Deep Packet Inspection and Inline Services

Feature Summary and Revision History

Summary Data

Applicable Product(s) or Functional Area	5G-UPF
Applicable Platform(s)	VPC-SI
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**
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

Revision Details	Release
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).

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

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

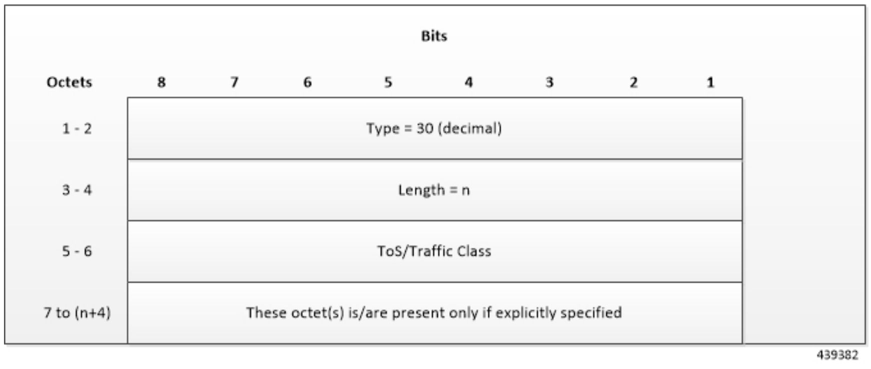
The SMF controls the transport-level marking by providing the DSCP in the ToS (IPv4) or Traffic Class (IPv6) within the "Transport Level Marking" IE in the FAR, that is associated to the PDR matching the traffic to be marked. The UPF performs the transport level marking for the detected traffic and sends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

The UPF also supports the inner packet marking in which it marks the tunnel packets. As the 3GPP specification does not determine any specific IE, the UPF uses a private IE named "Inner Packet Marking".

In addition, there is also a provision to copy the DSCP of inner packet to the outer IP header. As the 3GPP specification does not determine any specific IE, the UPF uses a private IE named "Transport Level Marking Options".

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.

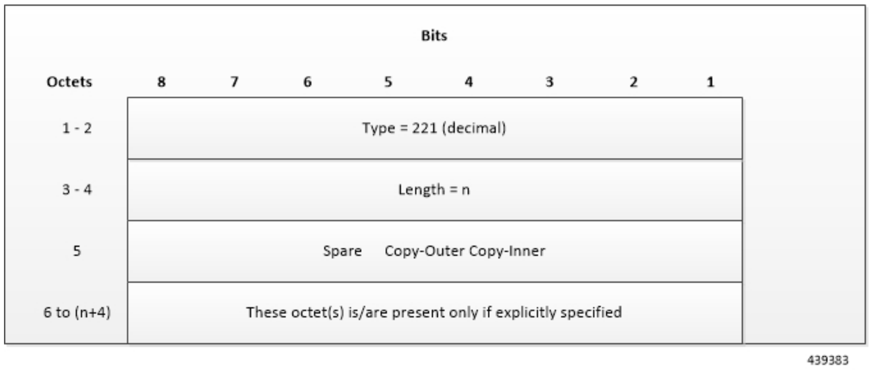


Octets	Bits 8 7 6 5 4 3 2 1
1 - 2	Type = 30 (decimal)
3 - 4	Length = n
5 - 6	ToS/Traffic Class
7 to (n+4)	These octet(s) is/are present only if explicitly specified

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.

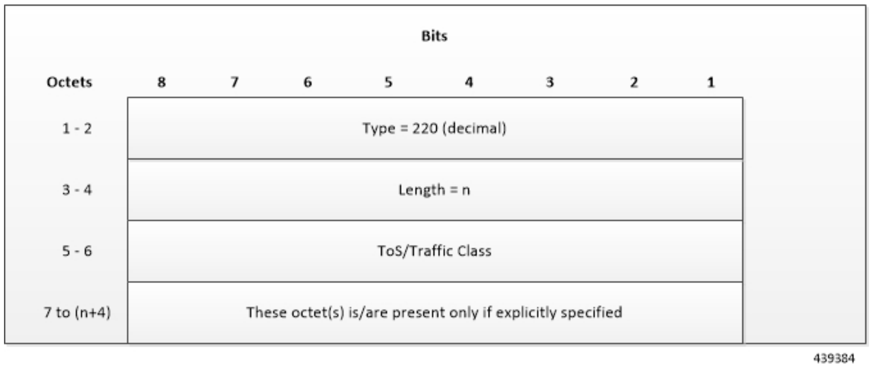


Octets	Bits 8 7 6 5 4 3 2 1
1 - 2	Type = 221 (decimal)
3 - 4	Length = n
5	Spare Copy-Outer Copy-Inner
6 to (n+4)	These octet(s) is/are present only if explicitly specified

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.



Octets	Bits 8 7 6 5 4 3 2 1
1 - 2	Type = 220 (decimal)
3 - 4	Length = n
5 - 6	ToS/Traffic Class
7 to (n+4)	These octet(s) is/are present only if explicitly specified

439384

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.

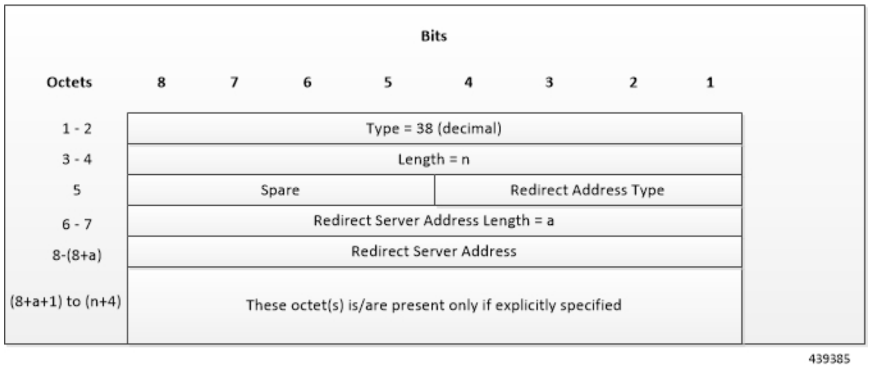
To enforce the traffic redirection in the UPF, the SMF takes the following actions:

- 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:



Octets	Bits 8	7	6	5	4	3	2	1
1 - 2	Type = 38 (decimal)							
3 - 4	Length = n							
5	Spare				Redirect Address Type			
6 - 7	Redirect Server Address Length = a							
8-(8+a)	Redirect Server Address							
(8+a+1) to (n+4)	These octet(s) is/are present only if explicitly specified							

439385

"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.

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

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 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 following packets are recorded to calculate and record 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:

- attribute sn-start-time
- attribute sn-end-time
- attribute sn-start-time format MM/DD/YYYY-HH:MM:SS:sss
- attribute sn-end-time format MM/DD/YYYY-HH:MM:SS:sss
- attribute radius-calling-station-id
- attribute radius-called-station-id
- rule-variable bearer 3gpp imsi
- rule-variable bearer 3gpp imei
- rule-variable bearer 3gpp rat-type
- rule-variable bearer 3gpp user-location-information
- rule-variable ip subscriber-ip-address
- rule-variable ip dst-address
- attribute sn-ruledef-name
- attribute sn-subscriber-port
- attribute sn-server-port

- attribute sn-app-protocol
- attribute sn-volume-amt ip bytes uplink
- attribute sn-volume-amt ip bytes downlink
- attribute sn-flow-start-time format seconds
- attribute sn-flow-end-time format seconds
- attribute sn-volume-amt ip pkts uplink
- attribute sn-volume-amt ip pkts downlink
- attribute sn-direction
- rule-variable traffic-type
- rule-variable p2p protocol
- rule-variable p2p app-identifier tls-cname
- rule-variable p2p app-identifier tls-sni
- rule-variable p2p app-identifier quic-sni
- rule-variable bearer 3gpp sgsn-address
- attribute sn-rulebase
- attribute sn-charging-action
- rule-variable flow tethered-ip-ttl
- rule-variable flow ttl
- rule-variable flow ip-control-param
- rule-variable bearer qci
- rule-variable tcp flag
- rule-variable ip server-ip-address
- attribute sn-flow-id
- attribute sn-closure-reason
- attribute sn-duration
- rule-variable ip src-address
- rule-variable ip protocol
- attribute sn-charge-volume ip bytes uplink
- attribute sn-charge-volume ip bytes downlink

The following HTTP EDR attributes are supported:

- rule-variable http url length 2000
- rule-variable http request method

- rule-variable http content type
- rule-variable http user-agent length 255
- rule-variable http reply code
- rule-variable http referer
- rule-variable http host
- rule-variable http cookie
- rule-variable http header-length
- attribute transaction-uplink-bytes
- attribute transaction-downlink-bytes

The following DNS EDR attributes are supported:

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

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.

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
        end
```

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 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 session manager creates an EDR with the specified format name for every flow that has had any activity since 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.

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
```

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

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

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 supports FTP proocol as part of L7 Analyzer.

HTTP

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

- Content-Length – Volume-based offloading is supported for methods like 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 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. Or, in case of features like x-header, redirection is configured which has dependencies on some of the 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.

URL-Based Redirection

For flow action redirect-url, encrypt is not supported. Currently, the following dynamic fields are supported:

- #HTTP.URI#
- #HTTP.HOST#
- #HTTP.URL#
- #ACSMGR_BEARER_CALLED_STATION_ID#
- #RULEBASE#
- #RTSP.URI#

X-Header Insertion

X-header Insertion is supported in HTTP Requests. Note that:

- Flows, for which X-header is inserted in a packet, are not offloaded.
- With X-header configuration, all TCP OOO packets irrespective of transmit order CLI, will be buffered and sent out after reordering.

Limitation

- X-Header Spoofing is not supported.
- X-Header Insertion in Response packet is not supported.
- X-Header Encryption with RSA and RC4MD5 is supported but not supported with AES.
- Monitor protocol for X-Header is not supported.
- Following X-Header fields insertion is not supported in a packet:
    - QoS
    - UIDH
    - Customer ID
    - Hash Value
    - Time of the Day
    - RADIUS String
    - Session-Id
    - Congestion Level
    - User-Profile

HTTPS

The UPF supports HTTPS protocol as part of L7 Analyzer.

RTP/RTSP

The UPF 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 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
```

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 serviceservice_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*

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.