

Deep Packet Inspection and Inline Services

- Feature Summary and Revision History, on page 1
- Feature Description, on page 2
- How it Works, on page 4
- Supported Inline Services, on page 7
- Configuring the Static and Pre-Defined Rules, on page 63
- Configuring ACS Ruledef for L7 Protocols for DPI, on page 64
- Charging Action Configuration for L7 Protocols for DPI, on page 66

Feature Summary and Revision History

Summary Data

Applicable Product(s) or Functional Area	5G-UPF
Applicable Platform(s)	VPC-SI SMI
Feature Default Setting	Disabled – Configuration Required
Related Changes in this Release	Not Applicable
Related Documentation	*UCC 5G UPF Configuration and Administration Guide*

Revision History

Revision Details	Release
• NAT Support on the N4 interface has been added. • Subscriber Firewall support in a TCP Idle Timeout has been added. • Support added for capturing data stall issues using 2 new attributes.	2023.04.0

Revision Details	Release
Support has been added for the following functionality: • IP Readdressing • RTP Dynamic Flow Detection • Rule-matching for Bearer-specific Filters • QUIC IETF implementation	2021.02.0
New L7 protocols have been introduced as part of Deep Packet Inspection (DPI).	2021.01.0
The following EDR attributes have been added for TCP: • SYN and SYN-ACK packet • SYN-ACK and ACK packet	2021.01.0
New DNS attributes have been introduced in EDRs.	2021.01.0
First introduced.	2020.02.0

Feature Description

One of the key product capability of Cisco 5G-UPF is integrated Deep Packet Inspection (DPI) based services. DPI is the examination of layer 7 (L7), which contains Uniform Resource Identifier (URI) information. In some cases, layer 3 (L3) and layer 4 (L4) analyzers that identify a trigger condition are insufficient for billing purposes, so layer 7 (L7) examination is used.

DPI performs packet inspection beyond L4 inspection and is typically deployed for detection of URI information at L7 (for example, DNS, HTTP, HTTPS, RTP, and RTSP URLs).

DNS Server Readdressing

Whenever you use an unauthorized DNS server, you can modify the request to readdress the DNS IPs to use authorized servers. A ruledef determines if a packet belongs to a DNS query. It also determines if the DNS query belongs to a set of authorized DNS servers. If the DNS query does not belong to the authorized DNS servers, the flow action picks up DNS servers from the readdress server list.

You can configure the **readdress-server-list** command under active-charging service. When the flow matches a **ruledef**, you can configure the flow action to use the servers from **readdress-server-list**.

To configure the readdress server list under active-charging service, use the following configuration:

```
configure
  active-charging service service_name
    readdress-server-list name_of_list
      server ipv4_address [ port ]
      server ipv6_address [ port ]
```

Note You can configure a maximum number of 10 servers in a readdress server list and a maximum of 10 readdress server lists under active-charging service. Both IPv4 and IPv6 addresses can be configured in the same **readdress-server-list**.

Select the **readdress-server-list** from the list using one of the following methods:

- Round-robin—Server selection occurs in a round-robin manner for every new flow. Inactive servers in the list are not considered during the selection.

  This method is the default selection.

- Hierarchy—The servers that are tagged in this approach are primary, secondary, tertiary, and so on, depending on the order they are defined in the **readdress-server-list**. All flows are readdressed to the primary server as long as it is available. If the primary server goes down, then flows are readdressed to the secondary server and the same logic recurs. Once the primary server is active, then flows switch back to the primary server for readdressing.

To configure the DNS readdress server list, use the following CLI configuration under active-charging service.

```
configure
  active-charging service service_name
    readdress-server-list name_of_list
      server ipv4_address [ port ]
      server ipv6_address [ port ]
      consecutive-failures integer_value
      response-timeout integer_value
      reactivation-time integer_value

      charging-action action_name
        flow action readdress server-list name_of_list
        exit
```

NOTES:

- **consecutive-failures**—Specify an integer ranging from 1–10. The default value is *5*.
- **response-timeout**—Specify an integer ranging from 1–10000 milliseconds. The default value is *1000*.
- **reactivation-time**—Specify an integer ranging from 1–1800 seconds. The default value is *300*.

Readdress Server States

This section describes the readdress server states:

- Active state—Once configured, all servers are marked as Active.
- Inactive state—If no response is received from the readdressed server, then the server is marked as Inactive.
- Active-Pending state—Once the server is in Active-Pending state, it is available to accept the requests for readdressing. In this state, if a request is readdressed to this server and response is returned from it, then the server state is changed to Active. Otherwise, it is moved back to Inactive state.

How it Works

This section describes the following functionality of DPI:

- DSCP Marking of downlink and uplink packets.
- Traffic Readdressing or Redirecting.

DSCP Marking for Downlink and Uplink Packets

Table 1: Feature History

Feature Name	Release Information	Feature Description
EDR Attributes for DSCP Mapping	2024.02.0	UPF supports two new EDR attributes **sn-dscp-uplink** and **sn-dscp-downlink** to mark the user plane traffic using the dscp value received from SMF or DSCP(derived from TOS) configured under the charging-action. These attributes allow the users to understand the QoS flow information of the uplink and downlink traffic in their network. **Default Setting**: Disabled — Configuration Required to Enable

Transport-level marking is the process of marking traffic at UPF with a Differentiated Services Code Point (DSCP) value. The transport-level marking, executed on per-QoS flow, is based on the mapping from 5QI and optional Allocation and Retention Policy (ARP) configuration from SMF.

SMF controls the transport-level marking by providing the DSCP in the ToS (IPv4) or Traffic Class (IPv6) within the Transport Level Marking IE in the FAR, that is associated to the PDR matching the traffic to be marked. UPF performs the transport level marking for detected traffic and sends the marked packet to the peer entity. The SMF can change the transport-level marking by changing the Transport Level Marking IE in the related FAR.

UPF supports the following IEs:

- Inner Packet Marking IE—Marks the tunnel packets for inner packet marking.
- Transport Level Marking Options IE—Copies DSCP of the inner packet to the outer IP header.

EDR Attributes

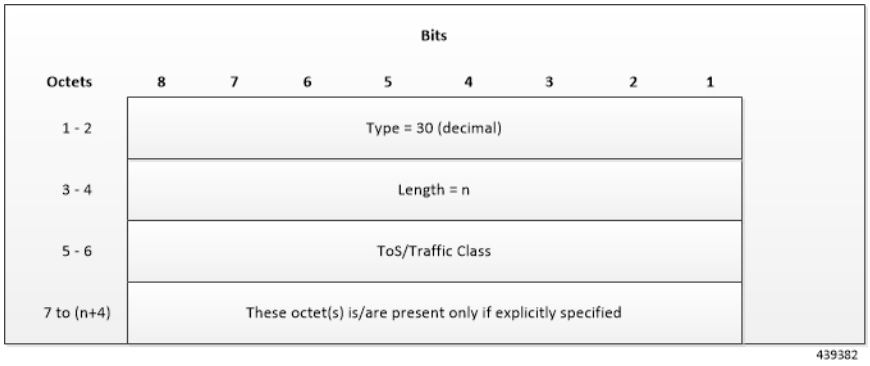
UPF records the DSCP values marked on the last outgoing traffic using the EDR attributes **sn-dscp-uplink** and **sn-dscp-downlink**.

UPF derives the DSCP values from the **ip tos** configuration under ACS charging-action or Transport Level Marking AVP received in FAR from SMF. The DSCP value from the IP ToS configuration takes higher precedence over that from the Transport Level Marking AVP.

For more information, see the *Event Data Records* section.

Transport Level Marking IE

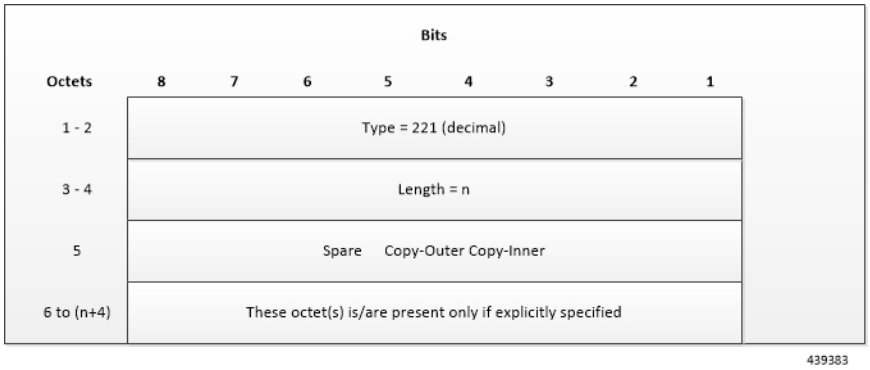
The "Transport Level Marking" IE type is encoded as shown in the following figure. It indicates the DSCP value for the downlink transport-level marking.



The encoding for Type-of-Service (ToS) or Traffic Class takes place in the form of two octets as an OctetString. The first octet contains the DSCP value in the IPv4 Type-of-Service or the IPv6 Traffic-Class field and the second octet contains the ToS/Traffic Class mask field, which is set to *0xFC*.

Transport Level Marking Options IE

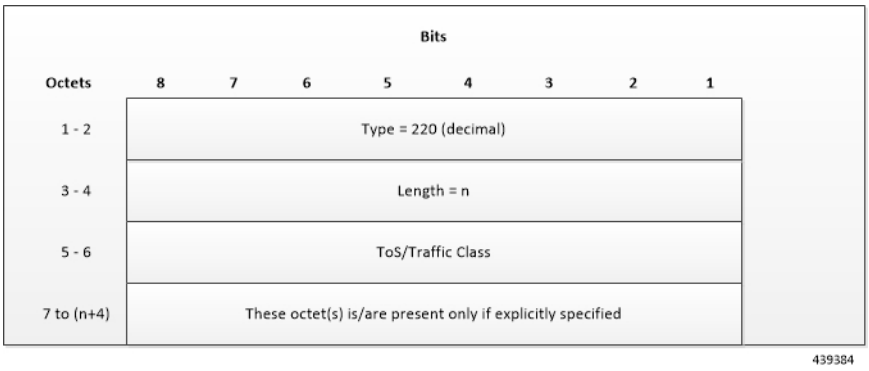
The "Transport Level Marking Options" IE type is encoded as shown in the following figure. The DSCP for downlink transport-level marking is copied from the inner packet.



The Copy-Inner and Copy-Outer flags are present in bit-0 and bit-1 of octet 5. Copy-Outer flag is not used for downlink packets because there is no outer header present in packets coming from ISP. If a Copy-Inner flag is present, then the UPF uses DSCP value from the inner packet to mark the transport-level IP header.

Inner Packet Marking IE

The "Inner Packet Marking" IE type is encoded as shown in the following figure. It indicates the DSCP value for the downlink inner packet marking.



The encoding for ToS/Traffic Class takes place in the form of two octets as an OctetString. The first octet contains the DSCP value in the IPv4 ToS or the IPv6 Traffic Class field and the second octet contains the ToS/Traffic Class mask field, which is set to *0xFC*.

NOTES:

- The original ECN bits in the IP header of User Plane packets do not change after applying transport-level marking or inner packet marking.
- If "Transport Level Marking" IE, "Inner Packet Marking" IE, or both the IEs are associated with uplink FAR, then the following rule applies for uplink packet marking:
  - If "Transport Level Marking" or "Inner Packet Marking" IE is present, its DSCP value is used.
  - If both "Transport Level Marking" and "Inner Packet Marking IE" are present, then the value from "Transport Level Marking" IE is used for uplink packet marking.

Traffic Readdressing or Redirecting

Traffic Redirection is the process of redirecting uplink application traffic to a redirect destination. For example, redirect some HTTP flows to service provisioning page. The redirect destination is provided by the PCF or it is preconfigured in the SMF or in the UPF.

The traffic redirection enforcement is applicable for the SMF or in the UPF if the traffic that the UPF supports subjects to traffic redirection. The UPF reports to the SMF whether it supports traffic redirection enforcement in the UPF through the "UP-Function Features" IE.

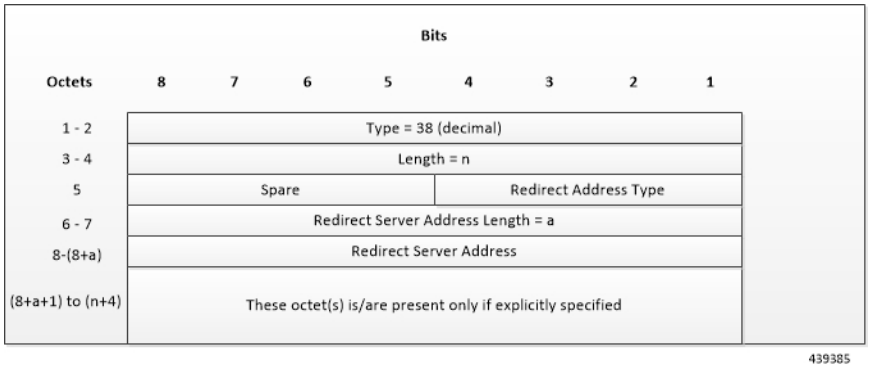
To enforce the traffic redirection in the UPF, the SMF takes the following actions:

- Creates the necessary PDRs, if it does not exist, to represent the traffic redirection.
- Creates a FAR with:
  - The "Redirect Information" IE, that includes the redirect destination, if the traffic needs redirection toward a redirect destination that is provided by the SMF. The redirect destination from the SMF prevails over a redirect destination that is preconfigured in the UPF.
    - For HTTP traffic redirection, the Redirection Address Type is set to "URL" and the SMF sets the "Destination Interface" IE in the FAR to "Access" (to forward the HTTP Response message with a status-code indicating redirect). For other types of traffic redirection, the "Destination Interface" IE in the FAR is set to "Core".

- Associates the FAR to the above PDRs of the PFCP session.

Redirect Information IE

Redirect information is encoded as follows:



"Redirect Address Type" indicates the type of the redirect address:

Redirect Address Type	Value (Decimal)
IPv4 address	0
IPv6 address	1
URL	2
SIP URI	3
Spare, for future use.	4–15

The "Redirect Server Address Length" indicates the length of the "Redirect Server Address". The "Redirect Server Address" encoding is in UTF8String format and contains the address of the redirect server (for example, HTTP redirect server, SIP server) with which the end user connects.

Important In this release, only Redirect Address Type URL is supported for dynamic rule when FAR is associated with URR where quota expires.

Supported Inline Services

Application Detection and Control

The ADC in-line service is used to detect Peer-to-Peer protocols by analyzing traffic. Other popular applications that generate the bulk of Internet traffic like Social Networking and Gaming applications can be detected.

The in-line service known as ADC is also referred as "P2P". Peer to Peer (P2P) is a term used in two slightly different contexts. At a functional level, it means protocols that interact in a peering manner, in contrast to client/server manner. There is no clear differentiation between the function of one node or another. Any node can function as a client, a server, or both—a protocol may not clearly differentiate between the two. For

example, peering exchanges may simultaneously include client and server functionality, sending and receiving information.

Note The ADC support is a licensed feature. Contact your Cisco Account or Support representative for information on how to obtain a license.

QUIC IETF Implementation

In the current framework, Deep Packet Inspection (DPI) is done for every packet in a flow when it reaches the plugin. The DPI is done by analyzing the packets and extracting deterministic patterns. The DPI is done in-order to detect the application and to classify its subtype. Plugin excludes the flow after the DPI. The flow is offloaded after the detection. As part of QUIC IETF, the initial QUIC handshake packets (Client/Server Hello) are encrypted over the network. Hence, there are no deterministic patterns available for detection of the application. Support is added in p2p plugin to decrypt and obtain the SNI (Server Name Indication) for detection.

Configuring QUIC IETF

Use the following configuration to enable or disable the QUIC IETF decryption.

```
configure
  active-charging service acs_service_name
    p2p-detection debug-param protocol-param p2p_quic_ietf_decrypt 1
    end
```

Note

- By default, the CLI is disabled and there's minimal impact on the performance due to TLS decryption.
- Runtime change of configuration doesn't impact the existing flow. Change is applicable only for new flow.

Statistics

show user-plane-service statistics analyzer name p2p

Use this show CLI command to determine the packets that are analyzed for QUIC and application.

SCONE support

Table 2: Feature history

Feature name	Release	Feature description
SCONE protocol for UPF	2026.03.0	This release introduces support for the SCONE protocol in the User Plane Function (UPF), enabling network-aware throughput optimization for QUIC-based applications. SCONE allows the UPF to provide explicit throughput advice to SCONE-capable endpoints by updating a rate signal in QUIC SCONE packets, helping applications adapt to network-enforced bandwidth policies without excessive congestion probing.

Modern applications that use QUIC often cannot determine the throughput constraints enforced by the network. As a result, congestion control algorithms continue probing for additional bandwidth, which can lead to unnecessary packet loss, increased latency, and inefficient bandwidth utilization. SCONE (Standard Communication with Network Elements) enables the UPF to communicate throughput advice to SCONE-capable QUIC endpoints by modifying a 7-bit rate signal field in QUIC transport parameters. This allows applications to optimize their transmission rate based on network policies rather than relying solely on congestion detection, making network-enforced throughput policies visible and reducing unnecessary congestion probing.

SCONE provides the following benefits:

- Makes network-enforced throughput policies visible to QUIC endpoints.
- Reduces unnecessary congestion probing performed by congestion control algorithms.
- Enables applications to optimize transmission rates based on network guidance instead of packet loss or delay.
- Improves bandwidth utilization by communicating explicit throughput advice to SCONE-capable endpoints.
- Enables network-aware traffic optimization without requiring application-specific modifications.

SCONE Operational Flow

SCONE processing consists of the following stages:

- **Flow Detection:** ADC-P2P identifies SCONE-capable QUIC flows by inspecting the QUIC Transport Parameters for the `scone_supported` transport parameter (ID `0x219e`) and verifying the SCONE indication signature (`0xC8 0x13`) in the QUIC Initial packet. If both identifiers are present, the flow is marked as SCONE-capable for its lifetime. Detection is performed only during flow establishment and is not applied to existing flows when SCONE is enabled mid-session.

- **Rule Matching:** The `scone-detected` ruledef keyword allows operators to distinguish SCONE-capable flows from other traffic and apply dedicated charging actions based on SCONE detection status.
- **Rate Signal Calculation:** The UPF determines the throughput advice using either an explicitly configured `ta-value` or the most restrictive value between the flow Maximum Bit Rate (MBR) and the APN-AMBR. For dynamic rules, the MBR is obtained from the QER associated with the matching PFCP PDR. The calculated bitrate is encoded into a 7-bit SCONE Rate Signal.
- **Rate Signal Modification:** For every downlink SCONE packet, the VPP validates the SCONE packet format, extracts the existing Rate Signal, and compares it with the value calculated by the UPF. The packet is modified only when the calculated value is lower than the Rate Signal already present in the packet. After modification, the UDP checksum is recalculated before forwarding the packet.
- **EDR Reporting:** When configured, the UPF reports SCONE-specific information in the Charging EDR, including the SCONE detection status (`scone-enabled`) and the last Rate Signal value (`scone-rate-signal`) applied to the flow.

Configure SCONE support

SCONE enables the UPF to detect SCONE-capable QUIC flows and communicate throughput advice to supported QUIC endpoints based on configured network policies.

Before you begin

- Ensure that the UPF is running a software release that supports the SCONE feature.
- Enable QUIC detection in the ADC service.
- Enable P2P dynamic flow detection on the rulebase before configuring SCONE.
- Ensure that the required rulebase, ruledefs, and charging actions are created.

Follow these steps to configure SCONE support:

Procedure

Step 1 Enable SCONE detection in the ADC service:

```
p2p-detection protocol all quic
p2p-detection debug-param protocol-param p2p_quic_ietf_decrypt 1
p2p-detection debug-param protocol-param p2p_quic_ietf_frag_detection 1
```

Step 2 Enable P2P dynamic flow detection on the rulebase.

```
rulebase <rulebase_name>
p2p dynamic-flow-detection
```

Step 3 Enable SCONE processing on the rulebase.

```
rulebase <rulebase_name>
scone-enabled
or
rulebase <rulebase_name>
scone-enabled all
```

Step 4 Configure SCONE matching in the ruledef.

```
ruledef <ruledef_name>
scone-detected
```

Step 5 Configure SCONE throughput advice in the charging action.

```
charging-action <charging_action_name>
scone-throughput-advice enabled
or
charging-action <charging_action_name>
scone-throughput-advice enabled ta-value <bps>
or
charging-action <charging_action_name>
scone-throughput-advice disabled
```

The UPF detects SCONE-capable flows during flow establishment and signals throughput advice using either the configured `ta-value` or the bitrate derived from the most restrictive value between the flow MBR and APN-AMBR, depending on the charging-action configuration. The throughput advice is applied to eligible downlink SCONE traffic.

SCONE monitoring and troubleshooting

Monitor SCONE flow status and troubleshoot configuration issues using the following EDR attributes and show commands.

EDR attributes

Configure the following EDR attributes to monitor SCONE flow status and throughput advice information.

- **scone-enabled:** Indicates whether the flow was identified as SCONE-capable. Possible values are **y** and **n**.
- **scone-rate-signal:** Reports the last rate signal value applied to the flow. Possible values include a numeric rate signal, **bypass**, **ta-disable**, and **n/a**.

Show commands

Use the following commands to verify configuration and monitor SCONE statistics:

Command	Recommended Purpose
`show user-plane-service statistics all`	Displays node-level SCONE statistics.
`show subs user-plane-only flows`	Displays subscriber flow information, including SCONE status and Rate Signal.
`show subscribers user-plane-only full all`	Displays number of SCONE flows in the detailed session information for all subsribers.
`show subscribers user-plane-only callid <call-id>`	Displays number of SCONE flows.
`show subscribers user-plane-only callid <call-id> flow-id <flow-id>`	Displays SCONE information for the specified subscriber flow.
`show user-plane-service rulebase name <name>`	Displays SCONE configuration for the specified rulebase.
`show user-plane-service ruledef name <name>`	Displays SCONE configuration for the specified ruledef.

Command	Recommended Purpose
`show user-plane-service charging-action name <name>`	Displays SCONE throughput advice configuration for the specified charging action.

App-based Tethering Detection

Feature Description

The App-based Tethering Detection solution is built around the existing ADC plugins for App identifications. Tethering-specific patterns are added on top of recognized App plugins. These plugins successively return if the App flow is tethered or not. The App based Tethering Detection interworks with other existing supported tethering technique.



Note App-based Tethering Detection feature supports detection of only Netflix and YouTube applications.

Limitation

If there are any new TLS patterns used by tethered devices in the network, then those are not identified for tethering detection.

Configuring App-based Tethering Detection

This section describes how to enable support for App-based Tethering Detection.

Enabling App-based Tethering Detection at Rulebase Level

Use the following commands to enable App-based Tethering Detection for ADC traffic under ACS Rulebase Configuration Mode:

```
configure
  active-charging service service_name
    rulebase rulebase_name
      tethering-detection application
      exit
    exit
  exit
```

NOTES:

- If previously configured, use the **no tethering-detection** command to remove the tethering detection configuration from the rulebase.

App-based Tethering Detection at Ruledef Level

The following configuration allows you to define rule expressions to match tethered or non-tethered flows..

```
configure
  active-charging service service_name
    ruledef ruledef_name
     tethering-detection application { flow-tethered | flow-not-tethered
```

```
}
      exit
    exit
  exit
```

NOTES:

- If previously configured, use the **no tethering-detection** command to remove the tethering detection configuration from the ruledef.

Monitoring and Troubleshooting the App-based Tethering Detection

This section provides information regarding commands available to monitor and troubleshoot the App-based Tethering Detection.

Show Commands and Outputs

This section provides information regarding show commands and their outputs in support of this feature.

show user-plane-service statistic tethering-detection

The output of this CLI command has been enhanced to include the following fields in support of this feature.

- Tethering Detection Statistics (Application):
  - Total flows scanned
  - Tethered flows detected
  - Tethered uplink packets
  - Tethered downlink packets

show user-plane-service statistic rulebase name <rulebase_name>

The output of this CLI command has been enhanced to include the following fields in support of this feature.

- Tethering Detection (Application):
  - Total flows scanned
  - Tethered flows detected
  - Tethered uplink packets
  - Tethered downlink packets

Content Filtering

Content Filtering is an in-line service available for 3GPP and 3GPP2 networks to filter HTTP requests from mobile subscribers based on the URLs in the requests. This enables operators to filter and control the content that an individual subscriber can access, so that subscribers are inadvertently not exposed to universally unacceptable content and/or content inappropriate as per the subscribers' preferences.

Content Filtering Configuration

Use the following additional configuration to enable the content filtering:

```
configure
  require user-plane content-filtering
    content-filtering category database directory path path_address
    content-filtering category database max-version version_number
    end
```



Note The above configuration must be configured on the UPF, during boot time, to enable Content Filtering. Defining the above configuration post the User Plane configuration will lead to errors and inconsistencies.

Show Commands Input and/or Outputs

show subscribers user-plane-only callid *call_id* full all

SMF provides Content Filtering Policy ID in the Session Establishment/Modification Request. The following fields are displayed in support of this feature:

- SUBSCRIBER PARAMS
  - Content Filtering Policy ID

DNS Snooping

Charging

The charging of DNS Snooping takes place at SM-P.

Rule Definitions

Use the following CLI commands for specifying the rule definition hostnames (domain-names) and part of the host names.

```
ruledef <ruledef_name>
      ip [server-domain-name {contains|=|ends-with|starts-with} <url_string>]

      ip [server-domain-name {contains|=|ends-with|starts-with} <url_string>]

       multi-line-OR enabled
```

Use the no version of this CLI to delete the ruleline for ip server- domain-name.

```
ruledef <ruledef_name>
  no ip [server-domain-name {contains|=|ends-with|starts-with} <url_string>]

   exit
```

Use the following CLI for configurable timer of DNS entries at ECS level.

```
configure
        active-charging service service_name
```

```
ip dns-resolved-entries timeout <value_secs>
end
```

Whenever the ruledef containing the ip server-domain-name keyword is defined and used in rulebase, the ip-table is created per rulebase per instance.

Rule Matching

The functionality remains the same as the non-CUPS architecture.

Show CLIs

Use the following CLIs to check the table for DNS IP entries:**show user-plane-service [statistics dns-learnt-ip-addresses {summary | sessmgr instance <id> |all [verbose] }]**

Bulkstats

The following bulkstats are available in support of DNS Snooping feature:

- ecs-dns-learnt-ipv4-entries
- ecs-dns-flushed-ipv4-entries
- ecs-dns-replaced-ipv4-entries
- ecs-dns-overflown-ipv4-entries
- ecs-dns-learnt-ipv6-entries
- ecs-dns-flushed-ipv6-entries
- ecs-dns-replaced-ipv6-entries
- ecs-dns-overflown-ipv6-entries

The above bulkstats are added in the ECS schema same as in the non-CUPS architecture.

Note The SNMP Trap generation commands are not supported in CUPS DNS snooping feature.

Event Data Records

Table 3: Feature History

Feature Name	Release Information	Feature Description
S-NSSAI Support in EDR	2024.02.0	UPF supports a new EDR attribute **sn-nssai** as part of Flow and Transaction EDRs in the EDR Format configuration. This attribute reports the S-NSSAI value that UPF receives from SMF in PFCP Session Establishment Request.

Feature Name	Release Information	Feature Description
EDR Last Uplink and Downlink Packet Time Attributes	2023.04	The EDR configuration supports two new attributes **sn-last-uplink-pkt-time** and **sn-last-downlink-pkt-time** to identify the data stall issue.

Feature Description

Event Data Records (EDR) are usage records with support to configure content information, format, and generation triggers by the system administrative user.

When a flow is terminated, the UPF generates EDRs with detail information of the terminated flow.

How It Works

EDRs are generated from the User Plane on flow termination. During call setup and call modification, all call-specific attributes that are required for EDR generation is sent from SMF to UPF as part of the "Subscriber Params" IE within the Sx Establishment/Modification request messages.

On flow termination, the charging counters are fetched from VPP. All configured call-level attributes in the EDR format configuration along with the charging/volume counter attributes is sent to the CDRMOD proclet. This proclet writes these records to a file/disk, which is transferred to a configured external server.

TCP Fast Open

TCP Fast Open (TFO) is an extension to speed up the opening of successive TCP connections between two endpoints. It works by using a TFO cookie (a TCP option), which is a cryptographic cookie that is stored on the client and set upon the initial connection with the server. When the client reconnects, it sends the initial SYN packet along with the TFO cookie data to authenticate itself. If successful, the server starts sending data to the client even before the reception of the final ACK packet of the three-way handshake. Due to this, the difference between the following packets are recorded to calculate and record the time difference between control packets of TCP flow in EDR:

- SYN and SYN-ACK packet
- SYN-ACK and ACK packet

For information about rule variables that are added to capture the information in EDRs, refer *Configuring Additional TCP Fields* section.

Transaction Complete EDR

Transaction Complete EDRs are generated for HTTP EDRs when an HTTP transaction is completed. On completion, the charging counters are fetched from VPP. All configured call-level attributes in the EDR format configuration along with the charging/volume counter attributes is sent to the CDRMOD proclet. This proclet writes these records to a file/disk, which is transferred to a configured external server.

The following EDR attributes are supported:

- sn-start-time
- sn-end-time
- sn-start-time format MM/DD/YYYY-HH:MM:SS:sss

- sn-end-time format MM/DD/YYYY-HH:MM:SS:sss
- radius-calling-station-id
- radius-called-station-id
- bearer 3gpp imsi
- bearer 3gpp imei
- bearer 3gpp rat-type
- bearer 3gpp user-location-information
- ip subscriber-ip-address
- ip dst-address
- sn-ruledef-name
- sn-subscriber-port
- sn-server-port
- sn-app-protocol
- sn-volume-amt ip bytes uplink
- sn-volume-amt ip bytes downlink
- sn-flow-start-time format seconds
- sn-flow-end-time format seconds
- sn-volume-amt ip pkts uplink
- sn-volume-amt ip pkts downlink
- sn-direction
- traffic-type
- p2p protocol
- p2p app-identifier tls-cname
- p2p app-identifier tls-sni
- p2p app-identifier quic-sni
- bearer 3gpp sgsn-address
- sn-rulebase
- sn-charging-action
- flow tethered-ip-ttl
- flow ttl
- flow ip-control-param
- bearer qci

- tcp flag
- ip server-ip-address
- sn-flow-id
- sn-closure-reason
- sn-duration
- ip src-address
- ip protocol
- sn-charge-volume ip bytes uplink
- sn-charge-volume ip bytes downlink
- sn-nssai
- sn-dscp-uplink
- sn-dscp-downlink

The following HTTP EDR attributes are supported:

- http url length 2000
- http request method
- http content type
- http user-agent length 255
- http reply code
- http referer
- http host
- http cookie
- http header-length
- transaction-uplink-bytes
- transaction-downlink-bytes

The following DNS EDR attributes are supported:

- dns answer-ip-list
- dns answer-name
- dns previous-state
- dns query-name
- dns query-type
- dns return-code

- dns state
- dns tid

EDR Attributes to Identify Data Stalling

EDR supports the following attributes to identify the data stalling issue:

- **sn-last-uplink-pkt-time**
- **sn-last-downlink-pkt-time**

When the average difference is higher between the last packet on uplink or downlink direction and **sn-end time** (flow release time), it indicates data stalling.

These EDR variables do not capture the packet time of the control packets (FIN/RST/FIN-ACK/TCP TEAR DOWN ACK). The packet time for these packets is captured at the session manager using **sn-end time**.

For the offloaded flows, the attributes **sn-last-uplink-pkt-time** and **sn-last-downlink-pkt-time** is updated for packets going through the fastpath.

Limitations

The EDR feature in UPF has the following limitations:

- EDR will be generated only for flow end condition: Idle timeout, HAGR, normal flow termination, and during the end of a session.
- Charging-Action based EDR configuration is not supported.
- Reporting EDRs are not supported.
- The packet time of the control packet will not be updated for last_uplink_packet_time and last_downlink_packet_time except mid session TCP ACKs in EDR.

Configuring Event Data Records

Configuring EDRs on UPF

Use the following configuration to configure EDRs on UPF:

```
active-charging service service_name
   rulebase rulebase_name
      flow end-condition { timeout | normal-end-signaling | session-end |
 hagr } charging-edr charging_edr_format_name
     edr transaction-complete { http | dns } charging-edr
charging_edr_format_name
     exit
     edr-format format_name
        attribute attribute_name format time_format priority priority_val
```

NOTES:

- **flow end-condition**—This command allows you to configure the end condition of the session flows related to a user session and triggers EDR generation.
- **timeout**—Creates an EDR with the specified EDR format whenever a flow ends due to a timeout condition.

- **normal-end-signaling**—Creates an EDR with the specified EDR format whenever the flow end is signaled normally. For example, detecting FIN and ACK for a TCP flow, and create an EDR for the flow using the specified EDR format.
- **session-end**—Creates an EDR with the specified EDR format whenever a subscriber session ends. By this option, the session manager creates an EDR with the specified format name for every flow that has had any activity since the last EDR was created for the flow on session end.
- **charging-edr** *charging_edr_format_name*—Specifies the charging EDR format.
- **hagr**—Creates an EDR with the specified EDR format whenever a flow is terminated due to Inter-chassis Session Recovery action.
- **http**—Specifies HTTP protocol-related configuration.
- **dns**—Specifies DNS protocol-related configuration.
- **edr-format** *format_name*—Specifies an EDR that has to be configured.
- **attribute** *attribute_name* **format** *time_format* **priority** *priority_val*—Specifies the specific attribute to capture in EDR. **format** *time_format* specifies the timestamp format. It supports the following time formats:
  - MM/DD/YY-HH:MM:SS
  - MM/DD/YY-HH:MM:SS:sss
  - MM/DD/YYYY-HH:MM:SS
  - MM/DD/YYYY-HH:MM:SS:sss
  - YYYY/MM/DD-HH:MM:SS
  - YYYY/MM/DD-HH:MM:SS:sss
  - YYYYMMDDHHMMSS
  - YYYYMMDDHHMMSSsss
  - Seconds

  priority *priority_val* specifies specifies the position priority of the value within the EDR record. It must be an integer from 1 through 65535.

Configuration to Enable EDR Module

Use the following configuration to enable EDR module.

```
configure
  context context_name
    edr-module active-charging-service
    end
```

Configuring Additional TCP Fields

Prior to using the following CLI commands to configure additional TCP fields in the EDR, ensure that all the other EDR configurations are present.

```
configure
  active-charging service service_name
    edr-format edr_format_name
      rule-variable tcp syn-synack-rtt priority priority_value
      rule-variable tcp synack-ack-rtt priority priority_value
      end
```

Enabling EDR Billing

Use the following configuration for the system to generate an EDR immediately when a packet is received and it matches a ruledef that is associated with this charging action.

```
configure
  active-charging service service_name
    charging-action charging_action_name
      billing-action edr edr_format_name
      end
```

Configure the purging of pending DNS records

Follow these steps to configure to purging the pending DNS records.

Procedure

Step 1 Enter the Active Charging Service configuration mode.

```
configure
   active-charging service service_name
```

Step 2 Configure Maximum pending DNS records.

max-pending-dns-records *number_of_records*

The default value of **max-pending-dns-records** is 200.

Example:

```
[local]laas-si-setup#configure
[local]laas-si-setup(config)#active-charging service ACS
[local]laas-si-setup(config-acs)#max-pending-dns-records 890
```

Step 3 Verify the configuration.

Example:

```
[local]laas-si-setup# show configuration active-charging service name ACS verbose
config
  active-charging service ACS
    max-pending-dns-records 200


[local]laas-si-setup# show configuration active-charging service name ACS
config
  active-charging service ACS
```

```
#exit
max-pending-dns-records 890
```

Monitoring and Troubleshooting

show user-plane-service statistics rulebase name *rulebase_name*

The following fields are displayed in support of this feature:

- Rulebase Name
  - EDRs
  - Charge Volume
    - Uplink Pkts
    - Uplink Bytes
    - Downlink Pkts
    - Downlink Bytes
  - Charging EDRs
    - Total Charging EDRs generated
    - EDRs generated for handoff
    - EDRs generated for timeout
    - EDRs generated for normal-end-signaling
    - EDRs generated for session end
    - EDRs generated for rule match
    - EDRs generated for hagr
    - EDRs generated for flow-end content-filtering
    - EDRs generated for flow-end url-blacklisting
    - EDRs generated for content-filtering
    - EDRs generated for url-blacklisting
    - EDRs generated for any-error packets
    - EDRs generated for firewall deny rule match
    - EDRs generated for transaction completion
    - EDRs generated for voip call end
    - EDRs generated for dcca failure handling
    - EDRs generated for TCP optimization on
    - EDRs generated for tethering signature change

- EDRs generated for interim interval
- Total Flow-Overflow EDRs
- Total zero-byte EDRs suppressed

- Reporting EDRs
  - Total Charging REDRs generated
  - REDRs generated for handoff
  - REDRs generated for timeout
  - REDRs generated for normal-end-signaling
  - REDRs generated for session end
  - REDRs generated for rule match
  - REDRs generated for hagr
  - REDRs generated for flow-end content-filtering
  - REDRs generated for flow-end url-blacklisting
  - REDRs generated for content-filtering
  - REDRs generated for url-blacklisting
  - REDRs generated for any-error packets
  - REDRs generated for firewall deny rule match
  - REDRs generated for transaction completion
  - REDRs generated for voip call end
  - REDRs generated for dcca failure handling
  - REDRs generated for TCP optimization on
  - REDRs generated for tethering signature change

show user-plane-service edr-format all

The following fields are displayed in support of Additional TCP Fields in EDR feature:

- Service Name
  - EDR Format Name
    - rule-variable tcp syn-synack-rtt priority 1
    - rule-variable tcp synack-ack-rtt priority 2

Firewall Support

Table 4: Feature History

Feature Name	Release Information	Description
Firewall TCP Idle Timeout Action	2023.04	The Firewall feature inspects subscriber traffic performing IP session-based access control to protect subscribers from security attacks. UPF supports the TCP Idle Timeout action to drop the subscriber flow or send reset on TCP timeout expiry.

Feature Description

The Firewall feature in UPF 2.0 inspects subscriber traffic and performs IP session-based access control of individual subscriber sessions to protect the subscribers from malicious security attacks. UPF 2.0 supports the TCP Idle Timeout action to drop the subscriber flow or send reset on TCP timeout expiry.

The firewall configuration allows the system to inspect each packet of the subscriber data session. It also evaluates the security threat and applies the policies configured on uplink and downlink traffic. Firewall supports validation at per flow-level and per packet.

The Firewall feature supports the following functionality:

- Protection against DoS and DDoS attacks
- Application-level Gateway
- Stateful Packet Inspection and Filtering
- Stateless Packet Inspection and Filtering
- SNMP Thresholding

Subscriber Firewall Configuration

The firewall configurations are available under active charging configuration in UPF:

- Access-Ruledefs
- Firewall-NAT Policy

Enabling Default Firewall for IPv4 and IPv6

To configure the default firewall for IPv4 and IPv6, use the following configuration:

```
configure
  active-charging service service_name
    fw-and-nat policy policy_name
```

```
firewall policy ipv4-and-ipv6
end
```

Configuring TCP Idle Timeout Action

To configure the TCP Idle timeout action, use the following configuration:

```
configure
  active-charging service service_name
    fw-and-nat policy policy_name
      firewall tcp-idle-timeout-action { drop | reset }
      end
```

NOTES:

- **firewall tcp-idle-timeout-action { drop | reset }**—Specify the Stateful Firewall action to be taken on TCP idle timer expiry.
  - **drop**—Subscriber flow will be cleared or dropped without sending a reset on TCP timeout expiry.
  - **reset**—Specify to send a reset on TCP timeout expiry. This is the default value.
- The **firewall tcp-idle-timeout-action reset** CLI command is applicable only to firewall.
- UPF does not support flow mapping.

Along with the preceding service configuration, the following configuration is the default CLI behavior of various Firewall-related CLI within the service.

```
Dos-Protection:
      Source-Route                          : Disabled
      Win-Nuke                              : Disabled
      Mime-Flood                            : Disabled
      FTP-Bounce                            : Disabled
      IP-Unaligned-Timestamp                : Disabled
      Seq-Number-Prediction                 : Disabled
      TCP-Window-Containment                : Disabled
      Teardrop                              : Disabled
      UDP Flooding                          : Disabled
      ICMP Flooding                         : Disabled
      SYN Flooding                          : Disabled
      Port Scan                             : Disabled
      IPv6 Extension Headers Limit          : Disabled
      IPv6 Hop By Hop Options               : Disabled
      Hop By Hop Router Alert Option        : Disabled
      Hop By Hop Jumbo Payload Option       : Disabled
      Invalid Hop By Hop Options            : Disabled
      Unknown Hop By Hop Options            : Disabled
      IPv6 Destination Options              : Disabled
      Invalid Destination Options           : Disabled
      Unknown Destination Options           : Disabled
      IPv6 Nested Fragmentation             : Disabled

    Max-Packet-Size:
      ICMP                            : 65535
      Non-ICMP                        : 65535
    Flooding:
      ICMP limit                      : 1000
      UDP limit                       : 1000
      TCP-SYN limit                   : 1000
      Sampling Interval               : 1
```

```
TCP-SYN Flood Intercept:
  Mode                          : None
  Max-Attempts                  : 5
  Retrans-timeout               : 60
  Watch-timeout                 : 30
Mime-Flood Params:
  HTTP Header-Limit             : 16
  HTTP Max-Header-Field-Size    : 4096

No Firewall Ruledef Match Action:
  Uplink Action                 : permit
  Downlink Action               : deny

TCP RST Message Threshold     : Disabled
ICMP Dest-Unreachable Threshold    : Disabled
Action upon receiving TCP SYN packet with ECN/CWR Flag set    : Permit
Action upon receiving a malformed packet  : Deny
Action upon IP Reassembly Failure   : Deny
Action upon receiving an IP packet with invalid Options : Permit
Action upon receiving a TCP packet with invalid Options : Permit
Action upon receiving an ICMP packet with invalid Checksum: Deny
Action upon receiving a TCP packet with invalid Checksum: Deny
Action upon receiving an UDP packet with invalid Checksum: Deny
Action upon receiving an ICMP echo packet with id zero : Permit
TCP Stateful Checks : Enabled
First Packet Non-SYN Action: Drop
ICMP Stateful Checks: Enabled
TCP Partial Connection Timeout: 30
```

IP Reassembly

This section describes the reassembly functionality:

- In the non-CUPS architecture, fragments are buffered up to 64K bytes with the default Firewall configuration. All buffered and subsequent fragments are dropped beyond 64K bytes. In UPF, it is possible to reassemble the packet size to 9K bytes in a maximum of six fragments.
- The **firewall ip-reassembly-failure** CLI configures teardrop attack, nested fragmentation, and general ip-reassembly-failure. The maximum IP packet size is limited to six fragments (~9000 bytes).
- The following counters in firewall statistics get incremented for all attacks related to reassembly:
  - Downlink Packets Dropped due to IPv4 Reassembly Failure
  - Uplink Packets Dropped due to IPv4 Reassembly Failure
  - Downlink Dropped Bytes on IPv4 Reassembly Failure
  - Uplink Dropped Bytes on IPv4 Reassembly Failure
  - Downlink Packets Dropped due to IPv6 Reassembly Failure
  - Uplink Packets Dropped due to IPv6 Reassembly Failure
  - Downlink Dropped Bytes on IPv6 Reassembly Failure
  - Uplink Dropped Bytes on IPv6 Reassembly Failure

Monitoring and Troubleshooting

This section provides information about show commands and their output in support of this feature.

show config active-charging service name acs verbose

```
fw-and-nat policy SFW_NAT_TEST
      no firewall dos-protection source-router
      no firewall dos-protection winnuke
      no firewall dos-protection mime-flood
      no firewall dos-protection ftp-bounce
      no firewall dos-protection ip-unaligned-timestamp
      no firewall dos-protection tcp-window-containment
      no firewall dos-protection teardrop
      no firewall dos-protection flooding udp
      no firewall dos-protection flooding icmp
      no firewall dos-protection flooding tcp-syn
      no firewall dos-protection port-scan
      no firewall dos-protection ipv6-extension-hdrs
      no firewall dos-protection ipv6-hop-by-hop
      no firewall dos-protection ipv6-hop-by-hop router-alert
      no firewall dos-protection ipv6-hop-by-hop jumbo-payload
      no firewall dos-protection ipv6-hop-by-hop invalid-options
      no firewall dos-protection ipv6-hop-by-hop unknown-options
      no firewall dos-protection ipv6-dst-options
      no firewall dos-protection ipv6-dst-options invalid-options
      no firewall dos-protection ipv6-dst-options unknown-options
      no firewall dos-protection ipv6-frag-hdr nested-fragmentation
      no firewall dos-protection ip-sweep tcp-syn
      no firewall dos-protection ip-sweep udp
      no firewall dos-protection ip-sweep icmp
      firewall max-ip-packet-size 65535 protocol icmp
      firewall max-ip-packet-size 65535 protocol non-icmp
      firewall flooding protocol icmp packet limit 1000
      firewall flooding protocol udp packet limit 1000
      firewall flooding protocol tcp-syn packet limit 1000
      firewall flooding sampling-interval 1
      firewall tcp-syn-flood-intercept mode none
      firewall tcp-syn-flood-intercept watch-timeout 30
      firewall mime-flood http-headers-limit 16
      firewall mime-flood max-http-header-field-size 4096
      no firewall icmp-destination-unreachable-message-threshold
      access-rule no-ruledef-matches uplink action permit
      access-rule no-ruledef-matches downlink action deny
      firewall tcp-idle-timeout-action reset
      no firewall tcp-reset-message-threshold
      firewall tcp-syn-with-ecn-cwr permit
      firewall malformed-packets drop
      firewall ip-reassembly-failure drop
      no firewall validate-ip-options
      firewall tcp-options-error permit
      firewall icmp-echo-id-zero permit
      firewall icmp-checksum-error drop
      firewall tcp-checksum-error drop
      firewall udp-checksum-error drop
      firewall tcp-fsm first-packet-non-syn drop
      firewall icmp-fsm
      firewall policy ipv4-and-ipv6
      firewall tcp-partial-connection-timeout 30
      no nat policy
      no nat binding-record
      no nat pkts-drop edr-format
      no nat pkts-drop timeout
      default nat suppress-aaa-update
      nat private-ip-flow-timeout 180
      nat check-point-info basic limit-flows 100
      nat check-point-info sip-alg
      nat check-point-info h323-alg
```

```
  nat max-chunk-per-realm single-ip
#exit
```

Show CLIs

This section describes the supported show CLI commands for UPF:

- **show subscribers user-plane-only full all**
- **show subscribers user-plane-only flows**
- **show user-plane-service inline-services firewall statistics verbose**
- **show user-plane-service statistics rulebase all**
- **show alarm outstanding all**
- **show alarm outstanding all verbose**
- **show alarm statistics**
- **show user-plane-service statistics rulebase name** *rulebase_name*

SNMP Traps

This section describes the SNMP traps. To enable the trap, use the respective trap CLIs on the User Plane.

- **Dos-Attacks**—When the number of DoS attacks exceeds the set threshold value, the SNMP trap is generated. When the number falls below the threshold value within the configured time interval, the trap is cleared.
- **Drop-Packets**—When the number of packets dropped exceeds the threshold value, the SNMP trap is generated. When the number falls below the threshold value within the configured time interval, the trap is cleared.
- **Deny-Rule**—When the number of Deny Rules exceeds the threshold value, the SNMP trap is generated. When the number falls below the threshold value within the configured time interval, the trap is cleared.
- **No-Rule**—When the number of No Rules exceeds the threshold value, the SNMP trap is generated. When the number falls below the threshold value within the configured time interval, the trap is cleared.

Flow Idle Timeout Randomization

Every two seconds, the Session Manager polls the time of the latest packet from Session Manager instance, or the fastpath stream to determine idle flows. Short length flows become idle quickly as they are short due to the lesser number of packets and are short lived, within 5–10 seconds. As a result, large number of idle flows must be deleted due to the timeout at the given polling cycle of two seconds. Deletion of idle flows is CPU intensive as it involves statistics reconciliation, EDR generation, and fast path stream deletion. You can accomodate more flows with this feature as the short lived flows get cleared aggressively.

Configuring Flow Idle Timeout Randomization in ACS

Use the following configuration to randomize the idle timeout flow.

```
configure
active-charging service service_name
```

```
idle-timeout randomize-range range
  { default | no } idle-timeout randomize-range
    end
```

NOTES:

- **idle timeout**: Specifies the maximum duration that a flow can remain idle for, in seconds. Seconds must be an integer from 5 through 30. The flow will then be terminated based on the random value.
- **randomize-range**: Specifies the range of a period of time in seconds. The idle timeout applied, will be different for each flow.

  For example,

  `idle-timeout randomize-range 20`

  An integer random number is generated from 0 through 20. This number is added to the configured idle timeout value to check if the flow has become idle in the two second timer processing. If the idle timeout configured is 60 seconds, the actual timeout that is applied to each flow will be random in the range between *60 + 20* seconds causing staggered flow deletion.
- **no**: Disables the idle timeout randomization. This command is disabled by default.
- **default**: Configures the idle timeout randomization command with its default setting in seconds. Seconds must be an integer from 0 through 30. Default range is 0–30 seconds.

  For example, **default idle-timeout randomize-range** is equal to **idle-timeout randomize-range 30**.

HTTP URL Filtering

The HTTP URL Filtering feature simplifies rule definitions used for URL detection.

The HTTP request packet can have a proxy (prefixed) URL and an actual URL. If a proxy URL is found in the HTTP request packet, the HTTP URL Filtering feature truncates this URL from the parsed information and only the actual URL is used for rule matching and Event Data Records (EDR) generation.

Configuring the HTTP URL Filtering Feature

This section describes how to configure the HTTP URL Filtering feature.

Configuring Group of Prefixed URLs

To configure the group of prefixed URLs, use the following CLI commands:

```
configure
  active-charging service ecs_service_name
    group-of-prefixed-urls prefixed_urls_group_name
    end
```

Configuring URLs in the Group of Prefixed URLs

To configure URLs to be filtered in the group of prefixed URLs, use the following CLI commands:

```
configure
  active-charging service ecs_service_name
    group-of-prefixed-urls prefixed_urls_group_name
      prefixed-url url_1
      ...
```

```
        prefixed-url url_10
        end
```

Enabling the Group of Prefixed URLs in Rulebase

To enable the group of prefixed URLs in rulebase for processing prefixed URLs, use the following CLI commands:

```
configure
   active-charging service ecs_service_name
      rulebase rulebase_name
        url-preprocessing bypass group-of-prefixed-urls
prefixed_urls_group_name_1
        ...
        url-preprocessing bypass group-of-prefixed-urls
prefixed_urls_group_name_64
        end
```

This configuration on the control plane chassis will be pushed to the user plane with a PFD message for “group-of-prefixed-urls” and “rulebase-url-preprocessing”separately.

The group of prefixed URLs has the list of proxy URLs, which must be truncated. The rulebase contains multiple group of prefixed urls, which must be filtered. Charging ruledefs contain rules for actual URLs that must be searched after truncating URLs in the group of prefixed URLs.

Note

- Each group of prefixed URLs can have a maximum of ten prefixed URLs.
- A maximum of 64 group of prefixed URLs can be created and configured.

Show Commands

show user-plane-service group-of-prefixed-urls all | name *group_name*

This show command can be used on the user plane to verify whether the group of prefixed URLs are pushed or not. The output of this command is as follows:

- Name of the group of prefixed URLs
- Prefixed URLs
- Total number of prefixed URLs found

show user-plane-service rulebase name *rbase_name*

This show command can be used on the user plane to check whether the group of prefixed URLs is configured in rulebase or not. The output of this command is as follows:

- Name of rulebase
- Name of the groups of prefixed Urls for URL pre-processing

show user-plane-service statistics analyzer name http

The output of this command is as follows:

- Total HTTP Sessions
- Current HTTP Sessions
- Total Uplink Bytes
- Total Downlink Bytes
- Total Uplink Pkts
- Total Downlink Pkts
- Uplink Bytes Retrans
- Downlink Bytes Retrans
- Uplink Pkts Retrans
- Downlink Pkts Retrans
- Total Request Succeed
- Total Request Failed
- GET Requests
- POST Requests
- CONNECT Requests
- PUT requests
- HEAD requests
- Websocket Flows
- Invalid packets
- Wrong FSM packets
- Unknown request method
- Pipeline overflow requests
- Corrupt request packets
- Corrupt response packets
- Unhandled request packets
- Unhandled response packets
- Partial HTTP Header Anomaly prevented
- New requests on closed connection
- Memory allocation failures
- Packets after permanent failure
- Prefixed Urls Bypassed
- FastPath Statistics

- Total FP Flows
- Uplink (Total FP Pkts)
- Downlink (Total FP Pkts)
- Uplink (Total FP Bytes)
- Downlink (Total FP Bytes)



Note Prefixed URLs Bypassed counter has been added in http analyzer stats as a performance measurement to show the number of truncated prefixed URLs.

IP Readdressing

The IP Readdressing feature enables redirecting unknown gateway traffic based on the destination IP address of the packets to known/trusted gateways.

IP Readdressing is configured in the flow action defined in a charging action. IP readdressing works for traffic that matches a particular ruledef, and hence the charging action. IP readdressing is applicable to both uplink and downlink traffic. In the Enhanced Charging Subsystem, uplink packets are modified after packet inspection, rule matching, and so on, where the destination IP or port is determined, and replaced with the readdress IP or port just before they are sent out. Downlink packets (containing the readdressed IP or port) are modified when they are received, before the packet inspection, where the source IP or port is replaced with the original server IP or port number.

For one flow from an MS, if one packet is re-addressed, then all the packets in that flow is re-addressed to the same server. Features like DPI and rule-matching remain unaffected. Each IP address and port combination are defined as a ruledef.

In case of IP fragmentation, packets with successful IP reassembly are readdressed. However, IP fragmentation failure packets are not readdressed.

There are two different approaches for the readdress server selection, in case, server-list is configured under charging-action.

- **round-robin**: In round-robin approach, server selection happens in round-robin manner for every new flow. In round-robin, only active servers in the list are considered for selection.
- **hierarchy-based approach**: In hierarchy-based approach, servers are tagged as primary, secondary, tertiary, and so on, depending on the order they are defined in the readdress server-list. All flows are readdressed to the primary server until it is up and running. If Primary server goes down, then flows are readdressed to secondary server and the same logic goes on. Once primary server is active then flows switch back to primary server for readdressing.

An extra CLI is provided that enables user to select from hierarchy or round-robin approach for server selection. Both round-robin and hierarchy-based server selection approaches are applicable for both IPv4 and IPv6 based servers.

Configuring IP Readdressing

Readdressing of packets based on the destination IP address of the packets enables redirecting unknown gateway traffic to known/trusted gateways. This is implemented by configuring the re-address server in the charging action.

To configure the IP Readdressing feature, use the following configuration:

```
configure
  active-charging service acs_service_name
    charging-action charging_action_name
      flow action readdress server ipv4_address/ipv6_address [
discard-on-failure ]  [ dns-proxy-bypass ]  [ port port_number [
discard-on-failure ] [ dns-proxy-bypass ] ]
      end
```

To configure the IP Readdressing feature when the readdress server-list is defined under charging-action, use the following configuration:

```
configure
  active-charging service acs_service_name
    charging-action charging_action_name
      flow action readdress server-list server_list_name [ hierarchy ] [
round-robin ] [ dns-proxy-bypass ] [ discard-on-failure ]
      end
```

Following is the sample server-list configuration:

```
readdress-server-list DRE
consecutive-failures 1
response-timeout 10000
server 209.165.200.225 port 53
server 209.165.200.226 port 53
server 2001:420:54fe::1019 port 53
server 2001:420:54fe::1039 port 53
server 2001:420:54fe::1049 port 53
server 2001:420:54fe::1059 port 53
server 209.165.201.30 port 8080
#exit
```

Note A maximum of 10 servers can be configured in the list and a maximum of 10 lists can be configured in active-charging service.

Show Commands

This section provides information about the show CLI commands available in support of IP Readdressing feature.

CLI Command	Description
show user-plane-service readdress-server-list statistics all **show user-plane-service readdress-server-list statistics instance** *instance* **show user-plane-service readdress-server-list statistics name** *name*	Use these show CLI commands to display the readdress server list statistics.
show user-plane-service readdress-server-list statistics	

Use the **clear user-plane-service readdress-server-list statistics all**CLI to clear the readdress server list statistics.

Use the **show user-plane-service statistics charging-action all** CLI to check the "flows readdressed" counter.

L7 Protocol

The following L7 protocols are supported as part of DPI:

- DNS
- FTP
- HTTP
- HTTPS
- RTP/RTSP
- SIP

DNS

The UPF supports DNS protocol as part of L7 Analyzer.

FTP

The UPF 2.0 supports FTP proocol as part of L7 Analyzer.

HTTP

On completion of HTTP Request or HTTP Response, the uplink or downlink data packets are offloaded to VPP in the following cases:

- Content-Length—Volume-based offloading is supported for methods such as GET and POST. The HTTP flow with chunk-encoding data transfer mechanism does not get offloaded irrespective of the method defined in HTTP. If the stream is offloaded based on content-length, then the stream on the other end will also get offloaded until the former is not onloaded.
- CONNECT Method—The method where both uplink and downlink streams are offloaded after the flow is upgraded to CONNECT.

- WebSocket Method—After the flow is classified as WebSocket protocol, both uplink and downlink streams are offloaded.
- The streams are onloaded back in either of the following cases:
  - FIN packet received
  - Content-length is breached
  - PDN update

Header Parsing

Only the header fields defined in ruledefs, which are included in rulebase, are parsed. For x-header features, redirection is configured with dependencies on some HTTP header fields.

HTTP Charging

- Complete packets are charged.
- Partial packets are charged on completion. Packet completing the partial packet is also charged.
- Concatenated packets are charged.
- Delay Charging is enabled – Control packets are charged against application-based rule, depending on delay charging CLI configuration.
- Response-based charging is enabled – After HTTP request's response is received, then the HTTP request is charged against response rule's CA.

X-Header Parsing and Rule-Matching

Ruledefs with x-header rule-lines are parsed and matched.

WebSocket

Involves charging of subsequent packets of the flow after HTTP GET request as per the HTTP request, if the HTTP flow is upgraded to be a websocket flow.

Response-based TRM

Transactional Rule Matching is engaged after HTTP response packet is received.

X-Header Insertion

X-header Insertion is supported in HTTP Requests.

URL-based redirection

URL-based redirection is a UPF function that:

- returns a redirect response to the subscriber and terminates the existing TCP connections,
- triggers the subscriber's web browser to automatically send the original HTTP packet to a specified destination URL, and

- allows for the extraction of specific destination URLs from composite strings (e.g., returnURL) received via dynamic policies.

If the traffic flow is not HTTP, the UPF ignores the redirect URL option and forwards the packet normally. Redirection is supported only for specific HTTP packets (e.g., GET requests) sent in the uplink direction.

Table 5: Feature History

Feature Name	Release Information	Description
Redirect URL Encryption	2024.01	UPF supports the URL encryption of the dynamic fields using the Blowfish and AES encryption methods. Default Setting: Disabled – Configuration Required
Redirect Rewrite and TCP FIN/RST	2026.01.1	The ADC dynamic rule on the N7 interface performs Redirect URL Support. This is supported only for AES encryption.

Dynamic Fields and Encryption:

- **Field Insertion:** The UPF supports appending dynamic subscriber data to the redirected URL.
- **Security:** Sensitive dynamic fields can be protected using symmetric block ciphers.
  - **Blowfish:** Uses a 128-bit key for encryption and decryption.
  - **AES:** Uses Cipher Block Chaining (CBC) mode with 256-bit keys. The UPF uses a hardcoded value of 10 rounds for AES encryption.

Note Redirect URL on the N7 interface is supported only for AES encryption.

Restrictions for URL redirection using N7 interface

- This feature supports UL L7 packets only. It does not support DL L7 packets, as the feature requirement is specifically to support UL L7.
- Do not modify existing charging action configurations.

Supported dynamic fields for redirection

The UPF supports the following dynamic fields for insertion into redirected URLs. Fields marked with an asterisk (*) support encryption.

- #ACSMGR_BEARER_CALLED_STATION_ID#
- #BEARER.IMSI# *

- #BEARER.MSISDN# *
- #HTTP.HOST#
- #HTTP.URL#
- #HTTP.URI#
- #RTSP.URI#
- #RULEBASE#

The UPF supports the following dynamic fields for insertion into redirected URLs on N7 interface

- #BEARER.IMSI# *
- #HTTP.URL#

How UPF redirection and termination work

The UPF uses the service-scheme framework to detect specific policy events. It then executes redirection or flow termination based on ADC dynamic rules received over the N4 interface from the SMF.

Summary

The key components involved in the process are:

- PCF: provides dynamic policies containing composite redirect URLs or block instructions.
- SMF: relays policy information to the UPF over the N4 interface.
- UPF: Performs redirect and encryption, inserts dynamic data, and manages TCP signaling (FIN/RST) for blocked traffic.

Workflow

The workflow to enable redirect or termination by N7 Interface:

1. **service-scheme Triggering:** The service-scheme framework detects a far-received or pdr-received event during session establishment or modification.
2. **Condition Matching:** The system evaluates trigger-conditions, such as checking if a redirect URL contains specific keywords like returnURL or other keyword based on the configuration.
3. **Action Execution:** The UPF executes the mapped trigger-action, which points to a specific charging-action based on the adc rules coming during establishment or modification of session.
4. **URL Processing:** For redirection, the UPF extracts the actual URL, populates dynamic fields, and applies encryption.
5. **Flow Management:** For termination, the UPF sends a TCP FIN to the web client. It also sends a TCP RST to the web server to gracefully close the connection.

Configuring URL Encryption

To enable encryption of the dynamic fields in the redirected URL, use the following sample configuration:

```
configure
  active-charging service service_name
    charging-action charging_action_name
      flow action redirect-url redirect_url [ encryption { blowfish128 |
aes256 } [ encrypted ] key key ]
    end
```

NOTES:

- **flow action redirect-url** *redirect_url*—Specify the redirect URL. It can include up to 16 dynamic fields. *redirect_url* must be an alphanumeric string of 1 through 511 characters.
- **encryption { blowfish128 | aes256 } [encrypted] key** *key*—Specify to enable encryption for dynamic fields of the redirect URL.
  - **blowfish128**—Specify to use Blowfish encryption with 128-bit key for encrypting the dynamic fields.
  - **aes256**—Specify to use AES-CBC encryption with 256-bit key for encrypting the dynamic fields.
  - **encrypted**—Specify to encrypt the key.
  - **key** *key*—Specify the key to use for encryption of dynamic fields. *key* must be an alphanumeric string of 1 through 523 characters.
- If encoding is enabled in the URL to be redirected, the encrypted URL supports up to 2300 characters. If the length exceeds, then the URL will be truncated at 2300 characters.

Configure N7 based redirect flow action

Set up the UPF to process dynamic redirect URLs using N7 interface and apply encryption.

Before you begin

- Verify that the L7 Analyzer is active (required for HTTPS support).
- If using AES encryption, prepare a key phrase (alphanumeric, 1 to 523 characters).
- Ensure the redirect URL string is between 1 and 511 characters.

Follow these steps to configure redirect flow actions:

Procedure

Step 1 Define the ruledefs for URL identification.

Example:

```
[upf] # ruledef rule_name
 [upf] #  http url contains test.html
 [upf] #  http url !contains abc.html
 [upf] #  multi-line-or all-lines
```

Step 2 Configure the trigger-conditions for redirect URLs.

Example:

```
[upf] # trigger-condition TC1
 [upf] #  redirect-url contains "returnURL="
 [upf] #  redirect-url starts-with "https://snap"
```

Step 3 Define trigger-actions for redirection.

Example:

```
[upf] # trigger-action TA1
    [upf] #  execute-charging-action url_redirect_ca
```

Step 4 Create charging-actions with encryption parameters.

Example:

```
[upf] # charging-action url_redirect_ca
[upf] #  flow action redirect-url <return url recieved over n4 interface with ADC rule>
```

Step 5 Bind the trigger to the service-scheme.

Example:

```
[upf] # service-scheme SS1
 [upf] #  trigger far-received
   [upf] #  priority 1 trigger-condition TC1 trigger-action TA1
```

Configure terminate flow action

Implement traffic blocking using TCP signaling.

Before you begin

Follow these steps to configure flow termination:

Procedure

Step 1 Define the ruledefs for traffic blocking.

Example:

```
[upf] # ruledef terminate_rule
  [upf] #  ip any-match = TRUE
```

Step 2 Configure the trigger-conditions for termination.

Example:

```
[upf] # trigger-condition TC2
  [upf] # tdf-appid contains <terminate app id recived with ADC dynamic rule over N4 interface>
```

Step 3 Define trigger-actions for termination.

Example:

```
[upf] # trigger-action TA2
   [upf] #   execute-charging-action terminate_all
```

Step 4 Create the charging-action for flow termination.

Example:

```
[upf] # charging-action terminate_all
   [upf] #  flow action terminate-flow
```

Step 5 Bind the trigger to the service-scheme.

Example:

```
[upf] # service-scheme SS1
[upf] #   trigger pdr-received
[upf] #   priority 2 trigger-condition TC2 trigger-action TA2
```

Redirect and terminate flow statistics

You can use the following commands to verify the feature status and view statistics for redirected or terminated flows.

Subscriber FAR Details

To view the extracted redirect URL and the applied charging action for a specific call, use the following syntax:

Example:

```
[upf] # show subscribers user-plane-only callid [call_id] far full all
```

Sample Output:

```
Callid: 00004e29
Interface Type: Sxb
 FAR-ID       : 0x0003
 Destination Interface : Core
 Apply Action     : FORWARD
 Outer Header Creation   :
 Remote TEID     : 0x0
 Transport Level Marking    : N/A
 Transport Level Marking Options :
   Copy Inner:                      No
   Copy Outer:                      No
 Inner Packet Marking            : N/A
 Layer2 Marking                  : N/A
 Remote IP Address   :
 Remote Port    : N/A
 Number of Associated PDRs: 1
 Linked Traffic Endpoint : 0x2
 Redirect Info  :
Redirect Address Type  : URL
Redirect Address Value :
https://snap.mobile.att.net/?nonce=ApolloHONDA&returnURL=https://myvehicle.att.com/%23/start%3Foem=honda

       Charging Action Name           : url_redirect_ca
```

Trigger Action Configuration

To verify which charging actions are mapped to specific trigger actions, use the following syntax:

Example:

```
[upf] # show user-plane-service trigger-action all
```

Sample Output:

```
Trigger-Action: TA1
       HTTP Response Based TRM      : none
       HTTP Response Based Charging : none
       Throttle Suppress            : Disabled
       Flow Recovery                : Disabled
       Traffic Optimization         : Disabled
       Step Up GBR                  : Disabled
       Step Down GBR                : Disabled
       TCP Acceleration             : Disabled
       TCP Acceleration Threshold   : Disabled
       Service-Chain                : none
       UP-Service-Chain             : none
       Flow-IP-Readdressing         : Disabled
       EDNS-Encode                  : Disabled
       Charging-Action-Map          : none
       Charging-Action-Name         : url_redirect_ca

Trigger-Action: TA2
       HTTP Response Based TRM      : none
       HTTP Response Based Charging : none
       Throttle Suppress            : Disabled
       Flow Recovery                : Disabled
       Traffic Optimization         : Disabled
       Step Up GBR                  : Disabled
       Step Down GBR                : Disabled
       TCP Acceleration             : Disabled
       TCP Acceleration Threshold   : Disabled
       Service-Chain                : none
       UP-Service-Chain             : none
       Flow-IP-Readdressing         : Disabled
       EDNS-Encode                  : Disabled
       Charging-Action-Map          : none
       Charging-Action-Name         : terminate_all
```

Trigger Condition Details

To review the criteria used to select charging actions, use the following syntax:

Example:

```
[upf] # show user-plane-service trigger-condition all
```

Sample Output:

```
Trigger-Condition: TC1
       redirect-url  contains returnURL=
       redirect-url  starts-with https://snap
       redirect-url  contains nonce=Name
       Multi-line-OR All lines : Disabled

Trigger-Condition: TC2
       Tdf-AppId  contains terminate_
       Multi-line-OR All lines : Disabled
```

Global Redirect Statistics

To view global counts for redirected flows, use the following syntax:

Example:

```
[upf] # show user-plane-service statistics all
```

Sample Output:

```
ADC Redirect Stats:
    ADC Redirected Flows         : 3
```

Terminated Flow Counts

To view the number of terminated flows for a specific call, use the following syntax:

Example:

```
[upf] # show subscribers user-plane-only full callid <call_id>
```

Sample Output:

```
Flow Action Terminated Flows: 5
```

PDR Termination Details

To view the number of packets matched while terminating flows and the applied charging action, use the following syntax:

Example:

```
[upf] # show subscribers user-plane-only callid <call_id> pdr full all
```

Sample Output:

```
Callid: 12345
Interface Type: Sxb
IP address: 11.11.0.1
PDR-ID:                    0x0003
Rule Name:                dynamicfacebook
Charging Action Name:      terminate_all
TDF App Id:   terminate_rule
TDF Notifications:   Enabled
Hits:                      5 Match Bypassed:            0
Matched Bytes:          285 Matched Packets:            5
```

HTTPS

The UPF supports HTTPS protocol as part of L7 Analyzer.

RTP/RTSP

The UPF 2.0 supports RTP and RTSP protocols as part of L7 Analyzer.

SIP

Session Initiation Protocol is an IETF-defined signaling protocol widely used for controlling communication sessions such as voice and video calls over Internet Protocol (IP). The protocol can be used for creating, modifying and terminating two-party (unicast) or multiparty (multicast) sessions. Sessions may consist of one or several media streams.

The UPF 2.0 supports SIP as part of L7 Analyzer.

Monitoring and Troubleshooting

This section provides information regarding the CLI command available in support of monitoring and troubleshooting the feature.

Show Command(s) and/or Outputs

This section provides information regarding show commands and/or their outputs in support of this feature.

DNS

Use the following CLI command to get statistics related to DNS:

```
show user-plane-service statistics analyzer name dns
```

FTP

Use the following CLI command to get statistics related to FTP:

```
show user-plane-service statistics analyzer name ftp
```

HTTP

Use the following CLI command to get statistics related to HTTP:

```
show user-plane-service statistics analyzer name http
```

HTTPS

Use the following CLI command to get statistics related to HTTPS:

```
show user-plane-service statistics analyzer name secure-http
```

RTP

Use the following CLI command to get statistics related to RTP:

```
show user-plane-service statistics analyzer name rtp
```

RTSP

Use the following CLI commands to get statistics related to RTSP:

- `show user-plane-service statistics analyzer name rtsp`
- `show user-plane-service statistics analyzer name rtsp verbose`

SIP

Use the following CLI command to get statistics related to SIP:

```
show user-plane-service statistics analyzer name sip
```

NAT Support on N4

Table 6: Feature History

Feature Name	Release Information	Description
NAT Support on the N4 interface	2023.04	The Network Address Translation (NAT) feature translates non-routable private IP addresses to routable public IP addresses from a pool of public IP addresses. UPF supports NAT on the N4 interface to configure network addresses and send NAT binding records to N4. The NAT policy and IP pool for NAT public IP addresses are configured on UPF for N4.

Feature Description

The Network Address Translation (NAT) feature translates non-routable private IP address(es) to routable public IP address(es) from a pool of public IP addresses. This feature enables the user to conserve on the number of public IP addresses required to communicate with external networks. This feature also ensures security as the IP address scheme for the internal network is masked from external hosts, and each outgoing and incoming packet goes through the translation process.

UPF supports NAT on the N4 interface to configure network addresses. The system is configured to automatically forward data packets encapsulating the source IP or source port address of the UE with NAT IP address and NAT port.

The supported NAT combinations include:

- NAT44 On Demand Many to One
- NAT44 On Demand One to One
- NAT64 On Demand Many to One
- NAT 64 On Demand One to One
- NAT44 Not On Demand Many to One
- NAT44 Not On Demand One to One
- NAT64 Not On Demand Many to One
- NAT64 Not On Demand One to One

For supplemental information about NAT, see the StarOS *NAT Administration Guide*.

NOTE: Not all features and/or functionality that are mentioned in the StarOS *NAT Administration Guide* are applicable to UPF.

Limitations

NAT support on the N4 interface has the following limitations:

- Supports only NAT44 with many-to-one and on-demand modes.
- All NAT pools are configured at respective User Plane in the destination context.
- Does not support charging action with CLI action deny in fw-and-nat policy and flow-any-error charging action in active-charging-service.
- Does not support access-rules which are configured with "dynamic-only" and "static-and-dynamic" rules from external servers.
- Does not support multiple IPs from the same realm.
- Does not support the next hop forwarding in NAT pool.
- Does not support the port range in NAT pool.
- Does not support the skip private IP check CLI.
- Does not support RADIUS and Gy returned Fw-and-nat policy-based applying NAT policy.
- Does not support the bearer-specific filters in access-ruledefs.
- Access-rules do not support trigger open-port port range configuration in fw-and-nat policy.
- Does not support the NAT port recovery (fw-and-nat action) after SR/ICSR.
- Does not support the NAT reassembly timeout CLI in active-charging service. The generic context-level CLI on UP must be used instead.
- Does not support the NAT fragmentation reassembly failure.
- Does not support the NAT flow-mapping timer.
- For N:M redundancy, the NAT IP pools to be configured from RCM done as part of interface configuration for each UP host and the pool name must be unique across all active user planes. It is mandatory to use NAT groups for all pools so that the same NAT realm referred in fw-and-nat policy can be applicable to all user planes.
- In N:M redundancy, the total number of NAT IP pools collectively configured on all UPs through RCM must be as per the maximum limit (2000) of IP pools. The configuration in standby User Plane fails if the cumulative total of all active UPs exceeds the maximum value.

Configuring NAT

The NAT policy (Firewall-and-NAT policy) and access-ruledef configurations are made under active-charging service in UPF.

For information on NAT-related CLI commands, refer to the *StarOS NAT Administration Guide > NAT Configuration* chapter.



Note Not all CLI commands and configurations mentioned in the *StarOS NAT Administration Guide > NAT Configuration* chapter are applicable to UPF.

Sample Configurations

User Plane

The following pool-related configuration is required at User Plane in ISP context.

```
configure
   context ISP1-UP
      ip pool NAT44_PUBLIC1 209.165.200.225 255.255.255.224 napt-users-per-ip-address 2
on-demand port-chunk-size 16 max-chunks-per-user 4 group-name NAT44_GRP1
      ip pool NAT44_PUBLIC2 209.165.200.226 255.255.255.224 napt-users-per-ip-address 2
on-demand port-chunk-size 16 max-chunks-per-user 4 group-name NAT44_GRP1
      ip pool NAT44_PUBLIC3 209.165.200.227 255.255.255.224 napt-users-per-ip-address 2
on-demand port-chunk-size 8 max-chunks-per-user 1 group-name NAT44_GRP2
      ip pool NAT44_PUBLIC4 209.165.200.228 255.255.255.224 napt-users-per-ip-address 4
on-demand port-chunk-size 32256 max-chunks-per-user 4 group-name NAT44_GRP2
      ip pool NAT44_PUBLIC5 209.165.200.229 255.255.255.224 napt-users-per-ip-address 8064
 on-demand port-chunk-size 8 max-chunks-per-user 2
      end
```

Sample NAT Pool Related Configuration for Different NAT Pool Types

```
1-1 on-demand:
---------------------------
config
context ISP1-UP
ip pool NAT44_ipv4_1_1 209.165.200.230 255.255.255.224 nat-one-to-one on-demand
nat-binding-timer 60
end

N-1 Not-on-demand:
------------------------
config
context ISP1-UP
ip pool NAT44_ipv4_N_1 209.165.200.231 255.255.255.224 napt-users-per-ip-address 2
max-chunks-per-user 2 port-chunk-size 8
end

1-1 Not-on-demand:
------------------------
config
context ISP1-UP
ip pool NAT44_ipv4_NOD_1_1 209.165.200.232 255.255.255.224 nat-one-to-one
end
```



Note Add the Control Plane configuration and map one or more access rule definitions to any required NAT Pool or Group configured in the User Plane.

Monitoring and Troubleshooting

Gathering NAT Statistics

The following table lists the commands that can be used to gather NAT statistics.

The first column lists the statistics to gather and the second column lists the command to use.

Statistics/Information	Show Command
Information for all current subscribers who have either active or dormant sessions. Checks IP address associated with subscriber. Also displays all the IP addresses that are in use in a NAT realm.	**show subscribers user-plane-only full all**
Information on NAT subsystem statistics.	**show user-plane-service statistics all**
All NAT-related statistics.	**show user-plane-service statistics nat all**
All NAT Realm-related statistics.	**show user-plane-service statistics nat nat-realm all**
Statistics of all NAT IP pools in a NAT IP pool group.	**show user-plane-service statistics nat nat-realm** *pool_name*
Information on NAT bind records generated.	**show user-plane-service edr-format statistics all**
Verifying association of fw-and-nat policy in APN on UP.	**show user-plane-service pdn-instance name** *name*
Verifying cofiguration of fw-an-nat policy on UP.	**show user-plane-service fw-and-nat policy all**
Information on NAT bind records generated for port chunk allocation and release.	**show user-plane-service rulebase name** *name*
Information on access ruledef.	**show user-plane-service ruledef all**
Verifying association of fw-and-nat policy in rulebase on UP.	**show user-plane-service rulebase name** *name*

Clear Commands

The following clear CLI commands are available in support of this feature:

- **clear user-plane-service statistics nat nat-realm all**
- **clear user-plane-service statistics nat all**

SNMP Traps for NAT Parameter Thresholds

The following SNMP traps for NAT parameter thresholds are supported.

SNMP Traps	Description
ThreshNATPortChunks	Generated when NAT port chunk usage reaches configured threshold limit
ThreshClearNATPortChunks	Generated when NAT port chunk usage reaches configured clear threshold limit.
ThreshNATPktDrop	Generated when NAT packet drop reaches configured threshold limit.
ThreshClearNATPktDrop	Generated when NAT packet drop reaches configured clear threshold limit.
ThreshIPPoolUsed	Generated when the number of IPs used in the IP Pool reaches configured threshold limit.

SNMP Traps	Description
ThreshClearIPPoolUsed	Generated when the number of IPs used in the IP Pool reaches configured clear threshold limit.
ThreshIPPoolFree	Generated when IP pool is free and threshold limit reached.
ThreshClearIPPoolFree	Generated when IP pool is used, and clear threshold limit reached.
ThreshIPPoolAvail	Generated when IP pool is available for next flow and configured threshold reached.
ThreshClearIPPoolAvail	Generated when IP pool is used, and configured threshold is reached.

NOTE: The respective CLIs must be configured in the User Plane to enable these traps.

Bulk Statistics

NAT-realm Schema

The NAT realms are configured in User Plane and statistics are stored per-context per-realm. These statistic variables, both cumulative and snapshot, are available in the nat-realm schema.

Table 7: NAT-realm Schema

Variable Name	Data Type	Counter Type	Description
Vpnname	String	Info	Context name.
Realmname	String	Info	Realm name.
nat-rlm-bind-updates	Int64	Counter	Total interim AAA NBU sent.
nat-rlm-bytes-txferred	Int64	Counter	Total number of NAT44 and NAT64 bytes transferred by realm (uplink + downlink).
nat-rlm-bytes-nat44-tx	Int64	Counter	Total number of NAT44 bytes transferred by realm.
nat-rlm-bytes-nat64-tx	Int64	Counter	Total number of NAT64 bytes transferred by realm.
nat-rlm-ip-flows	Int64	Counter	Total number of NAT44 and NAT64 flows used by the realm.
nat-rlm-nat44-flows	Int64	Counter	Total number of NAT44 flows processed by realm.
nat-rlm-nat64-flows	Int64	Counter	Total number of NAT64 flows processed by realm.
nat-rlm-ip-denied	Int32	Counter	Total number of NAT44 and NAT64 flows denied NAT IP address.
nat-rlm-ip-denied-nat44	Int64	Counter	Total number of NAT44 flows denied IP.

Variable Name	Data Type	Counter Type	Description
nat-rlm-ip-denied-nat64	Int64	Counter	Total number of NAT64 flows denied IP.
nat-rlm-port-denied	Int32	Counter	Total number of NAT44 and NAT64 flows denied ports.
nat-rlm-port-denied-nat44	Int64	Counter	Total number of NAT44 flows denied ports.
nat-rlm-port-denied-nat64	Int64	Counter	Total number of NAT64 flows denied ports.
nat-rlm-memory-denied	Int64	Counter	Total number of NAT44 and NAT64 flows denied memory.
nat-rlm-memory-denied-nat44	Int64	Counter	Total number of NAT44 flows denied memory.
nat-rlm-memory-denied-nat64	Int64	Counter	Total number of NAT64 flows denied memory.
nat-rlm-ttl-ips	Int32	Gauge	Total number of NAT public IP addresses, per context per NAT realm. Is a static value.
nat-rlm-ips-in-use	Int32	Gauge	Total number of NAT IP addresses currently in use, per context per NAT realm.
nat-rlm-current-users	Int32	Gauge	Total number of subscribers currently using the NAT realm.
nat-rlm-ttl-port-chunks	Int32	Gauge	Total number port-chunks, per context per NAT realm. Is a static value.
nat-rlm-chunks-in-use	Int32	Gauge	Total number of port-chunks currently in use, per context per NAT realm.
nat-rlm-port-chunk-size	Int32	Gauge	Size of the port chunk in the NAT realm.
nat-rlm-port-chunk-average-usage-tcp	Int32	Gauge	Average TCP port usage in the allocated TCP ports, i.e. out of allocated TCP ports how many got used. Not percentage value.
nat-rlm-port-chunk-average-usage-udp	Int32	Gauge	Average UDP port usage in the allocated UDP ports, i.e. out of allocated UDP ports how many got used. Not percentage value.
nat-rlm-port-chunk-average-usage-others	Int32	Gauge	Average other (ICMP or GRE) port usage in the allocated other ports, i.e. out of allocated ‘other’ ports how many got used. Not percentage value.
nat-rlm-max-port-chunk-subs	Int64	Counter	Total number of subscribers who used maximum number of port chunks.
nat-rlm-max-port-chunk-used	Int32	Counter	Maximum port chunks used.

Variable Name	Data Type	Counter Type	Description
nat-rlm-max-cur-port-chunk-subs	Int64	Gauge	Current number of subscribers using maximum number of port chunks.
nat-rlm-max-cur-port-chunk-used	Int32	Gauge	Maximum port chunks used by active subscribers.

NAT Binding Records

Whenever a NAT IP address or NAT port-chunk is allocated/deallocated to/from a subscriber, NAT Binding Records (NBR) can be generated. Generation of NBRs is configurable in the Firewall-and-NAT policy configuration.

Sample NBR

```
#sn-start-time,sn-end-time,ip-protocol,ip-subscriber-ip-address,ip-server-ip-address,sn-subscriber-port,
sn-server-port,sn-nat-ip,sn-nat-port-block-start,sn-nat-port-block-end,sn-subscriber-nat-flow-ip,sn-subscriber-nat-flow-port,
sn-nat-realm-name,sn-nat-subscribers-per-ip-address,sn-nat-binding-timer,sn-nat-gmt-offset,sn-nat-port-chunk-alloc-dealloc-flag,
sn-nat-port-chunk-alloc-time-gmt,sn-nat-port-chunk-dealloc-time-gmt,sn-nat-no-port-packet-dropped,sn-closure-reason
,,,209.165.200.225,,,,209.165.201.1,1024,1039,,,NAT44_PUBLIC2,2,60,+0530,1,02/18/2020
06:41:08,,,
,,,209.165.200.225,,,,209.165.201.2,1024,1031,,,NAT44_PUBLIC5,8064,60,+0530,1,02/18/2020
06:41:14,,,
,,,209.165.200.225,,,,209.165.201.3,1024,1039,,,NAT44_PUBLIC2,2,60,+0530,0,02/18/2020
06:41:08,02/18/2020 06:42:12,,
,,,209.165.200.225,,,,209.165.201.14,1024,1031,,,NAT44_PUBLIC5,8064,60,+0530,0,02/18/2020
06:41:14,02/18/2020 06:44:24,,
```

EDRs

The following NAT-specific attributes are supported in regular EDRs:

- sn-nat-subscribers-per-ip-address: Subscriber(s) per NAT IP address
- sn-subscriber-nat-flow-ip: NAT IP address of NAT-enabled subscribers
- sn-subscriber-nat-flow-port: NAT port number of NAT-enabled subscribers

Sample EDR

```
#sn-start-time,sn-end-time,ip-protocol,ip-subscriber-ip-address,ip-server-ip-address,sn-subscriber-port,sn-server-port,
sn-nat-ip,sn-nat-port-block-start,sn-nat-port-block-end,sn-subscriber-nat-flow-ip,sn-subscriber-nat-flow-port,sn-nat-realm-name,
sn-nat-subscribers-per-ip-address,sn-nat-binding-timer,sn-nat-gmt-offset,sn-nat-port-chunk-alloc-dealloc-flag,sn-nat-port-chunk-alloc-time-gmt,
sn-nat-port-chunk-dealloc-time-gmt,sn-nat-no-port-packet-dropped,sn-closure-reason
02/18/2020 12:11:11:630,02/18/2020
12:11:11:632,1,209.165.200.225,209.165.201.1,0,0,,,,209.165.200.230,1024,,2,,,,,,0,0
02/18/2020 12:11:08:672,02/18/2020
12:11:09:671,6,209.165.200.225,209.165.201.1,1001,3000,,,,209.165.200.230,1034,,2,,,,,,0,0
02/18/2020 12:11:14:499,02/18/2020
12:11:14:499,17,209.165.200.225,209.165.201.1,1001,3000,,,,209.165.200.240,1025,,8064,,,,,,0,0
```

Packet Drop EDR

Sample Packet Drop EDR

```
#sn-nat-no-port-packet-dropped,sn-start-time,sn-end-time,sn-subscriber-imsi
2,03/13/2020 08:28:24,03/13/2020 08:28:54,123456789012345
```

Tethering Detection

Feature Description

Tethering refers to the use of a mobile smartphone as a USB dongle or modem to provide Internet connectivity to PC devices (laptops, PDAs, tablets, and so on) running on the smartphone's data plan. Typically, for smartphone users, most operators have in place an unlimited data plan, the usage of which is intended to be from the smartphone as a mobile device. However, some subscribers use the low cost or unlimited usage data plan to provide Internet connectivity to their laptops in places where normal Internet connection through broadband or Wi-Fi may be costly, unavailable, or insecure.

The Tethering Detection feature enables detection of subscriber data traffic originating from PC devices tethered to mobile smartphones, and also provides effective reporting to enable service providers take business decisions on how to manage such usage and to bill subscribers accordingly. Tethering Detection is supported for IPv4 (TCP) and IPv6 traffic flows.

In this release, IP-TTL based tethering is supported. This feature is configurable at the rulebase level and is applicable on all flows for all subscribers having IP-TTL configuration within the rulebase.

Configuring Tethering Support

This section describes how to configure the Tethering Support feature.

Configuring the Tethering Support feature involves the following steps:

- Rulebase Configuration for Tethering
- Ruledef Configuration for Tethering
- EDR Configuration for Tethering

Rulebase Configuration for Tethering

Use the following commands to configure the rulebase parameters for tethering.

```
configure
  active-charging service service_name
    rulebase rulebase_name
      tethering-detection ip-ttl valuettl_value
      end
```

NOTES:

- **tethering-detection**: This command allows you to enable/disable the Tethering Detection feature for the current rulebase, and specifies the database to use.
- **ip-ttl value** *ttl_value*: Specifies to perform tethering detection using IP-TTL configuration. *ttl_value* must be an integer from 1 through 255 to configure TTL values for tethered flows.

Ruledef Configuration for Tethering

Use the following commands to configure ruldef parameters for tethering.

```
configure
   active-charging service service_name
      ruledef ruledef_name
         ip any-match operator_condition
         tethering-detection ip-ttl flow-tethered
         end
```

NOTES:

- **ip any-match** *operator_condition*: This command allows you to define rule expressions to match all IPv4/IPv6 packets.
- **ip-ttl**: Specifies to select flows that were tethered or non-tethered as per IP-TTL values.
- **flow-tethered**: Specifies to match if tethering is detected on flow.

EDR Configuration for Tethering

Use the following commands to configure EDR for tethering:

```
configure
   active-charging service service_name
      edr-format format_name
         rule-variable flow tethered-ip-ttl priority priority_value
         rule-variable flow ttl priority priority_value
         end
```

NOTES:

- **edr-format** *format_name*: configures EDR formats.
- **flow**: Configures the flow related fileds in an EDR.
- **tethered-ip-ttl**: IP-TTL based tethering detected on flow.
- **ttl**: Time To Live/Max hops value received in the first packet of the flow.

Monitoring and Troubleshooting

This section provides information regarding the CLI command available in support of monitoring and troubleshooting the feature.

Show Command(s) and/or Outputs

This section provides information regarding show commands and/or their outputs in support of this feature.

show user-plane-service statistics tethering-detection

The following fields are displayed in support of this feature:

- Current Tethered Subscribers
- Total Tethered Subscribers
- Total flows scanned

- Total Tethered flows detected
- Total Tethered flows recovered
- Total flows bypassed for scanning
- Tethering Detection Statistics (ip-ttl)
  - Total flows scanned
  - Tethered flows detected
  - Tethered uplink packets
  - Tethered downlink packets

show user-plane-service statistics rulebase name rulebase_name

The following fields are displayed in support of this feature:

- Tethering Detection (ip-ttl)
  - Total flows scanned
  - Tethered flows detected
  - Tethered uplink packets
  - Tethered downlink packets

RTP Dynamic Flow Detection

The **rtp dynamic-flow-detection** CLI command, under the ACS Rulebase Configuration mode, enables the Real Time Streaming Protocol (RTSP) and Session Description Protocol (SDP) analyzers to detect the child RTP and RTCP flows. If you configure the RTSP/SIP and SDP analyzers, and **rtp dynamic-flow-detection** CLI is present, then there's no need for configuring RTP/RTCP explicitly. With the **rtp dynamic-flow-detection** CLI command, the child RTP or RTCP flows get corelated to their parent RTSP/SIP-SDP flows.

Once the parent flow (RTSP/SIP-SDP) gets cleared, the child RTP/RTCP flows also gets cleared. In the absence of this CLI, the L7 layer analysis for RTP and RTCP needs a separate analyzer configuration. There’s no correlation of RTP/RTCP flows to RTSP/SIP-SDP flow.

Rule-matching for Bearer-specific Filters

The Rule-matching for Bearer-specific Filters functionality includes:

- IMSI-based rules are matched as per the subscribers IMSI.
- APN-based rules allow you to define rule expressions to match Access Point Name (APN) of the bearer flow.
- RAT-Type allows you to define rule expressions to match Radio Access Technology (RAT) in the bearer flow.

Configuring IMSI Pool

Use the following CLI commands to configure the IMSI pool.

```
configure
   active-charging service service_name
      imsi-pool pool_name
         imsi { imsi_number | range start_imsi to end_imsi }
```

The imsi-pool can contain either IMSI value or range of IMSI.

Configuring Rule-line ACS Ruledef

Use the following CLI commands to configure rule-line in ACS Ruledef Configuration mode.

```
configure
   active-charging service service_name
      ruledef ruledef_name
         bearer 3gpp imsi { = imsi_value } | { range imsi-pool pool_name }
         bearer 3gpp apn operator apn_name
         bearer 3gpp rat-type operator rat_type
```

IMSI range can be configured in a rule with the help of IMSI pool.

bearer 3gpp rat-type *operator rat_type*:

- *operator* must be one of the following:
  - **!=** : Does not equal
  - **=** : Equals
- **NOTE**: In this release, **wlan** is the qualified *rat_type*.

Configuring HTTP Content-Type

Use the following CLI commands to define rule expressions to match value in HTTP Content-Type entity-header field.

```
configure
   active-charging service service_name
      ruledef ruledef_name
         http content type [ case-sensitive ] operator content_type
```

Show CLIs

Use the following CLI on UPF to see information about IMSI pool that is configured in a service:**show user-plane-service imsi-pool name** *pool_name*

Enablement of IP Source Violation Detection for Uplink Packets

Table 8: Feature History

Feature Name	Release Information	Description
Enable IP Source Violation Detection for Uplink Packets	2024.03.0	This feature on cnSGWc lets UPF check if the origin of the uplink packets is from the right subscriber. This feature provides the following support: • Enhance the security and privacy of the subscribers by preventing the leakage of their data to unauthorized parties. This feature further reduces the risk of legal and regulatory issues for the service provider by complying with the lawful interception requirements. • Delayed reuse of the Tunnel Endpoint Identifier (TEID) that cnSGW assigns for a session until the session termination prevents tapping the wrong subscriber, risk of packet misassociation, and reporting by Lawful Interception (LI) agencies. This feature uses the existing CLI command **show subscribers user-plane-only full all | callid** *callid* to view the configured **IP Source Violation Action** option. **Default Setting**: Disabled – Configuration Required to Enable

The IP source violation detection feature is enabled on cnSGW. This information is communicated from cnSGWc to UPF. Based on the configuration done in cnSGW, which is to ignore or discard the packets, the User Plane acts on the source violated packet.

After a UE connected to a network, cnSGW assigns a unique Tunnel Endpoint Identifier (TEID) for a new session and includes TEID in the Create Session Request message to UPF. The UPF acknowledges the session establishment and uses the TEID to route user data packets.

With this feature, the Tunnel Endpoint Identifier (TEID) that the cnSGW assigns for a session per subscriber, is not reused immediately for another subscriber until the session termination. The delayed TEID reuse avoids tapping the wrong subscriber and reporting by LI agencies.

You can verify the IP source violation on User Plane through the **show subscribers user-plane-only full all | callid** *call_id* CLI command.

For details on the validating the uplink packets for IP source violation, see Validation of Uplink Packets for IP Source Violation in the *cnSGWc Configuration and Admininstration Guide*.

How Enabling of IP Source Violation for Uplink Packets Works

Following are the key points when you enable or disable this feature:

- By default, this feature is disabled.
- When this feature is enabled with the IGNORE option, the packets are validated for invalid source IP address. The corresponding statistics are incremented accordingly. However, there is no action on the packet because of the invalid IP.
- When this feature is enabled with the DISCARD option, the packets with invalid source IP address are discarded. The corresponding statistics are incremented accordingly.

Verifying IP Source Violation for Uplink Packets

Use this procedure to verify on UPF if the IP Source Violation feature is enabled or disabled on cnSGWc along with the configured values.

Procedure

Enter the **show subscribers user-plane-only full all | callid <>**.

Example:

```
show subscribers user-plane-only full all | callid <>
```

A new counter **ip source violations ignored:<>** is added to show the number of IP source violated packets that have been ignored and forwarded.

The following is an example output of the **show subscribers user-plane-only full all | callid <>** CLI command where the values for **IP Source Violation Action**, **ip source violations**, and **ip source violations ignored** are configured.

```
show subscribers user-plane-only full all
  Local SEID      : [0x0004000000000000] 1125899906842624
  Remote SEID     : [0x0004000000000005] 1125899906842629
  State           : Connected
  Connect Time    : Wed Jun  5 05:09:22 2024
  Idle time       : 00h11m19s
  Access Type: n/a                      Network Type: IP
  user-plane-service-name: UP21
  active-service-scheme-name:
  Callid: 00004e21
  Rulebase:
  Interface Type: Sxa
  eMPS Session: No
  eMPS Session Priority: 0
  Session-Type: normal
  Precedence-order: 0
  Data Queue: 0
  Card/Cpu: 1/0                         Sessmgr Instance: 1
  IP address: n/a
  Next Hop Ip Address: n/a
  Source context: EPC2-UP               Destination context: EPC2-UP
  PDN-Instance: n/a
  User-plane-Sx-addr: 20.20.20.106
  Control-plane-Sx-addr: 20.20.20.101
  Number of associated PDRs : 2
  Number of associated ADC PDRs : 0
  Number of associated FARs : 2
```

```
Number of associated QERs : 0
Number of associated BARs : 0
Number of associated URRs : 1
Uplink APN AMBR (bps) : 0       Downlink APN AMBR (bps) : 0
active input acl: n/a                active output acl: n/a
active input ipv6 acl: n/a             active output ipv6 acl: n/a
Bandwidth Policy: n/a
FW-and-NAT Policy: n/a
FW-and-NAT Policy ID: n/a
Firewall Policy IPv4: n/a
Firewall Policy IPv6: n/a
NAT Policy NAT44: n/a
NAT Policy NAT64: n/a
Converged Session: No                 Converged Peer Callid:      n/a
Visited Call: No
IP Source Violation Action: DISCARD
Subscriber Parameters:
IMSI: 404005123456789
IMEI: 112233445566778
MSISDN: 9890098900
Rat Type: 6
Old Rat Type: n/a
CF Policy ID: n/a
GX CF Policy ID: n/a
Roaming Status: n/a
Old Roaming Status: n/a
input pkts: 5                                 output pkts: 0
input bytes: 250                              output bytes: 0
input bytes dropped: 123                      output bytes dropped: 0
input pkts dropped: 3                         output pkts dropped: 0
CF Buffered Uplink Packets: 0                 CF Buffered Downlink Packets: 0
CF Buffered Uplink Bytes: 0                   CF Buffered Downlink Bytes: 0
Uplink Packets in Buffer: 0                   Downlink Packets in Buffer: 0
Buff Over-limit Uplink Pkts: 0                Buff Over-limit Downlink Pkts: 0
DDN buffered pkts : 0                         DDN buffered bytes : 0
DDN buffer overflow drop pkts : 0             DDN buffer overflow drop bytes : 0
pk rate from user(bps): 0                     pk rate to user(bps): 0
ave rate from user(bps): 0                    ave rate to user(bps): 0
sust rate from user(bps): 0                   sust rate to user(bps): 0
pk rate from user(pps): 0                     pk rate to user(pps): 0
ave rate from user(pps): 0                    ave rate to user(pps): 0
sust rate from user(pps): 0                   sust rate to user(pps): 0
ipv4 bad hdr: 0                               ipv4 ttl exceeded: 0
ipv4 fragments sent: 0                        ipv4 could not fragment: 0
ipv4 bad length trim: 0
ipv4 input mcast drop: 0                      ipv4 input bcast drop: 0
input pkts dropped (0 mbr): 0                 output pkts dropped (0 mbr): 0
ipv4 input acl drop: 0                        ipv4 output acl drop: 0
ipv6 input acl drop: 0                        ipv6 output acl drop: 0
ip source violations: 3                       ipv4 output no-flow drop: 0
ipv6 bad hdr: 0                               ipv6 bad length trim: 0
ip source violations ignored: 0
ipv4 icmp packets dropped: 0
APN AMBR Uplink Pkts Drop: 0                   APN AMBR Downlink Pkts Drop: 0
APN AMBR Uplink Bytes Drop: 0                  APN AMBR Downlink Bytes Drop: 0
APN AMBR Uplink Pkts IP pref lowered: 0        APN AMBR Downlink Pkts IP pref lowered: 0

APN AMBR Uplink Bytes IP pref lowered: 0       APN AMBR Downlink Bytes IP pref lowered: 0

ITC Uplink Pkts Drop: 0                        ITC Downlink Pkts Drop: 0
ITC Uplink Bytes Drop: 0                       ITC Downlink Bytes Drop: 0
ITC Uplink Pkts IP pref lowered: 0             ITC Downlink Pkts IP pref lowered: 0
ITC Uplink Bytes IP pref lowered: 0            ITC Downlink Bytes IP pref lowered: 0
ITC Terminated Flows:: 0                       ITC Redirected Flows: 0
```

```
  Flow Action Terminated Flows: 0                  Flow Action Redirected Flows: 0
  ToS marked Uplink Pkts: 0                        ToS marked Downlink Pkts: 0
  CC Dropped Uplink Pkts: 0                        CC Dropped Downlink Pkts: 0
  CC Dropped Uplink bytes: 0                       CC Dropped Downlink Bytes: 0
  Uplink Inflight Pkts: 0                          Downlink Inflight Pkts: 0
  QFI Mismatch Uplink Pkts Drop: 0                 QFI Mismatch Downlink Pkts Drop: 0
  Total QoS-Group(s) Active: 0
  DNS-to-EDNS Uplink Pkts: 0                     DNS-to-EDNS Uplink Bytes: 0
  EDNS Response Received: 0

  Flow information:
    Current Active Flows:
      TCP: 0
      UDP: 0
    Total Flows:
      TCP: 0
      UDP: 0
      FP:  0

Static & Predef Rule Match stats:
Rule Name             Pkts-Down  Bytes-Down Pkts-Up    Bytes-Up   Hits       Match-Bypassed
FP-Down(Pkts/Bytes)  FP-Up(Pkts/Bytes)
--------------------- ---------- ---------- ---------- ---------- ---------- --------------
-------------------  ------------------

Dynamic Rule Match stats:
PDR Id   Rule Name            Pkts-Down  Bytes-Down Pkts-Up    Bytes-Up   Hits       Match-Bypassed
  FP-Down(Pkts/Bytes)  FP-Up(Pkts/Bytes)
-------- -------------------- ---------- ---------- ---------- ---------- ---------- --------------
  -------------------  ------------------
0x0002   N/A                         0          0          5        250          0              0
      0/0               5/250

Post-Processing Rule Match stats:
Rule Name             Pkts-Down  Bytes-Down Pkts-Up    Bytes-Up   Hits       Match-Bypassed
FP-Down(Pkts/Bytes)  FP-Up(Pkts/Bytes)
--------------------- ---------- ---------- ---------- ---------- ---------- --------------
-------------------  ------------------

Firewall-NAT Access Rule Match stats:
Firewall-Ruledef Name Pkts-Down Bytes-Down    Pkts-Up   Bytes-Up Dr-Pkts-Dn Dr-Byts-Dn Dr-Pkts-Up
Dr-Byts-Up       Hits
--------------------- ---------- ---------- ---------- ---------- ---------- ---------- ----------
---------- ----------

QoS-Group Statistics:
QGR Name              Pkts-Down  Bytes-Down Pkts-Up    Bytes-Up   Hits       Match-Bypassed
FP-Down(Pkts/Bytes)  FP-Up(Pkts/Bytes)
--------------------- ---------- ---------- ---------- ---------- ---------- --------------
-------------------  ------------------

SGACL Match stats:
ACL Name              Pkts-Down  Bytes-Down Pkts-Up    Bytes-Up   Pkts dropped
--------------------- ---------- ---------- ---------- ---------- -------------

  CIoT APN Rate Control:
    Allowed UL Limit:   0                        Allowed DL limit:    0
    Remaining UL Limit: 0                        Remaining DL limit:  0
    Allowed Time unit: unrestricted              Status Validity Time: N/A

Total subscribers matching specified criteria: 1
```

URL Blockedlisting

Feature Description

The URL blockedlisting feature regulates the subscriber's access to view or download content from websites whose URL or URI has been blockedlisted. It uses a database that records a list of URLs that indicates if the detected URL is categorized to be blocked or not.

How it Works

To enable the URL blockedlisting feature on UPF, URL blockedlisting database should be present with a name "optblk.bin" under flash, or SFTP or under its subdirectory. This database directory path must be configured on user-plane, after user-plane services are brought up.

HTTP Analyzer must be enabled for URL blockedlisting. The HTTP analyzer extracts URL information from the incoming HTTP request data packet. Extracted URL content is compared with the URL Blockedlisting database. When the URL of incoming HTTP data packet matches with the database URL entry, that URL is treated as blockedlisted URL and one of the following actions takes place on that HTTP packet:

- Termination of flow
- Packet is discarded

The URL blockedlisting configurations must be configured under Rulebase configuration in Active Charging Service. Also, two URL blockedlisting methods – Exact and Generic, are supported at Active Charging Service-level configuration.



Important Blockedlisting database(s) are provided by – Internet Watch Foundation (IWF) and National Center for Missing and Exploited Children (NCMEC). The UPF always receives the blockedlisting database in Optimized Format.

URL Blockedlisting Database Upgrade

URL database upgrade is supported in following two ways:

- Timer-based upgrade or Auto upgrade
- CLI-based upgrade or Manual upgrade

Timer-based or Auto-upgrade

After the database is loaded on the chassis for the first time, a timer, for a duration of 5 minutes, is started. This process is started to auto upgrade the database.

If at the expiry of the timer, a valid database with higher version is available at the directory path, then database upgrade procedure is initiated, and a newer version of the database is loaded on the UPF.

To upgrade a URL blockedlisting database, a higher version of valid URL Blockedlisting database with name "optblk_f.bin" should be present at same directory as that of current database "optblk.bin".

After the database is upgraded successfully, the earlier "optblk.bin" file gets renamed as "optblk_0.bin" and "optblk_f.bin" file gets renamed as "optblk.bin". Here, "optblk_0.bin" file is treated as a backup file of older database.

If an additional upgrade is performed, then “optblk_0.bin” file will be renamed as “optblk_1.bin” file and current “optblk.bin” will get renamed as “optblk_0.bin”, and so on.

See the *Loading URL Blockedlisting Database on UPF* section to configure the number of backup files to be stored in the database.

CLI-based or Manual Upgrade

See the *Upgrading the URL Blockedlisting Database* section to upgrade the current database to a newer version.

Configuring URL Blockedlisting

Loading URL Blockedlisting Database on UPF

Use the following configuration to load URL blockedlisting database on UPF.

In releases prior to 2022.01.0:

```
configure
  url-blacklisting database directory path database_directory_path
  url-blacklisting database max-versions max_version_value
  end
```

From 2022.01.0 and later releases:

```
configure
  url-blockedlisting database directory path database_directory_path
  url-blockedlisting database max-versions max_version_value
  end
```

NOTES:

- **database directory path**: Configures the database directory path.

  The *database_directory_path* is a string of size 1 to 255.

- **max-versions**: Configures the maximum database upgrade versions.

  The *max_version_value* is an integer from 0 to 3.

Upgrading the URL Blockedlisting Database

Use this configuration to manually upgrade the URL blockedlisting database.

In releases prior to 2022.01.0:

```
upgrade url-blacklisting database
end
```

From 2022.01.0 and later releases:

```
upgrade url-blockedlisting database
end
```

Configuration to Enable URL Blockedlisting

Use the following configuration to enable URL blockedlisting feature on UPF.

In releases prior to 2022.01.0:

```
configure
  active-charging service service_name
    url-blacklisting match-method [ exact | generic ]
    rulebase rulebase_name
      url-blacklisting action [ discard | terminate-flow ]
      end
```

From 2022.01.0 and later releases:

```
configure
  active-charging service service_name
    url-blockedlisting match-method [ exact | generic ]
    rulebase rulebase_name
      url-blockedlisting action [ discard | terminate-flow ]
      end
```

NOTES:

- **match-method [exact | generic]**: Specifies the match method used for URL blockedlisting.
  - **exact**: URL Blockedlisting perform an exact-match of URL.
  - **generic**: URL Blockedlisting perform generic-match of URL.
- **url-blockedlisting action [discard | terminate-flow]**:
  - **discard**: Discards the HTTP packet received.
  - **terminate-flow**: Terminates the flow of the HTTP packet received.

Monitoring and Troubleshooting

This section provides information regarding the CLI command available in support of monitoring and troubleshooting the feature.

Show Command(s) and/or Outputs

This section provides information regarding show commands and/or their outputs in support of this feature.

show user-plane-service url-blacklisting database

The following fields are displayed in support of this feature:

- URL Blacklisting Static Rating Databases:
  - Last Upgrade Status
  - Path
    - Database Status
    - Number of URLs in DB
    - Type
    - Version
    - Creation Time

- Hostname
- Comment
- Last Access Time
- Last Modification Time
- Last Status Change Time

show user-plane-service url-blacklisting database url database_directory_path

The following fields are displayed in support of this feature:

- URL Blacklisting Static Rating Databases:
  - Last Upgrade Status
  - Path
    - Database Status
    - Number of URLs in DB
    - Type
    - Version
    - Creation Time
    - Hostname
    - Comment
    - Last Access Time
    - Last Modification Time
    - Last Status Change Time

show user-plane-service url-blacklisting database facility sessmgr all

The following fields are displayed in support of this feature:

- URL-Blacklisting SessMgr Instance Based Database Configuration
  - SessMgr Instance
  - BL DB Load Status
  - BL DB Version
  - Number of URLs
  - Checksum

show user-plane-service rulebase name rulebase_name

The following fields are displayed in support of this feature:

- URL-Blacklisting Action
- URL-Blacklisting Content ID

show user-plane-service inline-services info

The following fields are displayed in support of this feature:

- URL-Blacklisting: Enabled
  - URL-Blacklisting Match-method: Generic

show user-plane-service inline-services url-blockedlisting statistics

The following are displayed in support of this feature:

- Cumulative URL-Blockedlisting Statistics
  - Blockedlisted URL hits
  - Blockedlisted URL misses
  - Total rulebases matched

show user-plane-service inline-services url-blacklisting statistics rulebase name rulebase_name

The following fields are displayed in support of this feature:

- Rulebase Name
  - URL-Blacklisting Statistics
  - Blacklisted URL hits
  - Blacklisted URL misses
- Total rulebases matched

Configuring the Static and Pre-Defined Rules

This section describes how to configure the static and pre-defined rules under the charging action configuration.

```
configure
   active-charging service service_name
      charging-action charging_action_name
         flow action { discard [ downlink | uplink ] | redirect-url
redirect_url | terminate-flow }
         end
```

NOTES:

- **flow action**: Specifies the action to take on packets that match rule definitions.
  - **discard [downlink | uplink]**: Specifies to discard downlink or uplink packets.
  - **redirect-url** *redirect_url*: Specifies the URL to be redirected. For example, http://search.com/subtarg=#HTTP.URL#
  - **terminate-flow**: Specifies to terminate the flow.
- For redirect-url, configure HTTP analyzer under rulebase. Example:

```
route priority 70 ruledef http-port analyzer HTTP
   ruledef http-port
   tcp either-port = 80
   rule-application routing
   exit
```

Configuring ACS Ruledef for L7 Protocols for DPI

A ruledef represents a set of matching conditions across multiple L3 – L7 protocol based on protocol fields and state information. Each ruledef can be used across multiple rulebases within the active charging service.

Note In UPF, if rule-line addition or deletion inside a ruledef is done during active calls and data flows, then this configuration change is not applied for current flows. However, the configuration change applies to new calls and new flows on same calls.

The following is a sample configuration that describes how to create, configure, or delete ACS rule definitions.

```
configure
   active-charging service service_name
      ruledef ruledef_name
         dns { any-match value | query-type query_type | query-name query_name
 }
         ip any-match [ = | != ] [ TRUE | FALSE ]
         ip dst-address { operator { { ipv4_address | ipv6_address } | {
ipv4_address/mask | ipv6_address/mask} | address-group ipv6_address } | { !range |
 range } host-pool host_pool_name }
         ip server-ip-address { operator { { ipv4_address | ipv6_address } | {
ipv4_address/mask | ipv6_address/mask} | address-group ipv6_address } | { !range |
 range } host-pool host_pool_name }
         multi-line-or all-lines
         rule-application { charging | post-processing | routing }
         { tcp | udp } { either-port port_number }
         end
```

NOTES:

- **ruledef** *ruledef_name*: Specifies the ruledef to add, configure, or delete. *ruledef_name* must be the name of an ACS ruledef, and must be an alphanumeric string of 1 to 63 characters, and can contain punctuation

characters. Each ruledef must have a unique name. Host pool, port map, IMSI pool, and firewall, routing, and charging ruledefs must have unique names.

- If the named ruledef does not exist, it is created, and the CLI mode changes to the ACS Ruledef Configuration Mode wherein the ruledef can be configured.
- If the named ruledef already exists, the CLI mode changes to the ACS Ruledef Configuration Mode for that ruledef. The ACS Ruledef Configuration Mode is used to create and manage rule expressions in individual rule definitions (ruledefs).
- **ip any-match [= | !=] [TRUE | FALSE]:** This command defines the rule expressions to match IPv4/IPv6 packets. The *operator* and *condition* in the command specifies the following:
  - *operator*
    - !=: Does not equal
    - < =: Equals
  - *condition*
    - FALSE
    - TRUE
- **ip dst-address {** *operator* **{ {** *ipv4_address* **|** *ipv6_address* **} | {** *ipv4_address/mask* **|***ipv6_address/mask* **} | address-group** *ipv6_address* **} | { !range | range } host-pool** *host_pool_name* **}**: This command allows defining rule expressions to match IP destination address field within IP headers.
  - *ipv4_address* **|** *ipv6_address*: Specifies the IP address of the destination node for outgoing traffic. *ipv4_address* **|** *ipv6_address* must be an IP address in IPv4 dotted-decimal or IPv6 colon-separated-hexadecimal notation.
  - *ipv4_address/mask* **|** *ipv6_address/mask*: Specifies the IP address of the destination node for outgoing traffic. *ipv4_address/mask* **|** *ipv6_address/mask* must be an IP address in IPv4 dotted-decimal or IPv6 colon-separated-hexadecimal notation with subnet mask bit. The mask bit is a numeric value which corresponds to the number of bits in the subnet mask.
  - *address-group ipv6_address*: Specifies a group of IPv6 addresses configured with wildcard input and/or specialized range input. Multiple wildcard characters can be accepted as input and only one 2 byte range input will be accepted. Both wildcard character input and 2-byte range input can be configured together within a given IPv6 address.
  - **host-pool** *host_pool_name*: Specifies the name of the host pool. *host_pool_name* must be an alphanumeric string of 1 to 63 characters.
  - The *operator* in the command specifies the following:
    - !=: Does not equal
    - <: Lesser than or equals
    - =: Equals
    - >=: Greater than or equals

- **multi-line-or all-lines**: This command allows a single ruledef to specify multiple URL expressions. When a ruledef is evaluated, if the multi-line-or all-lines command is configured, the logical OR operator is applied to all the rule expressions in the ruledef to decide if the ruledef matches or not. If the multi-line-or all-lines command is not configured, the logical AND operator is applied to all the rule expressions.
- **rule-application { charging | post-processing | routing }**: This command specifies the rule application for a rule definition**.**
  - **charging**: Specifies that the current ruledef is for charging purposes.
  - **post-processing**: Specifies that the current ruledef is for post-processing purposes. This enables processing of packets even if the rule matching for them has been disabled.
  - **routing**: Specifies that the current ruledef is for routing purposes. Up to 256 rule definitions can be defined for routing in an Active Charging Service. Default: Disabled.
- **dns { any-match** *value* **| query-type** *query_type* **| query-name** *query_name* **}**: This command allows you to define rule expressions to match all DNS packets, or packets based on the query type or query name.

  ip server-ip-address *ip_address_value*: This command allows you to define rule expressions to match the IP address of the destination end of the connection.
- **{ tcp | udp } { either-port** *port_number* **}**: This command allows you to define rule expressions to match either a destination or source port number in UDP/TCP headers.

Charging Action Configuration for L7 Protocols for DPI

This section describes how to configure charging action. The charging action represents actions to be taken when a configured rule is matched. Actions could range from generating an accounting record (for example, an EDR) to dropping the IP packet, and so on. The charging action will also determine the metering principle—whether to count retransmitted packets and which protocol field to use for billing (L3, L4, L7, and so on).

The charging action configuration is used to define the QoS and charging related parameters associated with ruledefs.

```
configure
   active-charging service service_name
      charging-action charging_action
       allocation-retention-priority priority  [ pci pci_value  | pvi pvi_value

        billing-action egcdr
       cca charging credit [ rating-group coupon_id ] [ preemptively-request
 ]
        content-id content_id
        flow action { discard [ downlink | uplink ] | redirect-url
redirect_url | terminate-flow }
        flow limit-for-bandwidth { { direction { downlink | uplink }
peak-data-rate bps peak-burst-size bytes violate-action { discard |
lower-ip-precedence } [ committed-data-rate bps committed-burst-size bytes
 [ exceed-action { discard | lower-ip-precedence } ] ] } | { id id } }
```

```
        nexthop-forwarding-address ipv4_address/ipv6_address
        qos-class-identifier qos_class_identifier
        service-identifier service_id
        tft packet-filter packet_filter_name
        tft-notify-ue
        tos { af11 | af12 | af13 | af21 | af22 | af23 | af31 | af32 |
af33 | af41 | af42 | af43 | be | ef | lower-bits tos_value } [ downlink |
uplink ]
```

NOTES:

- **charging-action** *charging_action_name*: Specifies the name of a charging action. *charging_action_name* must be an alphanumeric string of 1 to 63 characters and can contain punctuation characters. Each charging action must have a unique name.
- If the named charging action does not exist, it is created, and the CLI mode changes to the ACS Charging Action Configuration Mode wherein the charging action can be configured.
- If the named charging action already exists, the CLI mode changes to the ACS Charging Action Configuration Mode for that charging action.
- **allocation-retention-priority** *priority* **[pci** *pci_value* **| pvi** *pvi_value* **]**: Configures the Allocation Retention Priority (ARP). *priority* must be an integer value in the range of 1-15.
  - **pci** *pci_value* : **Specifies the Preemption Capability Indication (PCI) value. The options are:**
    - MAY_PREEMPT - Flow can be preempted. This is the default value.
    - NOT_PREEMPT - Flow cannot be preempted
  - **pvi** *pvi_value*: Specifies the Preemption Vulnerability Indication (PVI) value. The options are:
    - NOT_PREEMPTABLE - Flow cannot be preempted. This is the default value.
    - PREEMPTABLE - Flow can be preempted
- **billing-action**: Configures the billing action for packets that match specific rule definitions.
- **cca charging credit**: Enables or disables credit control charging credit behaviour.
- **content-id**: Configures the rating group.
- **flow action**: Specifies the action to take on packets that match rule definitions.
  - **discard [downlink | uplink]**: Specifies to discard downlink or uplink packets.
  - **redirect-url** *redirect_url*: Specifies the URL to be redirected.
  - **terminate-flow**: Specifies to terminate the flow.
- **flow limit-for-bandwidth**: Configures the QoS parameters such as MBR, GBR, and so on.
  - peakdatarate(MBR): Default is 3000 bps
  - peakburstsize: Default is 3000 bytes
  - committedDataRate(GBR): Default is 144000 bps
  - committedBurstSize: Default is 3000 bytes

- **nexthop-forwarding-address** *ipv4_address/ipv6_address* ,: Configures the nexthop forwarding address.
- **qos-class-identifier** *qos_class_identifier* : Configures the QCI for a charging action. *qos_class_identifier* must be an integer value in the range of 1-9 or from 128-254 (Operator specific).
- **service_identifier** *service_id*: Configures the service identifier to use in generated billing records.*service_id* must be an integer value in the range of 1-2147483647.
- **tft packet-filter** *packet_filter_name*: Specifies the packet filter to add or remove from the current charging action. *packet_filter_name* must be the name of a packet filter, and must be an alphanumeric string of 1 to 63 characters.
- **tft-notify-ue**: Control the TFT updates towards the UE based on certain trigger conditions.
- **tos**: Configures the Type of Service (ToS) octets.