



Traffic Protection for Third-Party Applications

Traffic Protection for Third-Party Applications provides a mechanism for securing management traffic on the router. Without Traffic Protection for Third-Party Applications, if the service is enabled, the Cisco IOS XR allows the service traffic to pass through any interface with a network address.



Note Prior to Cisco IOS XR Release 6.5.2, Traffic Protection for Third-Party Applications was termed as MPP for Third-Party Applications.

Traffic Protection for Third-Party Applications helps in rate limiting or throttling the traffic through configuration with the help of LPTS. Traffic Protection for Third-Party Applications filters traffic based on the following tuples: address family, vrf, port, interface, local address and remote address.



Note It is mandatory to configure address family, protocol, local port, and vrf, as well as at least one of interface or local or remote address.

Prior to Cisco IOS XR Release 6.5.2, only the default vrf is supported for TPA configuration.

- [gRPC Protocol, on page 1](#)
- [Limitations for Traffic Protection for Third-Party Applications, on page 2](#)
- [Prerequisites for Traffic Protection for Third-Party Applications Over GRPC, on page 2](#)
- [Configuring Traffic Protection for Third-Party Applications, on page 2](#)
- [Troubleshooting Traffic Protection for Third-Party Applications, on page 3](#)

gRPC Protocol

Google-defined Remote Procedure Calls (gRPC) is an open-source RPC framework. It is based on Protocol Buffers (Protobuf), which is an open source binary serialization protocol. gRPC provides a flexible, efficient, automated mechanism for serializing structured data, like XML, but is smaller and simpler to use. The user needs to define the structure by defining protocol buffer message types in .proto files. Each protocol buffer message is a small logical record of information, containing a series of name-value pairs.

Cisco gRPC Interface Definition Language (IDL) uses a set of supported RPCs such as get-config, merge-config, replace-config, cli-config, delete-config, cli-show, get-models, action-json, commit, and commit-replace.

gRPC server runs in Extensible Manageability Services Daemon (emsd) process. gRPC client can be on any machine.

gRPC encodes requests and responses in binary. gRPC is extensible to other content types along with Protobuf. The Protobuf binary data object in gRPC is transported over HTTP/2.

**Note**

It is recommended to configure TLS before enabling gRPC. Enabling gRPC protocol uses the default HTTP/2 transport with no TLS enabled on TCP. gRPC mandates AAA authentication and authorization for all gRPC requests. If TLS is not configured, the authentication credentials are transferred over the network unencrypted. Non-TLS mode can only be used in secure internal network.

gRPC supports distributed applications and services between a client and server. gRPC provides the infrastructure to build a device management service to exchange configuration and operational data between a client and a server. The structure of the data is defined by YANG models.

Limitations for Traffic Protection for Third-Party Applications

The following limitations are applicable for the Traffic Protection for Third-Party Applications:

- If the TPA entry is configured with only the active RP management interface and redundancy switchover is performed, the gRPC connection fails.

Prerequisites for Traffic Protection for Third-Party Applications Over GRPC

Ensure that the gRPC is configured.

gRPC Configuration

```
Router(config)# grpc port port-number
Router(config)# grpc no-tls
Router(config-grpc)# commit
```

Running Configuration

```
Router# show running-config grpc

grpc port 57600
no-tls
!
```

Configuring Traffic Protection for Third-Party Applications

The following task shows how to configure traffic protection for third-party applications

```
RP/0/0/CPU0:ios#configure
RP/0/0/CPU0:ios(config)#tpa
RP/0/0/CPU0:ios(config-tpa)#vrf default
```

```
RP/0/0/CPU0:ios(config-tpa-vrf)#address-family ipv4
RP/0/0/CPU0:ios(config-tpa-vrf-afi)#protection
RP/0/0/CPU0:ios(config-tpa-vrf-afi-prot)#allow protocol tcp local-port port-number
remote-address IP remote address interface interface-name local-address IP local address
```

Running Configuration

```
Router# show running-config
tpa
vrf default
address-family ipv4
protection
allow protocol tcp local-port 57600 remote-address 10.0.0.2/32 local-address 192.168.0.1/32
allow protocol tcp local-port 57600 remote-address 10.0.1.0/24 local-address 192.168.0.1/32
allow protocol tcp local-port 57600 remote-address 10.0.2.0/24 local-address 192.168.0.1/32
address-family ipv6
protection
allow protocol tcp local-port 57600 remote-address 2001:DB8::1/128 local-address
2001:DB8:0:ABCD::1/128
allow protocol tcp local-port 57600 remote-address 2001:DB8::2/128 local-address
2001:DB8:0:ABCD::1/128
allow protocol tcp local-port 57600 remote-address 2001:DB8::3/128 local-address
2001:DB8:0:ABCD::1/128
!
!
!
!
```

Troubleshooting Traffic Protection for Third-Party Applications

The following show command output verifies whether TPA is configured or not.

```
Router# show running-config grpc

grpc
no-tls
!
```

The following show command output displays the TPA configuration.

```
Router# show running-config tpa

tpa
vrf default
address-family ipv4
allow local-port 57600 protocol tcp inter mgmtEth 0/RP0/CPU0/0 local-address
192.168.0.1/32 remote-address 10.0.0.2/32
!
```

gRPC Configuration without TPA

```
Router# show kim lpts database
```

```
State:
Prog - Programmed in hardware
Cfg - Configured, not yet programmed
Ovr - Not programmed, overridden by user configuration
Intf - Not programmed, interface does not exist
```

```
Owner  AF Proto State          Interface      VRF          Local ip,port > Remote ip,port
----- -- -- -- --
```

```
Linux 2 6 Prog global-vrf any,57600
> any,0
```

```
Router# show lpts bindings brief | include TPA
0/RP0/CPU0 TPA LR IPV4 TCP default any any,57600 any
```

gRPC Configuration with TPA

The following show command output displays the things that are configured in the LPTS database. It also checks if gRPC configuration is owned by Linux without using any filters.

```
Router# show kim lpts database
```

```
State:
```

```
Prog - Programmed in hardware
Cfg - Configured, not yet programmed
Ovr - Not programmed, overridden by user configuration
Intf - Not programmed, interface does not exist
```

Owner	AF	Proto	State	Interface	VRF	Local ip,port	>	Remote ip,port
Client	2	6	Prog		default	192.168.0.1/32,57600	>	10.0.0.2/32,0
Linux	2	6	Ovr		global-vrf	any,57600	>	any,0

```
Router# show lpts bindings brief | include TPA
```

```
0/RP0/CPU0 TPA LR IPV4 TCP default Mg0/RP0/CPU0/0 192.168.0.1,57600 10.0.0.2
```

```
Router#
```

```
Router# 0/RP0/ADMIN0:Mar 19 15:22:26.837 IST: pm[2433]:
```

```
%INFRA-Process_Manager-3-PROCESS_RESTART : Process tams (IID: 0) restarted
```