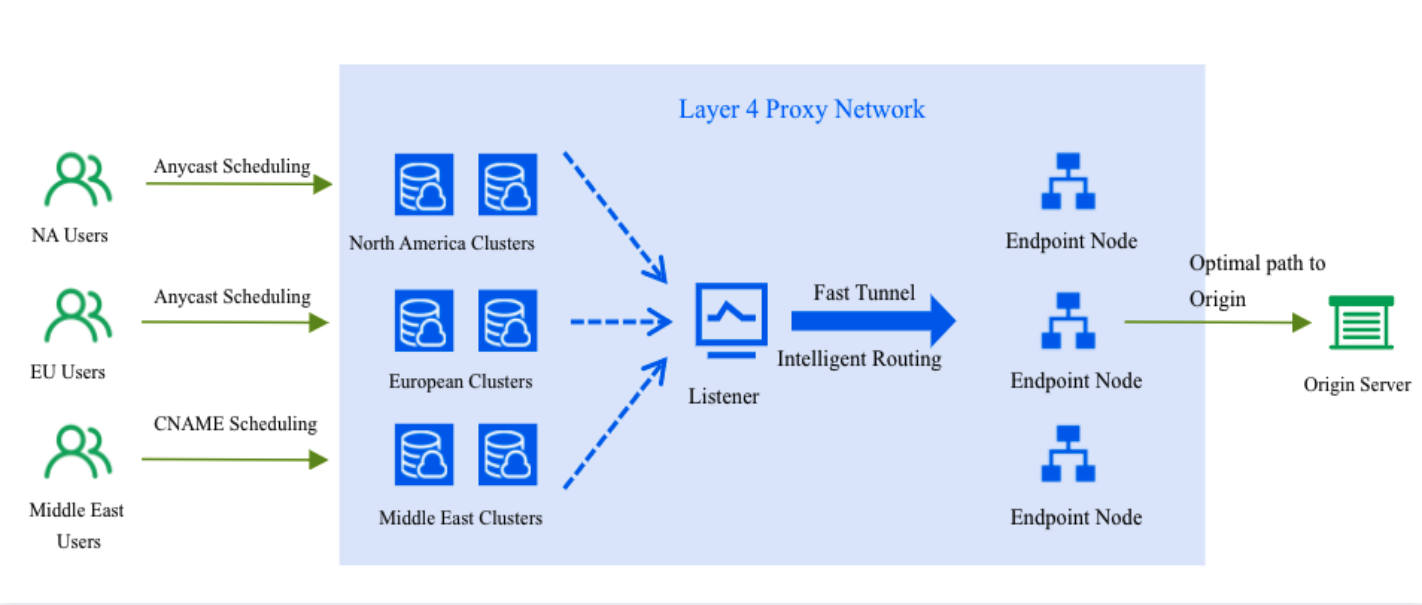
Contents

L4 Proxy Overview

Last updated: 2023-09-07 18:32:23

How It Works

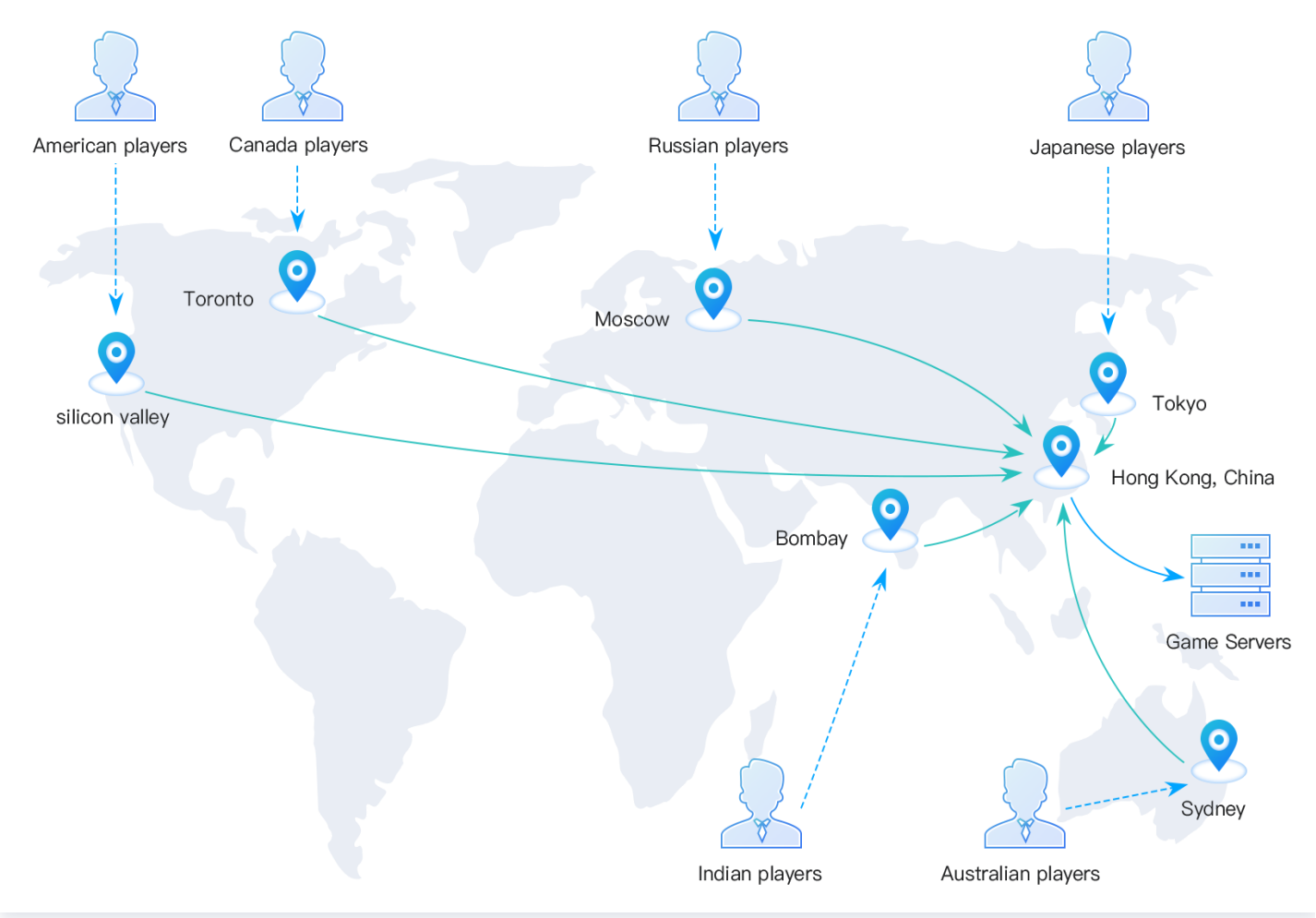
L4 proxy is the acceleration service of EdgeOne based on TCP/UDP. By leveraging widely distributed layer-4 proxy nodes, unique DDoS module, and smart routing technology, EdgeOne implements nearby access for end users, edge traffic cleansing, and port monitoring and forwarding. It thus offers high-availability and low-latency DDoS mitigation and acceleration services for layer-4 applications.



Scenarios

Game Acceleration

L4 proxy accelerates data transmission over TCP/UDP for mobile and PC games, such as real-time battle games and MMORPGs that require global access to a unified server. L4 proxy connects players to the nearest high-speed channels to reduce the packet loss rate and latency of the game due to varying network conditions across regions.



OA Application Acceleration

Generally, in cross-regional office scenarios, the business data of a company is stored in the master data center at its headquarters. This often results in a high packet loss rate with high latency during cross-regional communication due to network issues, causing troubles in cross-regional business access and data synchronization. L4 proxy effectively solves those network issues and improves the business access experience by connecting users to the nearest EdgeOne nodes and optimizing the access links.

Real-time Audio/Video

L4 proxy supports forwarding acceleration over UDP. This ensures reliable audio and video transmission in real-time interactive scenarios, such as video meetings and video communication between anchors and audience members. L4 proxy solves network issues such as audio/video lags, packet loss, and high latency during cross-ISP, long distance, and cross-border communication.

Quota Description

By default, the L4 proxy service provides a quota of one instance for CNAME type access. If you need to access via Anycast IP or add more CNAME type instances, you can purchase by clicking **Adjust Quota** on the L4 Proxy Page. For detailed pricing of instances, please refer to L4 Proxy Instance.

Creating an L4 Proxy Instance

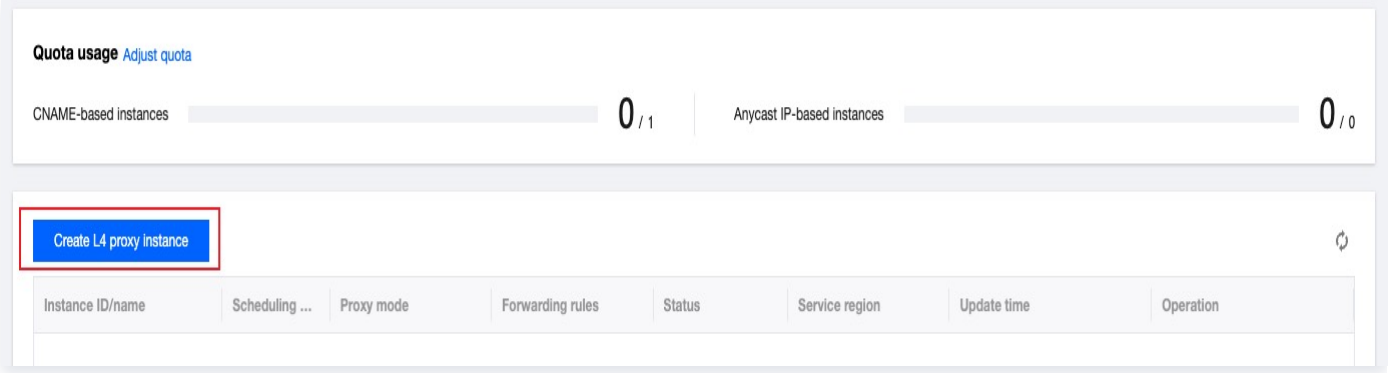
Last updated: 2023-09-07 15:39:27

Use Cases

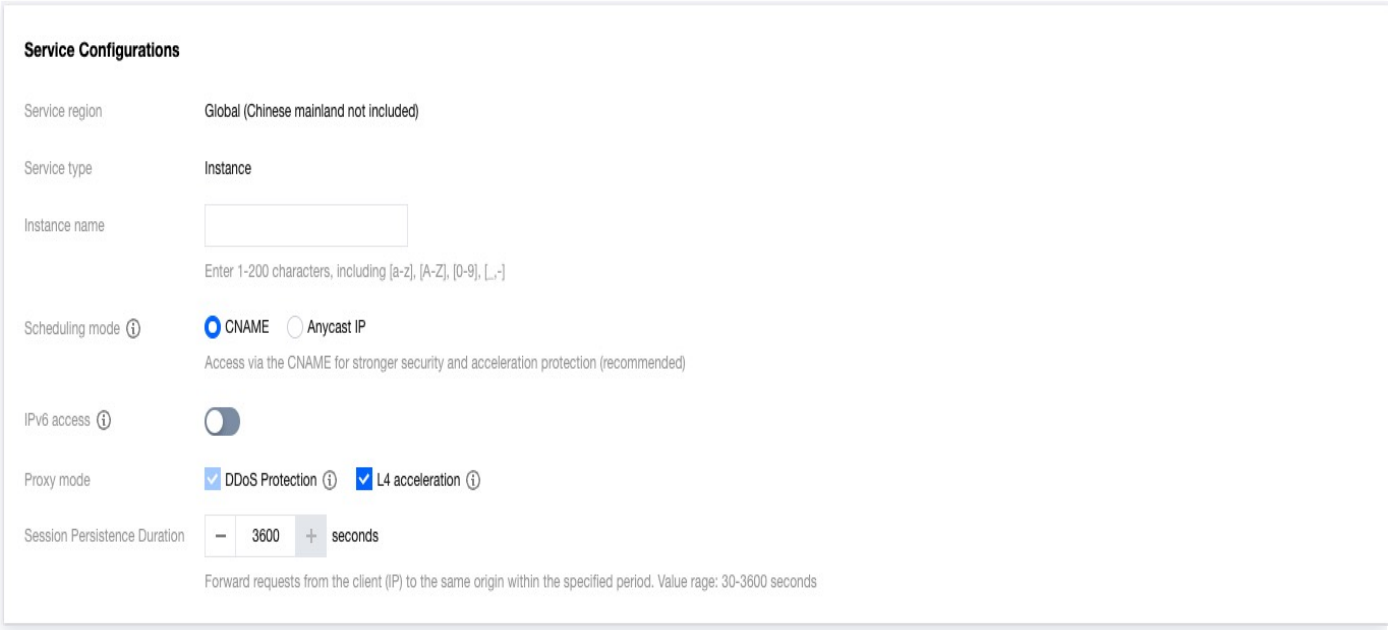
This document describes how to create and configure an L4 proxy instance.

Instructions

1. Log in to the EdgeOne console. In the left-hand menu, click on **Site List**. Within the site list, click on the **site** you wish to configure.
2. On the site details page, click **L4 proxy**.
3. On the page that appears, click **Create L4 proxy instance**.



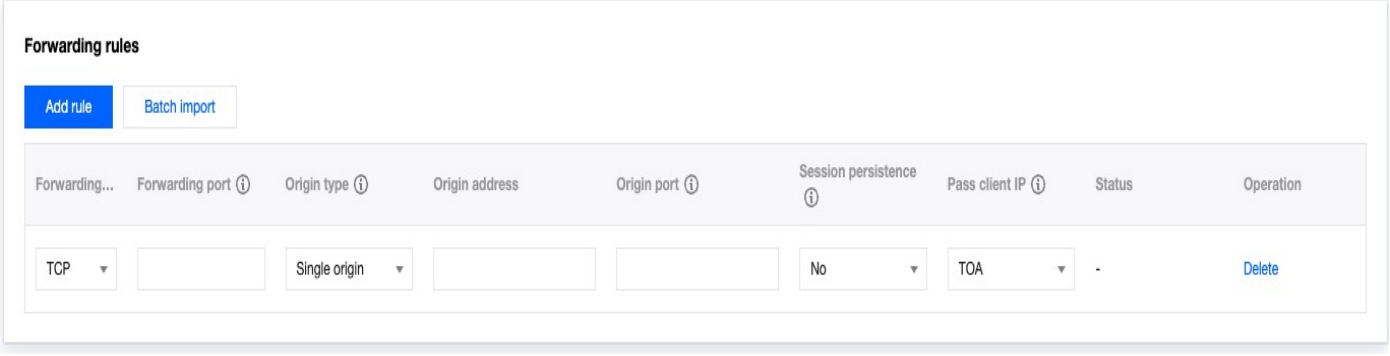
4. To create an L4 proxy, you need to specify the service configuration and forwarding rules. Start by specifying the service configuration. By default, the service region is the accelerated region of the current site. The following table describes the configuration parameters:



> **Note:**
> If site acceleration is also enabled for the host, the scheduling mode can only be set to **CNAME**.

Configuration items	Note
Instance Name	The name must be 1 to 200 characters in length and can contain uppercase and lowercase letters, digits, underscores (_), and hyphens (-).
Scheduling Mode	• **CNAME (Recommended)**: Utilizing a CNAME as the access address provides enhanced DDoS protection, supports proximate access acceleration, and L4 forwarding acceleration. • **Anycast IP**: An anycast IP address is used as the connection address, which supports DDoS protection and L4 forwarding and acceleration.
IPv6 access	If you enable this feature, EdgeOne nodes can be accessed over the IPv6 protocol.
Proxy mode	• **DDoS Protection**: Layer-3 and layer-4 DDoS protection. This feature is enabled by default and cannot be disabled. You can go to DDoS Mitigation to modify the default DDoS policy. • **L4 Acceleration:** The L4 forwarding acceleration feature reduces network transmission latency. You can choose to enable or disable this feature.
Session persistence duration.	During the specified session persistence duration, traffic from the same client IP will always be forwarded to the same origin. Value range: 30-3600 seconds.

5. Specify the forwarding rules. You can also import multiple forwarding rules at a time. For more information, see Batch Configuring Forwarding Rules. The table below lists the fields of a forwarding rule:



Note:

1. If site acceleration is also enabled for the host, forwarding ports 80 and 443 are not supported.
2. If you specify `Origin group` for **Origin type**, you can specify only self-owned origins. In this case, a COS bucket is not supported as the origin.
3. You can specify at most 2,000 forwarding rules for each L4 proxy instance.

Configuration items	Note
Forwarding protocol	Forwarding protocol of L4 proxy. Valid values: TCP and UDP.
Forwarding port	The supported port range is 1-64999. Multiple ports can be entered, separated by semicolons, and a hyphen can be used to denote a range of ports, such as 80-90. A maximum of 20 ports can be entered for a single forwarding rule. The following ports are reserved for internal use, please do not use them: • For TCP forwarding protocol: 3943, 3944, 6088, 36000, 56000. • For UDP forwarding protocol: 4789, 4790, 6080, 61708.
Origin type and Origin address	• **Single Origin:** Supports the input of a single origin IP address or domain name. • **Origin Group:** Select an origin from an existing origin group, or create a new origin group here.
Origin Port	You can specify a single port or a port range. If you specify a port range here, you must specify the same port range for **Forwarding port**. For instance, if the origin port range is 80-90, the forwarding port range must also be 80-90.
Session persistence	As long as an origin server IP remains unchanged, traffic from the same client IP will always be forwarded to the same origin server IP.
Pass client IP	• TOA: Pass client IPs via TCP Option (type 200), which only supports TCP protocols. • Proxy Protocol V1 (recommended): The Proxy Protocol V1 conveys the client's IP through the TCP header in plaintext. It supports TCP protocol but does not support UDP protocol. • Proxy Protocol V2: Client IPs are conveyed via the header. The V2 version employs a binary format and is compatible with both

	TCP and UDP protocols. Each TCP data packet carries a PPv2 header, while only the initial data packet of a UDP stream does so. • **Not passed**: Real client IPs will not be transferred.

6. Click **Submit configuration** to complete the creation of a new L4 proxy instance.

Modifying an L4 Proxy Instance

Last updated: 2023-09-07 15:39:32

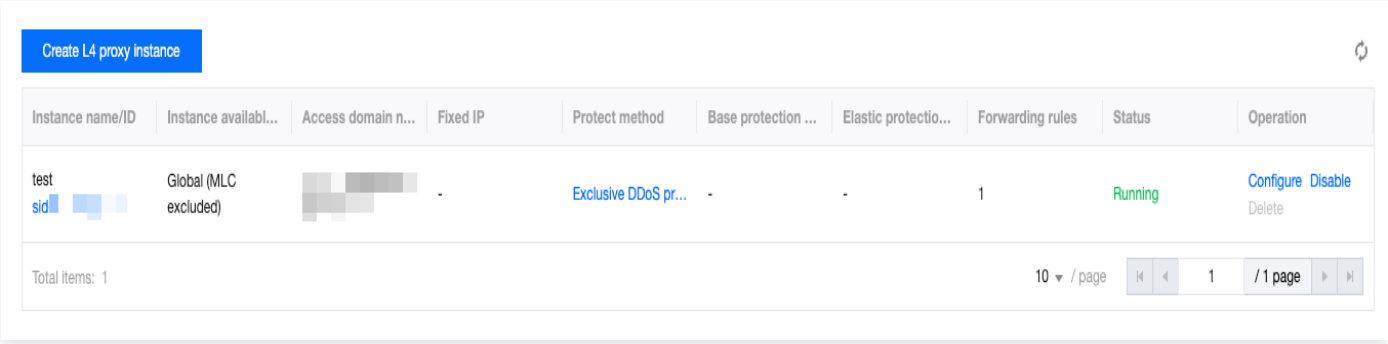
Use Cases

This document describes how to modify the configuration of an L4 proxy instance.

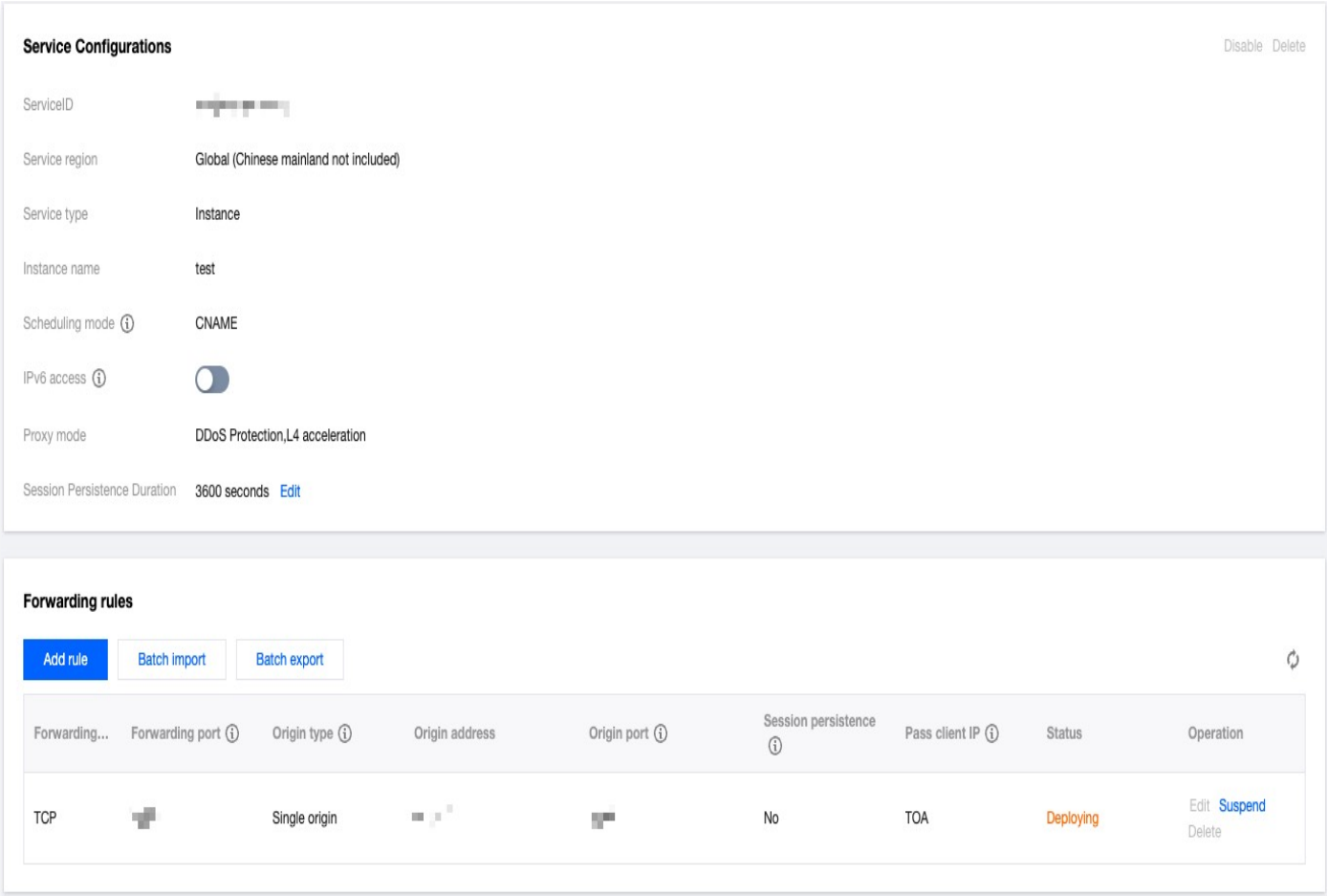
> **Note:**
> - After an L4 proxy instance is created, you cannot modify its scheduling mode or proxy mode. To do so, you can delete the instance and create a new one.
> - You can disable and then delete a forwarding rule.

Instructions

1. Log in to the EdgeOne console. In the left-hand menu, click on **Site List**. Within the site list, click on the **site** you wish to configure.
2. On the site details page, click **L4 proxy**.
3. On the Layer 4 proxy page, select the Layer 4 proxy rule that needs to be modified and click Configure.



4. For the established Layer 4 proxy mode, it supports the modification of IPv6 access configuration and session persistence duration. Additionally, on this page, you can add, edit, pause/resume, or delete forwarding rules.



Disabling or Deleting an L4 Proxy Instance

Last updated：2023-09-07 15:39:38

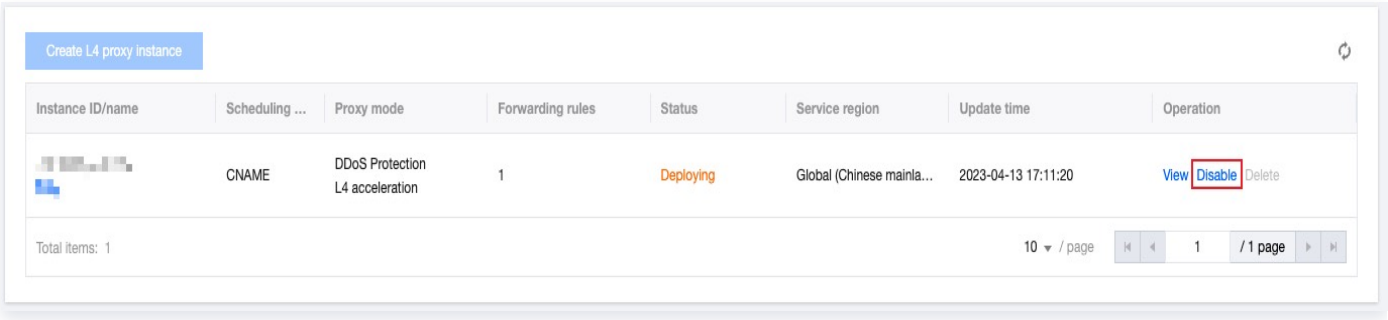
Use Cases

This document describes how to disable or delete an L4 proxy instance.

> **Note:**
> To delete an L4 proxy instance, you must disable it first, which usually takes a few minutes.

Instructions

1. Log in to the EdgeOne console. In the left-hand menu, click on **Site List**. Within the site list, click on the **site** you wish to configure.
2. On the site details page, click **L4 proxy**.
3. On the L4 Proxy page, select the instance you wish to disable and click **Disable**.



4. After disabling, if you wish to delete this instance, click **Delete** to remove the instance configuration.

Batch Configuring Forwarding Rules

Last updated: 2023-09-07 15:39:44

Use Cases

Tencent Cloud EdgeOne allows you to configure multiple forwarding rules for an L4 proxy instance. This document describes how to import and export multiple forwarding rules at a time.

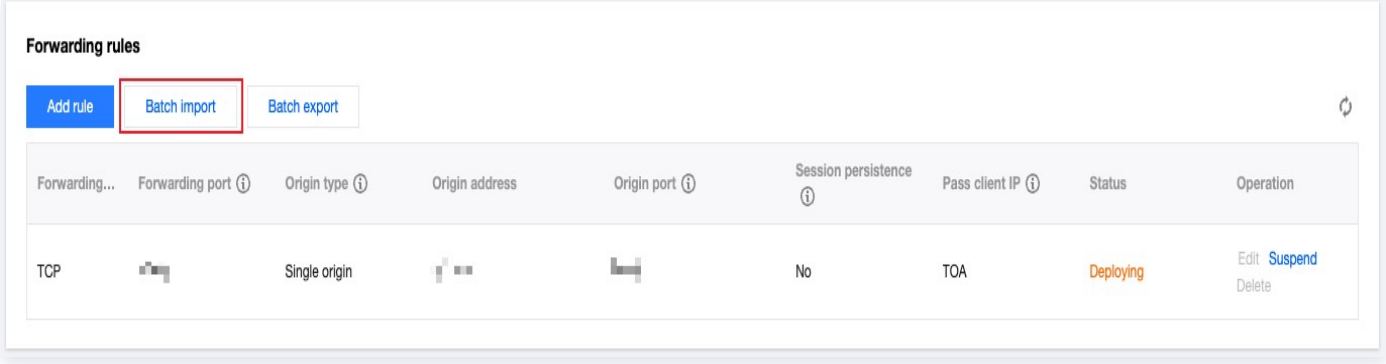
Note:

1. You can import up to 2,000 forwarding rules at a time. Each L4 proxy instance supports up to 2,000 forwarding rules.
2. The fields in batch imported forwarding rules are not case-sensitive.
3. The imported forwarding rules cannot use the forwarding ports of existing forwarding rules.

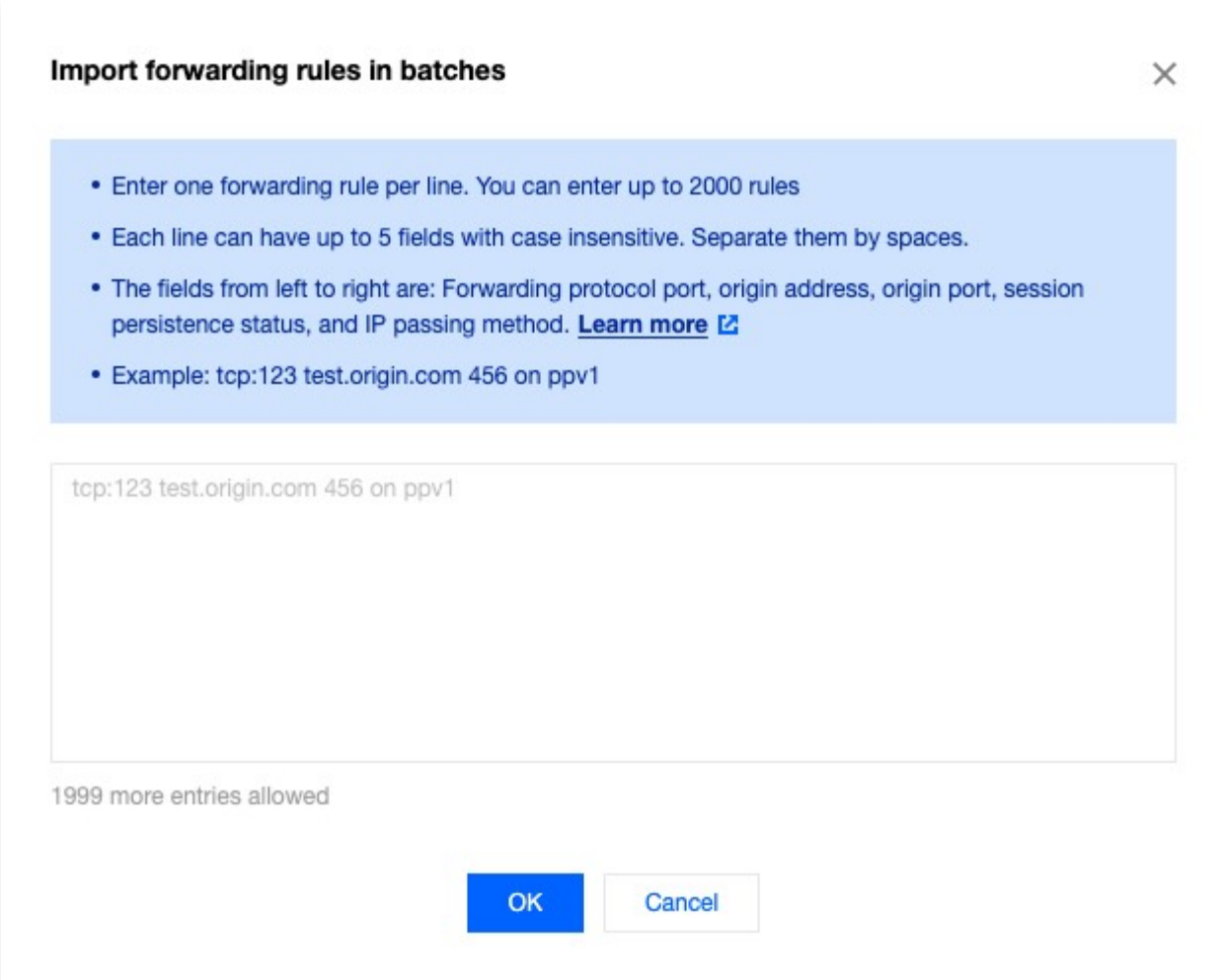
Instructions

Importing multiple forwarding rules at a time

1. Log in to the EdgeOne console. In the left-hand menu, click on **Site List**. Within the site list, click on the **site** you wish to configure.
2. On the site details page, click **L4 proxy**.
3. Navigate to the L4 Proxy page, select the L4 proxy instance you wish to modify, and click **View**.
4. On the forwarding rules page, click **Batch Import**.



5. In the pop-up window, enter the forwarding rules to be imported. You must enter one rule per row and specify the forwarding protocol, forwarding port, origin address, origin port, session persistence status, and IP passing mode. Separate fields with spaces. Example: `tcp:123 test.origin.com 456 on ppv1` .



The table below lists the fields of a forwarding rule:

Parameter	Note

Forwarding protocol:Forwarding port	• The supported forwarding protocols are TCP and UDP.
	• The supported port range is 1–64999. Multiple ports can be entered, separated by semicolons, and a hyphen can be used to denote a range of ports, such as 80–90. A maximum of 20 ports can be entered for a single forwarding rule.
	• The following ports are reserved for internal use, please do not use them:
	  ○ For TCP forwarding protocol: 3943, 3944, 6088, 36000, 56000.
	  ○ For UDP forwarding protocol: 4789, 4790, 6080, 61708.

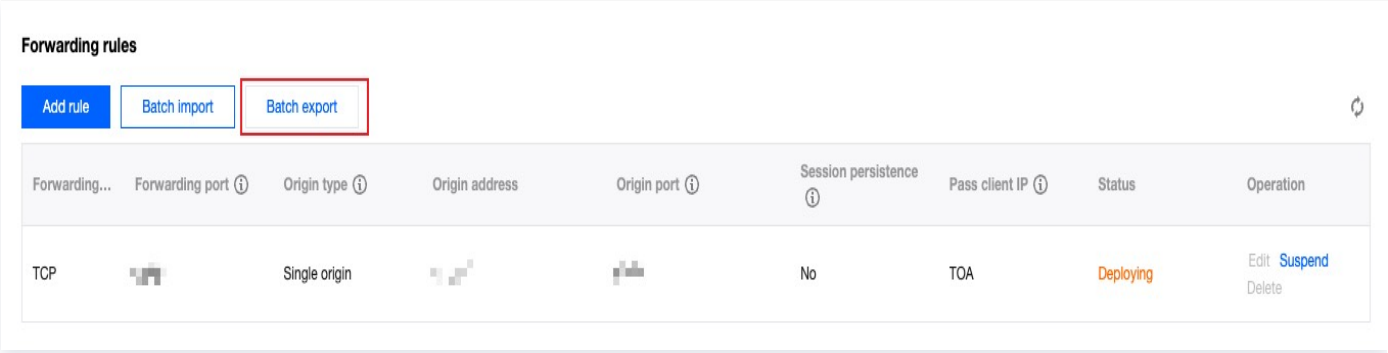
Origin address	• If you specify `Single origin` for **Origin type**, you can enter the IP address or domain name of a single origin. • You can specify the name of the origin server group in the format: `og:{OriginGroupName}` . For example: `og:testorigin` .

Origin Port	You can specify a single port or a port range. If you specify a port range here, you must specify the same port range for **Forwarding port**.
Session persistence	Valid values: on and off.
Pass client IP	Valid values: toa, ppv1, ppv2, and off.

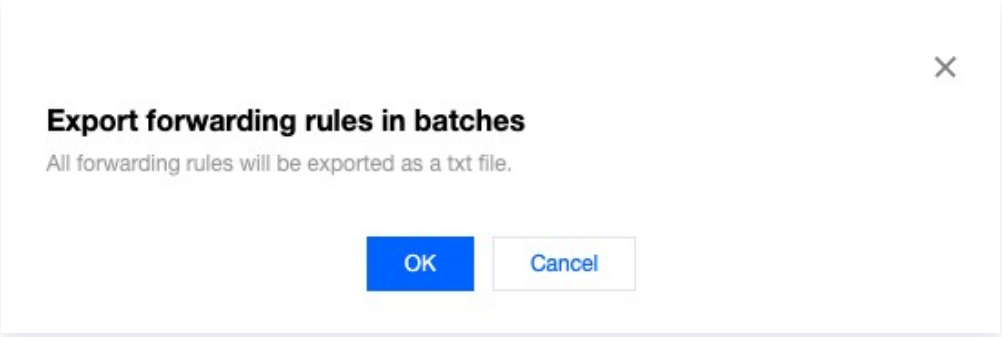
6. Click **OK** to import the forwarding rules.

Exporting multiple forwarding rules at a time

1. Log in to the EdgeOne console. In the left-hand menu, click on **Site List**. Within the site list, click on the **site** you wish to configure.
2. On the site details page, click **L4 proxy**.
3. On the Layer 4 proxy page, select the Layer 4 proxy rule that needs to be modified and click **View**.
4. On the Forwarding Rules page, click **Batch Export**.



5. In the dialog box that appears, click **OK** to export all forwarding rules. The rules will be exported in a .txt file, maintaining the same format as the imported rules.



Obtaining Real Client IPs
Obtaining Real TCP Client IPs via TOA

Last updated: 2023-09-07 15:39:53

This guide illustrates how to obtain the genuine TCP client IP via TOA during the utilization of Layer 4 proxy acceleration.

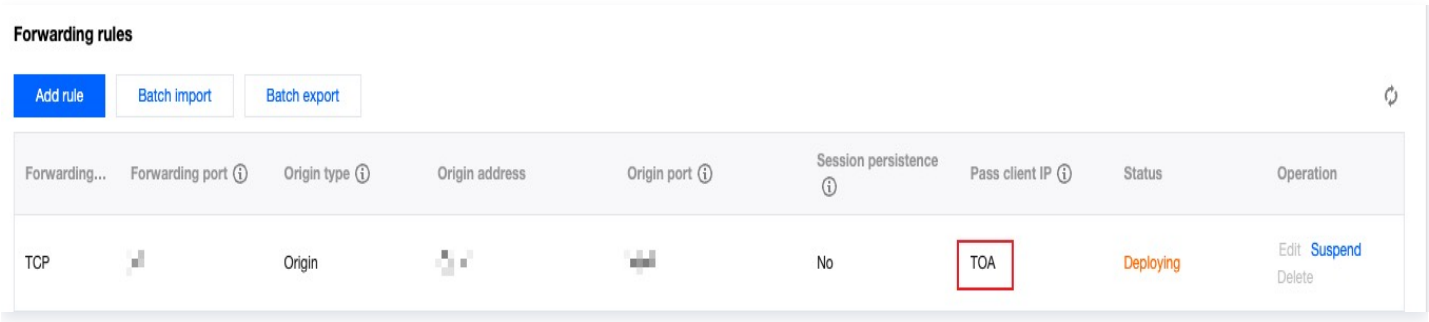
Use Cases

When data packets are accelerated through the Layer 4 acceleration channel, both the source IP address and source Port of the data packets are altered, preventing the origin server from directly obtaining the real client IP and Port information. To convey the genuine client IP and Port data to the origin server during the creation of the acceleration channel, you can opt to use TOA. The Layer 4 acceleration channel will place the real client IP and Port information into a custom tcp option field. It is necessary to install the TOA module on the origin server to retrieve the authentic client address information.

Instructions

Step 1: Select TOA as the method for transmitting the client IP

To obtain the genuine TCP client IP via TOA, it is necessary to configure the method of transmitting the client IP in the Layer 4 proxy forwarding rules within the console to TOA. For guidance on how to modify the Layer 4 proxy rules, refer to: Modifying a Layer 4 Proxy Instance.



Step 2: Load the TOA module on the backend server

You can load the TOA module using either of the following methods:

- Method 1 (Recommended): Based on the Linux version of the origin server, download the corresponding pre-compiled toa.ko file and load it directly.

- Method 2: If you cannot find the appropriate Linux version, download the TOA source code file and compile and load it yourself.

> **Note:**
> Due to variations in installation environments, if you encounter issues during the loading process using Method 1, please attempt Method 2, loading after independently installing the compilation environment.

Method 1: Download the precompiled TOA module and load it

1. Download and decompress the TOA package corresponding to the version of Linux OS on Tencent Cloud.
   - centos
     - CentOS-7.2-x86_64.tar.gz
     - CentOS-7.3-x86_64.tar.gz
     - CentOS-7.4-x86_64.tar.gz
     - CentOS-7.5-x86_64.tar.gz
     - CentOS-7.6-x86_64.tar.gz
     - CentOS-7.7-x86_64.tar.gz
     - CentOS-7.8-x86_64.tar.gz
     - CentOS-7.9-x86_64.tar.gz
     - CentOS-8.0-x86_64.tar.gz
     - CentOS-8.2-x86_64.tar.gz
   - debian
     - Debian-11.1-x86_64.tar.gz
     - Debian-10.2-x86_64.tar.gz
     - Debian-9.0-x86_64.tar.gz
   - suse linux
     - openSUSE-Leap-15.3-x86_64.tar.gz
   - ubuntu
     - Ubuntu-14.04.1-LTS-x86_64.tar.gz
     - Ubuntu-16.04.1-LTS-x86_64.tar.gz
     - Ubuntu-18.04.1-LTS-x86_64.tar.gz
     - Ubuntu-20.04.1-LTS-x86_64.tar.gz
2. Upon completion of the extraction, execute the cd command to enter the recently

decompressed folder, then proceed to load the TOA module as follows:

Execute the script with one click

```
/bin/bash -c "$(curl -fsSL https://edgeone-document-file-1258344699.cos.ap-gu
```

When it is loaded successfully, you will see the following information:

```
[root@VM-0-14-centos toa]# /bin/bash -c "$(curl -fsSL https://eo-toa-1258348367.cos.ap-shanghai.myqcloud.com/install_toa.sh)"
toa.ko install successfully
[root@VM-0-14-centos toa]#
```

Manual Configuration Loading

```
# Decompress the tar package
tar -zxvf CentOS-7.2-x86_64.tar.gz
# Enter the directory of the decompressed package
cd CentOS-7.2-x86_64
# Load the TOA module
insmod toa.ko
# Copy the TOA module to the kernel module directory.
cp toa.ko /lib/modules/`uname -r`/kernel/net/netfilter/ipvs/toa.ko
# Configure the TOA module to load automatically at system startup.
echo "insmod /lib/modules/`uname -r`/kernel/net/netfilter/ipvs/toa.ko" >> /etc/rc
```

Run the following command to check whether the loading is successful:

```
lsmod | grep toa
```

If you see "TOA" in the message, the module is loaded successfully:

```
[root@VM-0-14-centos toa]# lsmod | grep toa
toa                    282624  0
[root@VM-0-14-centos toa]#
```

Method 2: Compile and load the TOA module independently

1. Install the compilation environment.
   1.1 Ensure that kernel-devel and kernel-headers are installed and their version numbers align with the kernel version.
   1.2 Make sure the gcc and make dependencies are installed.
   1.3 If these environmental dependencies are not installed, run the installation command:

Centos

```
yum install -y gcc
yum install -y make
yum install -y kernel-headers kernel-devel
```

Ubuntu/Debian

```
apt-get install -y gcc
apt-get install -y make
apt-get install -y linux-headers-$(uname -r)
```

2. After the compilation environment is installed, execute the following command to download, compile, and load the source code.

Script for one-click compilation and loading

```
/bin/bash -c "$(curl -fsSL https://edgeone-document-file-1258344699.cos.ap-gu
```

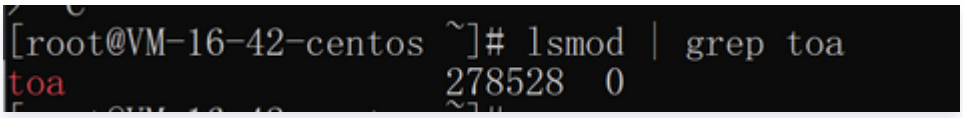
Manual Compilation and Loading

```
# Create a compilation directory and enter it.
mkdir toa_compile && cd toa_compile
# Download the source code (tar.gz)
curl -o toa.tar.gz https://edgeone-document-file-1258344699.cos.ap-guangzhou
# Decompress the tar package
tar -zxvf toa.tar.gz
Compile the toa.ko file. Upon successful compilation, a toa.ko file will be genera
make
# Load the TOA module
insmod toa.ko
# Copy the TOA module to the kernel module directory.
cp toa.ko /lib/modules/`uname -r`/kernel/net/netfilter/ipvs/toa.ko
# Configure the TOA module to load automatically at system startup.
echo "insmod /lib/modules/`uname -r`/kernel/net/netfilter/ipvs/toa.ko" >> /etc/rc
```

3. Run the following command to check whether loading is successful:

```
lsmod | grep toa
```

If you see "TOA" in the message, the module is loaded successfully, as shown in the image below:



Step 3: Verify the acquisition of client IP information

You can verify the configuration by building a TCP server to receive client requests from another server. See the sample:

1. On the current server, create an HTTP server in Python to act as a TCP server, as shown below:

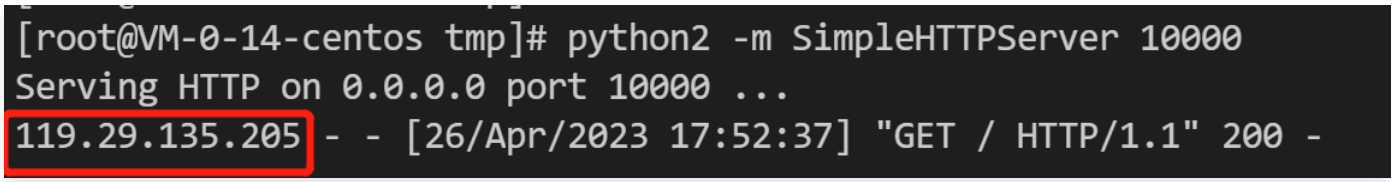
```
# Use python2
python2 -m SimpleHTTPServer 10000

# Use python3
```

```
python3 -m http.server 10000
```

2. Utilize an alternate server to function as a client, formulate client requests, and simulate TCP requests using Curl:

```
# Use curl to initiate an HTTP request, where the hostname and forwarding port of the
curl -i "http://a8b7f59fc8d7e6c9.example.com.edgeonedy1.com:10000/"
```

3. If TOA is fully loaded, the server with TOA installed will display the actual client address information as shown in the red box below:



If your current operation falls under either of the two scenarios where only one type of client address, either IPv4 or IPv6, is required, then by following the aforementioned steps to load the TOA module on the server side, you can obtain the real IP address of the client.

- For origin IPv4 addresses, get the client IPv4 address.
- For origin IPv6 addresses, get the client IPv6 address.

However, if your current business origin needs to simultaneously obtain both IPv4 and IPv6 client addresses, you will need to modify the origin's business code while loading the TOA module. Please continue to refer to the following guide: Modify the origin's business code to support the simultaneous acquisition of real IPv4/IPv6 client addresses .

Modify the origin's business code to simultaneously obtain the genuine IPv4/IPv6 client IP.

> **Note:**
> This section provides guidance on how to obtain both IPv4 and IPv6 client address information by modifying the business code of the origin server.

The origin can listen on requests in either of the following methods:

1. Use the structure `struct sockaddr_in` to listen on IPv4 addresses.
2. Use the structure `struct sockaddr_in6` to listen on IPv6 addresses.

Listening IPv4 Address
Sample code in C language

```
#include <sys/socket.h>
#include <stdio.h>
#include <unistd.h>
#include <netinet/in.h>
#include <memory.h>
#include <arpa/inet.h>

int main(int argc,  char **argv){
    int    l_sockfd;
    // The server address is an IPv4 address.
    struct sockaddr_in serveraddr;
    // In this case, the client address must adopt the IPv6 structure.
    struct sockaddr_in6 clientAddr;
    int server_port = 10000;

    memset(&serveraddr, 0, sizeof(serveraddr));
    // Create a socket
    l_sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (l_sockfd == -1){
        printf("Failed to create socket.\n");
        return -1;
    }

    // Initialize the server address information
    memset(&serveraddr, 0, sizeof(struct sockaddr_in));
    serveraddr.sin_family = AF_INET;
    serveraddr.sin_port = htons(server_port);
    serveraddr.sin_addr.s_addr = htonl(INADDR_ANY);

    int isReuse = 1;
    setsockopt(l_sockfd, SOL_SOCKET,SO_REUSEADDR,(const
char*)&isReuse,sizeof(isReuse));

    // Associate the socket with the server address information
    int nRet = bind(l_sockfd,(struct sockaddr*)&serveraddr, sizeof(serveraddr));
    if(-1 == nRet)
    {
        printf("bind error\n");
        return -1;
    }
    // Listen on the socket.
    listen(l_sockfd, 5);

    int clientAddrLen = sizeof(clientAddr);
    memset(&clientAddr, 0, sizeof(clientAddr));
```

```
    // Accept connections from the client
    int linkFd = accept(l_sockfd, (struct sockaddr*)&clientAddr, &clientAddrLen);
    if(-1 == linkFd)
    {
        printf("accept error\n");
        return -1;
    }
    // Modifications to make: Decide whether the client is an IPv4 or IPv6 address
based on sin6_family.
    //    AF_INET indicates that the client adopts IPv4. In this case, convert the
client address pointer to struct sockaddr_in* and get the IPv4 address.
    //    AF_INET6 indicates that the client adopts IPv6. In this case, use struct
sockaddr_in6* to get the IPv6 address.
    if (clientAddr.sin6_family == AF_INET) {
        printf("AF_INET accept getpeername %s : %d successful\n",
                inet_ntoa(((struct sockaddr_in*)&clientAddr)->sin_addr),
                ntohs(((struct sockaddr_in*)&clientAddr)->sin_port));
    }else if (clientAddr.sin6_family == AF_INET6){
        char addr_p[128] = {0};
        inet_ntop(AF_INET6, (void *)&((struct sockaddr_in6*)&clientAddr)->sin6_addr,
addr_p, (socklen_t )sizeof(addr_p));
        printf("AF_INET6 accept getpeername %s : %d successful\n",
                addr_p,
                ntohs(((struct sockaddr_in6*)&clientAddr)->sin6_port));
    }else{
        printf("unknow sin_family:%d \n", clientAddr.sin6_family);
    }
    close(l_sockfd);
    return 0;
}
```

Listening to IPv6 addresses

Sample code in C language

```
#include <sys/socket.h>
#include <stdio.h>
#include <unistd.h>
#include <netinet/in.h>
#include <memory.h>
#include <arpa/inet.h>
```

```
int main(int argc,  char **argv)
{
    int    l_sockfd;
    // The server address is an IPv6 address.
    struct sockaddr_in6 serveraddr;
    // The client address is an IPv6 address.
    struct sockaddr_in6 clientAddr;
    int server_port = 10000;

    memset(&serveraddr, 0, sizeof(serveraddr));

    // Create a socket
    l_sockfd = socket(AF_INET6, SOCK_STREAM, 0);
    if (l_sockfd == -1){
        printf("Failed to create socket.\n");
        return -1;
    }
    // Set the server address information
    memset(&serveraddr, 0, sizeof(struct sockaddr_in6));
    serveraddr.sin6_family = AF_INET6;
    serveraddr.sin6_port = htons(server_port);
    serveraddr.sin6_addr = in6addr_any;

    int isReuse = 1;
    setsockopt(l_sockfd, SOL_SOCKET,SO_REUSEADDR,(const char*)&isReuse,sizeof(is
    // Associate the socket with the server address information
    int nRet = bind(l_sockfd,(struct sockaddr*)&serveraddr, sizeof(serveraddr));
    if(-1 == nRet)
    {
        printf("bind error\n");
        return -1;
    }
    // Listen on the socket.
    listen(l_sockfd, 5);

    int clientAddrLen = sizeof(clientAddr);
    memset(&clientAddr, 0, sizeof(clientAddr));

    // Accept connection requests from the client
    int linkFd = accept(l_sockfd, (struct sockaddr*)&clientAddr, &clientAddrLen);
    if(-1 == linkFd)
    {
        printf("accept error\n");
        return -1;
    }
```

```
    // The client addresses received here are all stored in the IPv6 structure.
    // The IPv4 addresses are mapped to IPv6 addresses, for example, "::ffff:119.29.1.
    char addr_p[128] = {0};
    inet_ntop(AF_INET6, (void *)&clientAddr.sin6_addr, addr_p, (socklen_t )sizeof(addr
    printf("accept %s : %d successful\n", addr_p, ntohs(clientAddr.sin6_port));

    // Modifications to make: Use the macro definition IN6_IS_ADDR_V4MAPPED to dec
    if(IN6_IS_ADDR_V4MAPPED(&clientAddr.sin6_addr)) {
        struct sockaddr_in real_v4_sin;
        memset (&real_v4_sin, 0, sizeof (struct sockaddr_in));
        real_v4_sin.sin_family = AF_INET;
        real_v4_sin.sin_port = clientAddr.sin6_port;
        // The last four bytes represent the IPv4 address of the client.
        memcpy (&real_v4_sin.sin_addr, ((char *)&clientAddr.sin6_addr) + 12, 4);
        printf("connect %s successful\n", inet_ntoa(real_v4_sin.sin_addr));
    }

    close(l_sockfd);
    return 0;
}
```

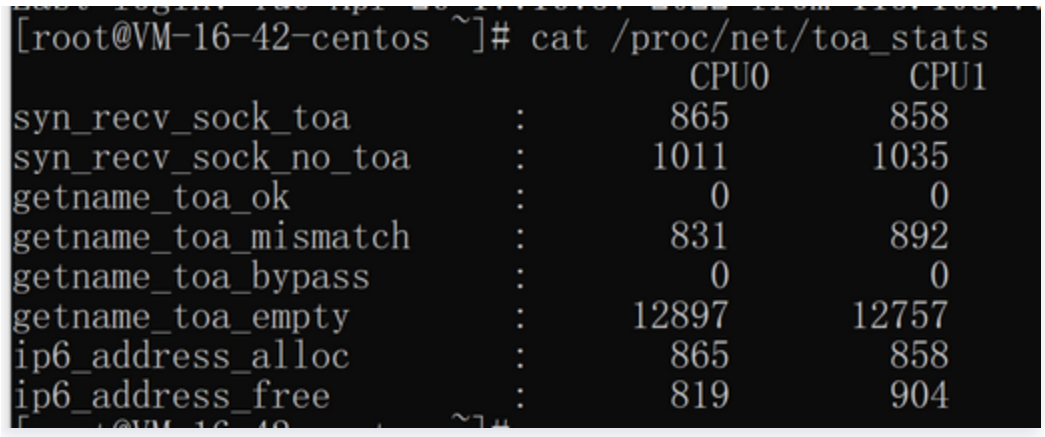
References

Monitoring TOA Running Status

To ensure the stability of the TOA kernel module, it also provides a monitoring function. After inserting the toa.ko kernel module, you can monitor the working status of the TOA module by executing the following command.

```
cat /proc/net/toa_stats
```

The TOA running status is as follows:



The monitoring metrics are described as follows:

Description	Note
syn_recv_sock_toa	Receives connections with TOA information.
syn_recv_sock_no_toa	Receives connections without TOA information.
getname_toa_ok	This count increases when you call `getsockopt` and get the source IP successfully or when you call `accept` to receive client requests.
getname_toa_mismatch	When invoking getsockopt to retrieve the source IP, this count increases when the type does not match. For instance, if an IPv4 source IP is stored in a client connection instead of an IPv6 address, this count will increment.
getname_toa_empty	This count increases when the `getsockopt` function is called in a client file descriptor that does not contain TOA.
ip6_address_alloc	Allocates space to store the information when TOA gets the source IP and source port saved in the TCP data packet.
ip6_address_free	When the connection is released, TOA will release the memory previously used to save the source IP and source port. If all connections are closed, the total count of `ip6_address_alloc` for each CPU should be equal to the count of this metric.

Obtaining Real Client IPs Through Protocol V1/V2
Overview

Last updated: 2023-09-07 15:40:00

This document elucidates the process of obtaining the genuine client IPs via Proxy Protocol V1/V2 when employing the L4 proxy acceleration.

Use Cases

When data packets are expedited via the L4 acceleration channel, in order to convey the authentic client IP and Port details to the origin server, you may opt to transmit the client IP and Port information through the Proxy Protocol V1/V2. For a comprehensive understanding of the protocol, refer to Proxy Protocol V1/V2.

When the origin server is parsing to obtain the actual client IP, depending on the business scenario and deployment method, the following two methods can be referred to understand how to acquire the real client IP:

- Method 1: If the TCP protocol is utilized by your origin server, and Nginx natively supports the Proxy Protocol, it is advisable to append a Nginx server that supports Proxy Protocol V1/V2 ahead of the business server to procure the genuine client IP. For detailed steps, please refer to Obtaining Real Client IPs Through Nginx.
- Method 2: If your origin server employs the UDP protocol, or if there is a need to directly parse the real client IP under the TCP protocol scenario within the business origin service for business scheduling, you can develop your own parsing of the Proxy Protocol field within the business origin by referring to the sample code in the Proxy Protocol. For detailed steps, please refer to Parsing Real Client IPs on Application Server.

Method 1: Obtaining Real Client IPs Through Nginx

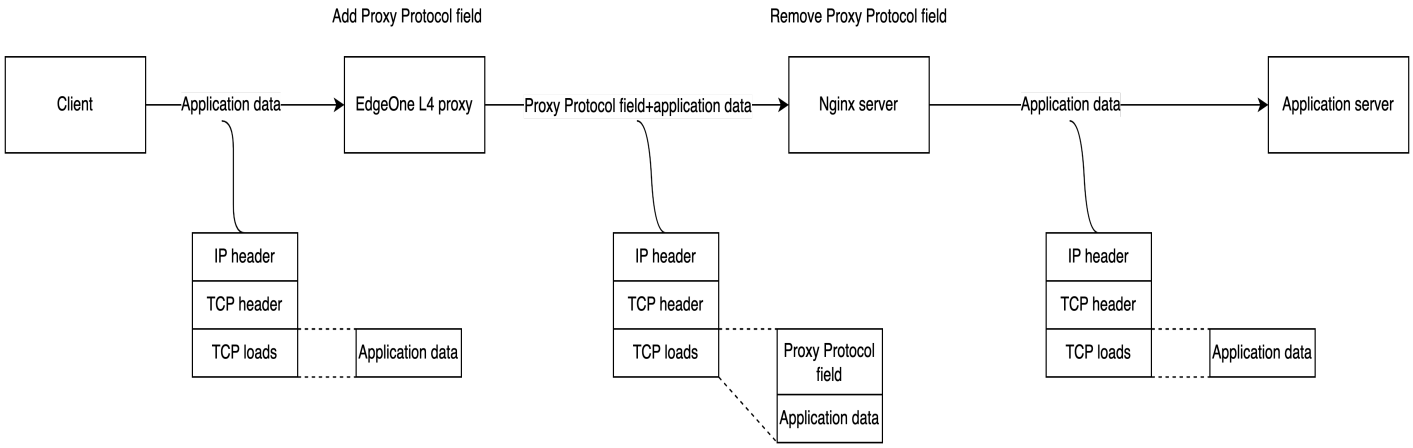
Last updated: 2023-09-07 15:40:06

Use Cases

If your origin server operates on the TCP protocol and the current Nginx natively supports the Proxy Protocol, it is advisable to incorporate a Nginx server that supports Proxy Protocol V1/V2 ahead of the application server to acquire the genuine client IP. You may follow the subsequent steps for implementation.

> **Note:**
> If your origin server operates on the TCP protocol and you prefer not to deploy an Nginx service for individual parsing of the real client IP, but rather wish to directly parse and obtain the real client IP within the application server to aid business decision logic, you may refer to: Parsing Real Client IPs on Application Server.

Deployment directions



As depicted in the diagram above, you need to deploy a Nginx server ahead of the application server. The Nginx server will handle the unloading of the Proxy Protocol field. The collection of real client IP addresses can be accomplished by analyzing the Nginx logs on the Nginx server, thereby relieving the application server from the concern of real client addresses. At this point, when configuring the origin address in the EdgeOne L4 proxy service, you can direct the origin address to this Nginx service.

Instructions

Step 1: Deploy Nginx Service

Please select a Nginx version corresponding to the Proxy Protocol version you want to use:

- For Proxy Protocol V1: Nginx Plus R11 and later versions, Nginx Open Source 1.11.4 and later versions.
- For Proxy Protocol V2: Nginx Plus R16 and later versions, Nginx Open Source 1.13.11 and later versions.
- For information on other Nginx versions' support for the Proxy Protocol, please refer to the Nginx documentation: Accepting the PROXY Protocol.

To facilitate the L4 proxy service on Nginx, it is necessary to install Nginx-1.18.0 and its stream module. The installation process is as follows:

```
# Install the nginx build environment dependencies
yum -y install gcc gcc-c++ autoconf automake
yum -y install zlib zlib-devel openssl openssl-devel pcre-devel

# Decompress the source package
tar -zxvf nginx-1.18.0.tar.gz
# Enter the directory
cd nginx-1.18.0
# Configure the nginx compilation and installation settings, including --with-stream
./configure --prefix=/opt/nginx --sbin-path=/opt/nginx/sbin/nginx --conf-path=/opt/nginx/c
# Compilation
make
# Installation
make install
```

Step 2: Configure the stream module within Nginx

Using Nginx-1.18.0 as an example, the following command can be executed to access the Nginx configuration file, nginx.conf:

```
vi /opt/nginx/conf/nginx.conf
```

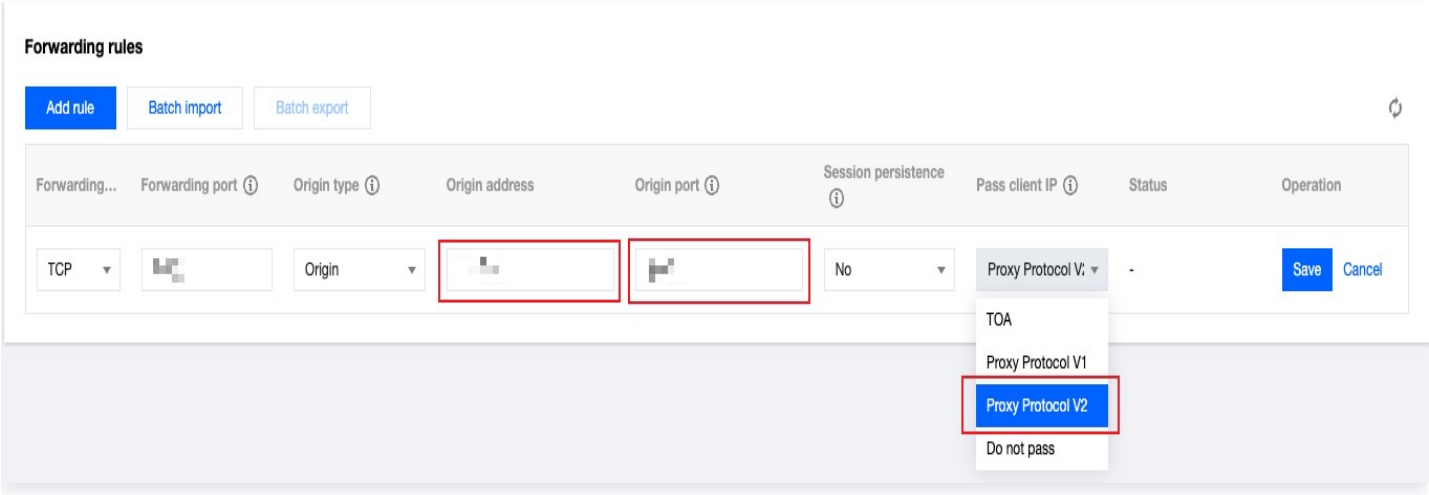
Configuration of the stream module is as follows:

```
stream {
    # Set the log format, where proxy_protocol_addr is the client address obtained by par
    log_format basic '$proxy_protocol_addr -$remote_addr [$time_local] '
                     '$protocol $bytes_sent $bytes_received '
                     '$session_time';
```

```
    access_log  logs/stream.access.log  basic;
    # upstream configuration
    upstream RealServer {
        hash $remote_addr consistent;
        # 127.0.0.1:8888 is the IP address and port of the application server
        server 127.0.0.1:8888 max_fails=3 fail_timeout=30s;
    }
    # Server Configuration
    server{
        # L4 listening port, which corresponds to the origin port configured in L4 proxy servi
        listen 10000 proxy_protocol;
        proxy_connect_timeout 1s;
        proxy_timeout 3s;
        proxy_pass RealServer;
    }
}
```

Step 3: Configure L4 proxy forwarding rule

Upon configuring the Nginx service, you can proceed to the L4 proxy service in the console, modify the L4 proxy forwarding rules. Alter the origin address to the IP of the current Nginx service, and the origin port to the L4 listening port configured in Step Two. When transmitting the client IP, select either Proxy Protocol V1 or Proxy Protocol V2, depending on the support status of your current Nginx version.



Step 4: Simulate client requests and verify results

You can establish a TCP service, then utilize an alternate server to simulate client requests for validation. The specific example is as follows:

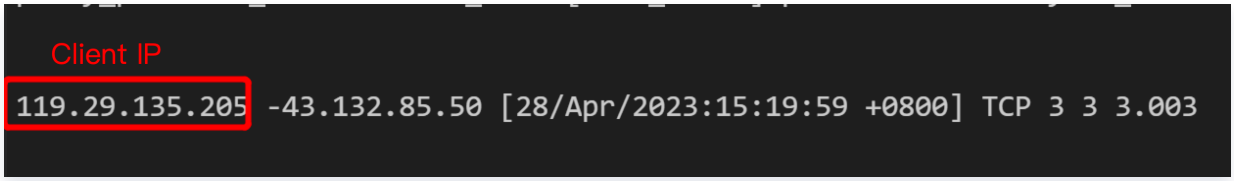
1. An HTTP service can be established on the current server using Python to emulate a TCP service.

```
# Use python2
python2 -m SimpleHTTPServer 8888

# Use python3
python3 -m http.server 8888
```

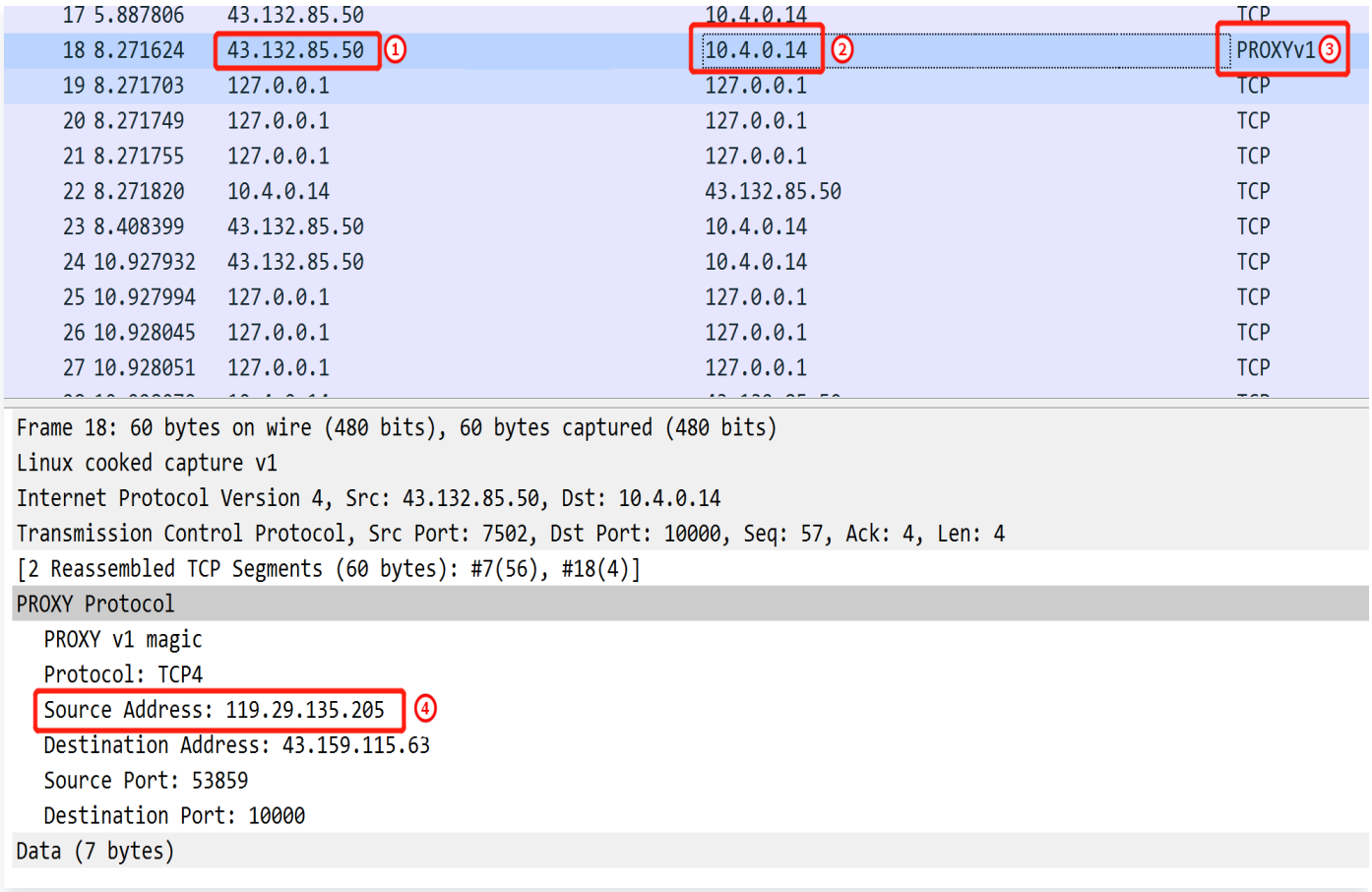
2. Utilize an alternate server to function as a client, formulate client requests, and simulate TCP requests using Curl:

```
# Initiate an HTTP request with curl, where the domain is the L4 proxy domain, and 8
curl -i "http://d42f15b7a9b47488.davidjli.xyz.acc.edgeonedy1.com:8888/"
```

3. Inspect the Nginx logs on the Nginx server as shown below:



You can capture packets on the Nginx server and analyze them using Wireshark. After the TCP handshake is completed, the Proxy Protocol field will be added to the front of the first business data packet. Here is an example of Proxy Protocol V1: ①L4 Proxy exit IP, ②Nginx server IP, ③Protocol version, ④Real client IP address.



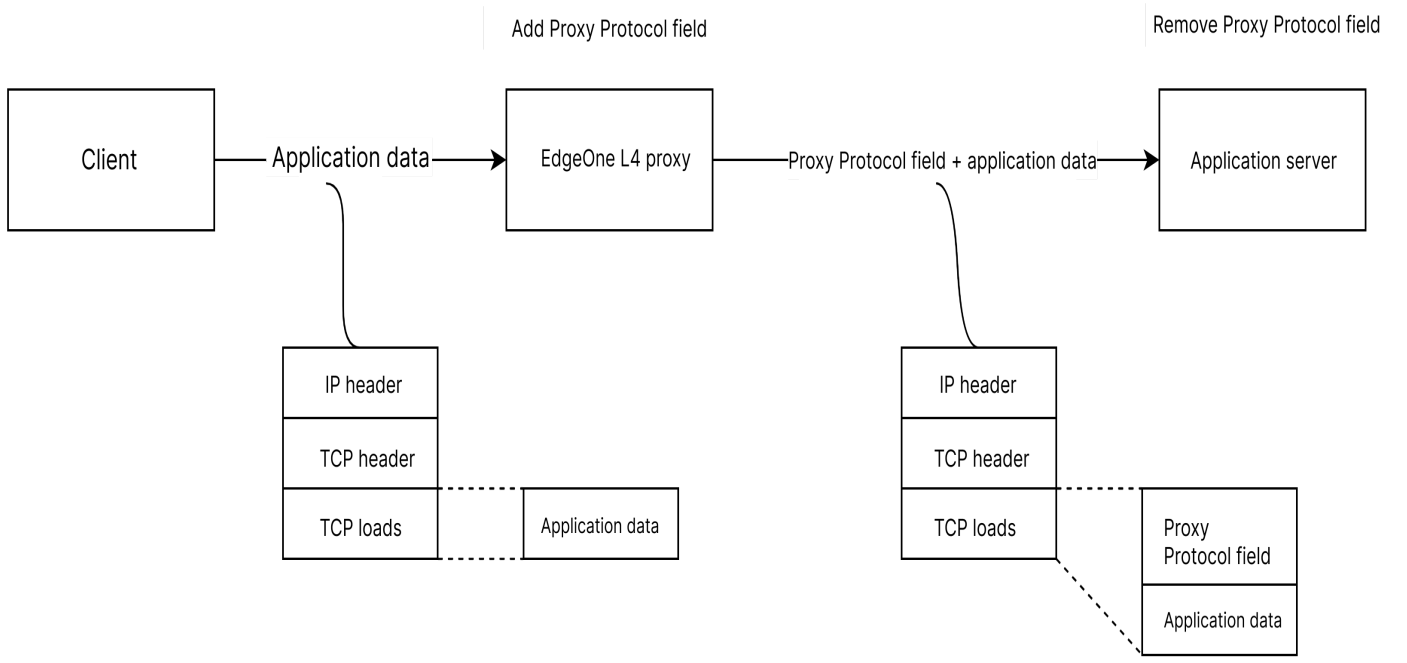
Method 2: Parsing Real Client IPs on Application Server

Last updated: 2023-09-07 15:40:12

Use Cases

- Scenario 1: If the UDP protocol is used on the origin, only Proxy Protocol V2 supports the transmission of real client IPs via the UDP protocol. However, as Nginx does not support the parsing of the Proxy Protocol V2 in UDP scenarios, it is necessary to manually parse the Proxy Protocol V2 on the application server to obtain the real client IPs.
- Scenario 2: If the TCP protocol is used on the origin, and you want to implement application judgment via the real client IPs on the application server, you need to parse the Proxy Protocol V1/V2 on the application server to obtain the real client IPs.

Deployment Diagram

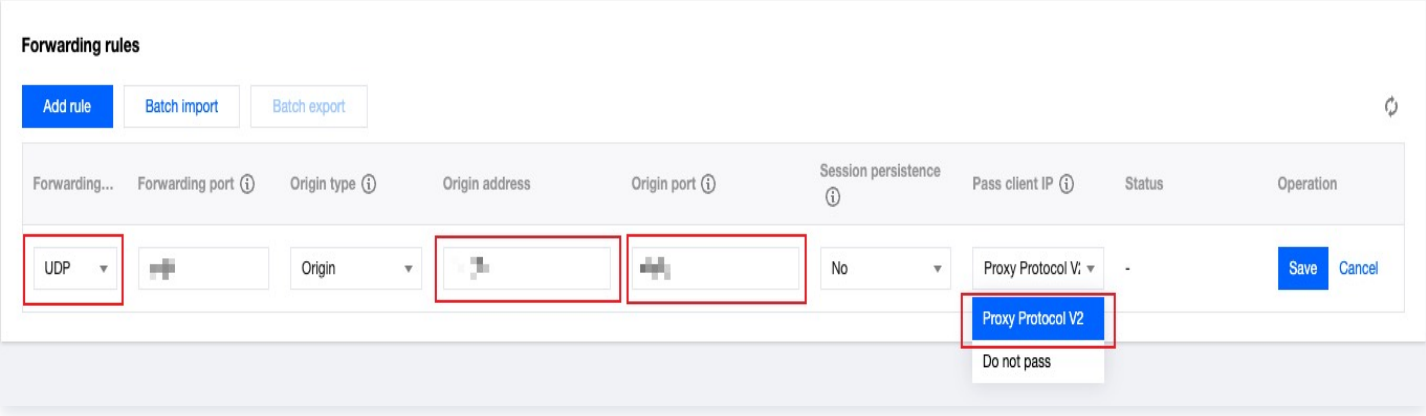


As depicted in the above diagram, you can configure the L4 proxy to point to your application server via the EdgeOne L4 proxy module. The EdgeOne L4 proxy service adds the Proxy Protocol field during data transmission, which is then parsed by the application server.

Instructions

Step 1: Configure the L4 Proxy Forwarding Rule

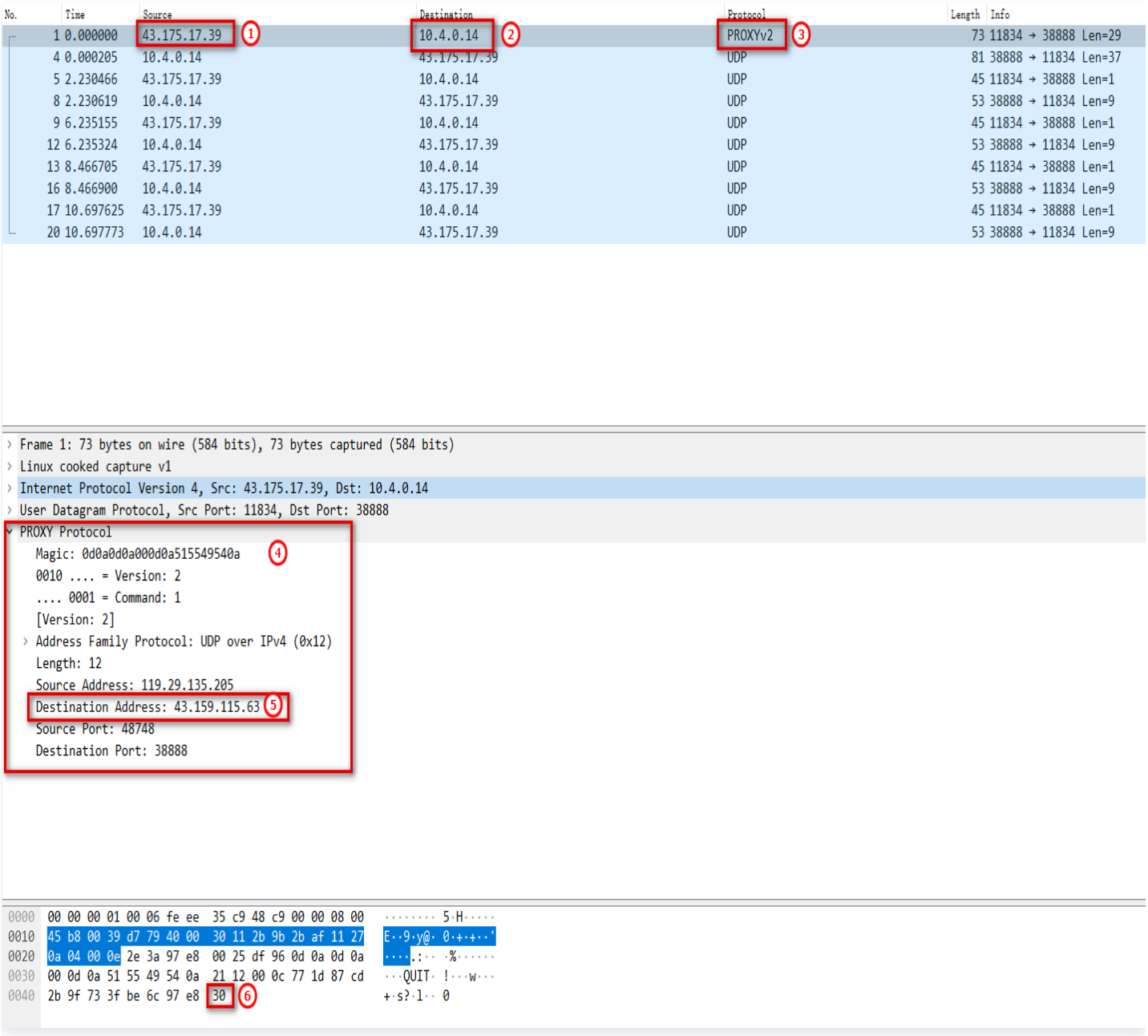
Navigate to the L4 proxy service in the console and modify the L4 proxy forwarding rules. Fill in the corresponding application origin address and origin port. If your current forwarding protocol is UDP, select Proxy Protocol V2 for passing the client IP. If the current forwarding protocol is TCP, either Proxy Protocol V1 or V2 can be selected for passing the real client IP.



Step 2: Obtain real client IPs on the application server by parsing the Proxy Protocol field.

You need to parse the Proxy Protocol filed with reference to the sample code in the Proxy Protocol. For the format of the client IPs, see Format of Real Client IPs Obtained Through Proxy Protocol V1/V2.

In the context of UDP transmission, when using Proxy Protocol V2, the Proxy Protocol field is appended to the first UDP datagram. The diagram illustrates the following: ① represents the L4 proxy egress IP, ② denotes the origin address, ③ signifies the protocol version, ④ indicates the Proxy Protocol field, ⑤ refers to the actual client IP address, and ⑥ points to the application data.

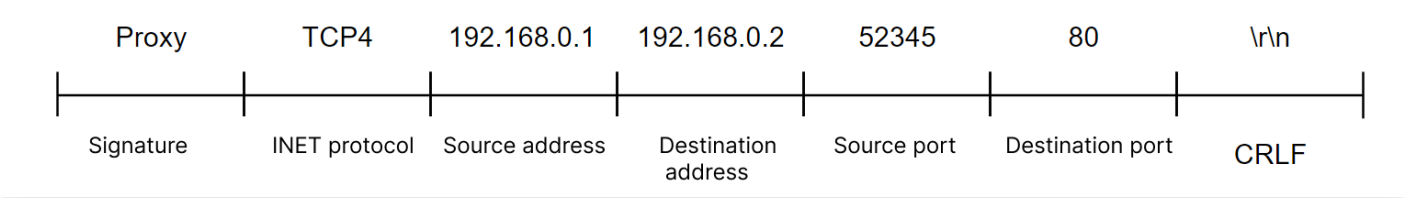


Format of Real Client IPs Obtained Through Proxy Protocol V1/V2

Last updated: 2023-09-07 15:40:17

Proxy Protocol V1

Proxy Protocol V1 supports TCPv4 and TCPv6, and adopts string format. See details below:




```
PROXY TCP4 192.168.0.1 192.168.0.11 56324 443\r\n
```

The following information can be observed by utilizing the Wireshark packet capture tool:

```
PROXY Protocol
    PROXY v1 magic
    Protocol: TCP4
    Source Address: 119.29.135.205
    Destination Address: 43.159.115.63
    Source Port: 53859
    Destination Port: 10000
Data (7 bytes)
```

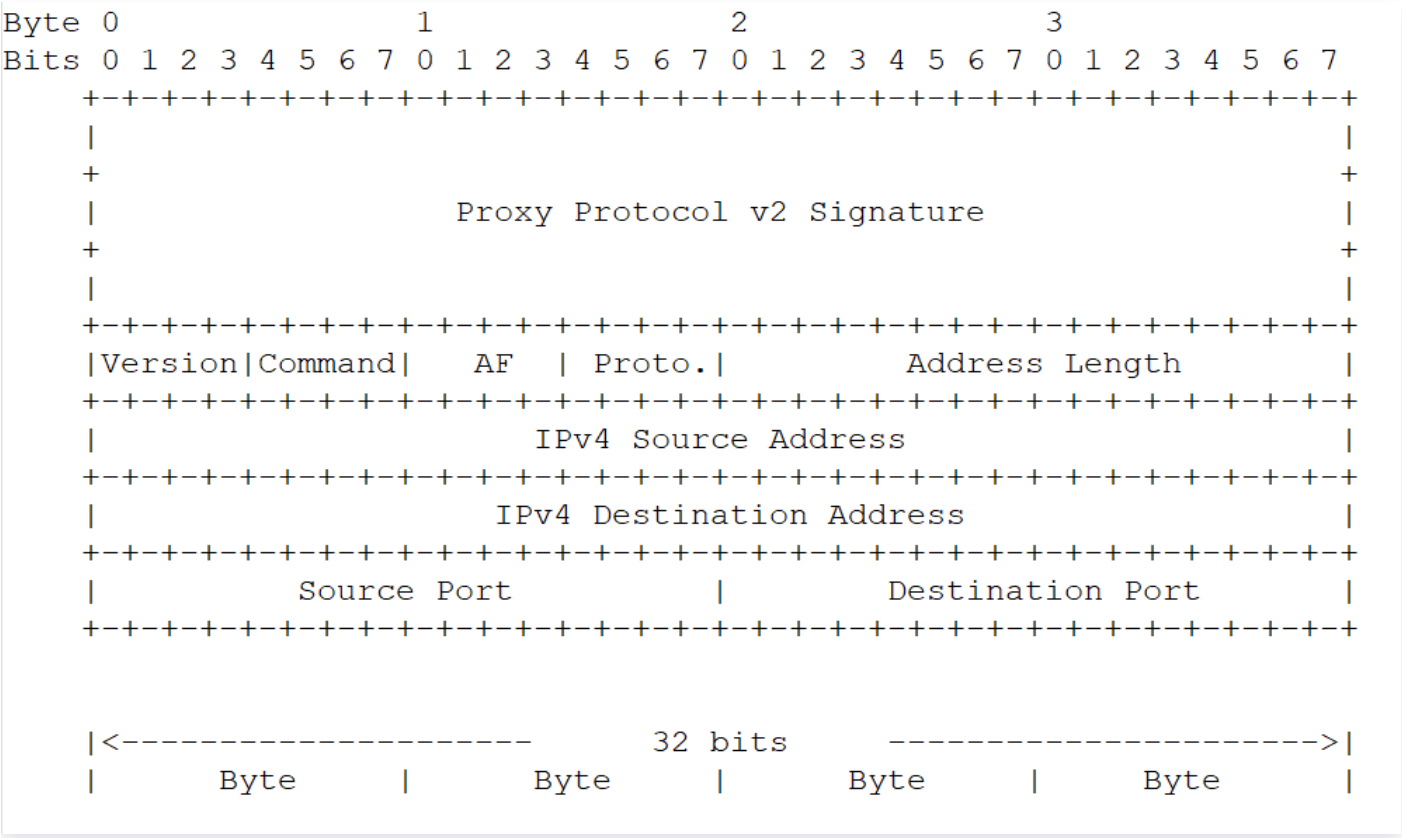
String format

```
0000  50 52 4f 58 59 20 54 43  50 34 20 31 31 39 2e 32   PROXY TC P4 119.2
0010  39 2e 31 33 35 2e 32 30  35 20 34 33 2e 31 35 39   9.135.20 5 43.159
0020  2e 31 31 35 2e 36 33 20  35 33 38 35 39 20 31 30   .115.63  53859 10
0030  30 30 30 0d 0a 61 62 63  31 32 34 33               000··abc 1243
```

Proxy Protocol V2

Proxy Protocol V2 supports TCPv4, TCPv6, UDPv4 and UDPv6, and adopts the binary format. See details below:

IPv4 format



IPv6 format