PCRF Simulator

Introduction

This document describes the PCRF (Policy and Charging Rule Function) simulator, based on Diameter Sd Interface (a 3GPP standardized interface).

The PCRF simulator implements a limited PCRF server which main purpose is to support testing using PureLoad of:

- Enterprise Enterprise Bring Your Own Device (BYOD) support in MSP
- Traffic Detection Function (TDF) in MSP
- Diameter Gx in MSP

Based on the Diameter Sd and Gx interface.

Diameter Support

The Diameter Base Protocol Stack supports the parts of IETF RFC 3588 that is required to support Diameter Sd Application and Gx Application.

Connection Establishment

The PCRF simulator supports standard TCP connection establishment including receiving Diameter CER messages and returning CEA messages. The connection is initiated by the MSP.

Details regarding TCP connections (IP addresses and ports) CEA messages (such as Origin-Host AVP) will be controlled using configuration files.

Connection Monitoring

The PCRF simulator supports receiving standard DWR messages and returning DWA messages.

Limitations

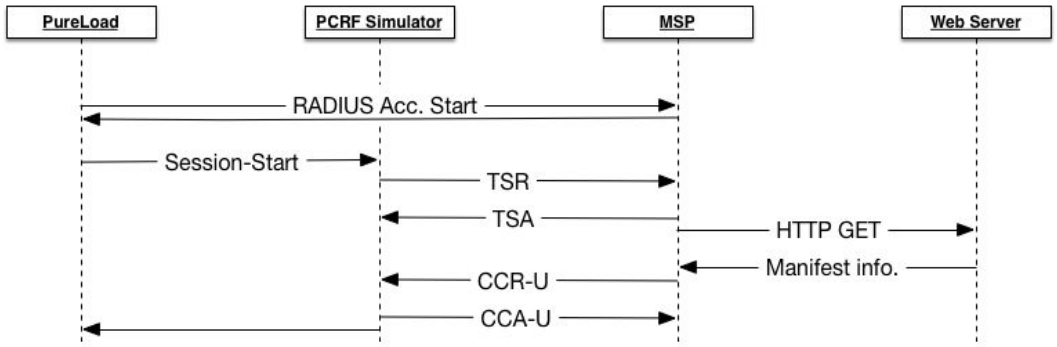
The Diameter Sd support i based on 3GPP TS 29.212 version 12 and Gx support is based on 3GPP TS 29.214 version 9. But note that all AVPs in later versions of the specifications might not be fully supported.

Simulation Call Flows

BYOD Call Flows

User Session Establishment

Session establishment is simulated by PureLoad sending a “Session-Start” custom HTTP POST message to the PCRF simulator:



The “Session-Start” message is a custom HTTP POST based message (described later) that must be sent after the RADIUS message to initiate a Sd session. The “Session-Start” message must be generated by PureLoad using a standard HTTP task: HttpPostContentTask.

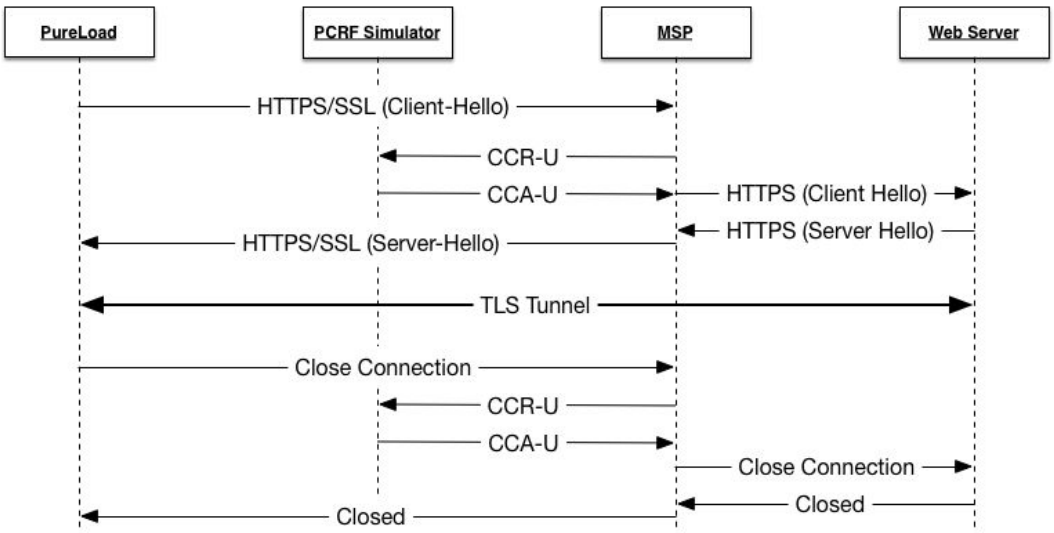
The HTTP POST request includes information in XML format that is used by the PCRF simulator to generate the TSR being sent. The XML data includes information about AVPs to be generated and added to the TSR.

When the simulator receives a TSA message information session information is stored (in memory) for the session (one per simulated UE).

The response time reported will give an indication of the time to get the BYOD manifest etc.

Transparent HTTPS - Session Update

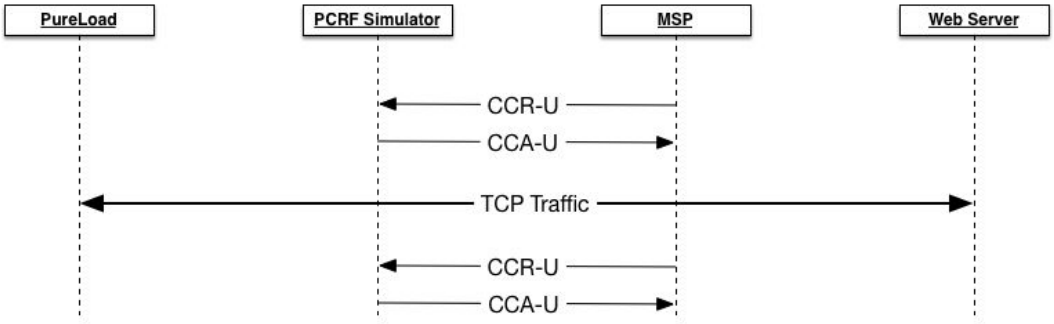
Session updates are initiated by MSP:



The PCRF simulator will in these cases only respond with a standard CCA-U message, indicating success. Not further action will be performed by the PCRF Simulator.

General TCP Traffic - Session Update

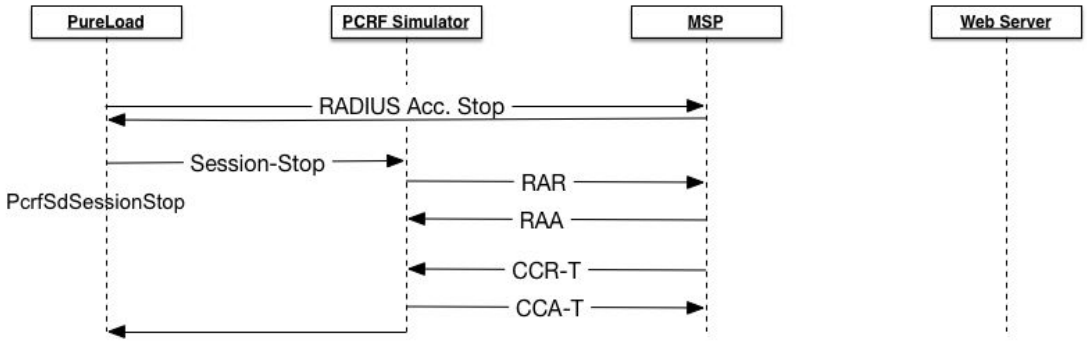
Session updates are initiated by MSP:



The PCRF simulator will in these cases only respond with a standard CCA-U message, indicating success. Not further action will be performed by the PCRF Simulator.

User Session Termination

Session termination is simulated by PureLoad sending a “Session-Stop” custom HTTP DELETE message to the PCRF simulator:

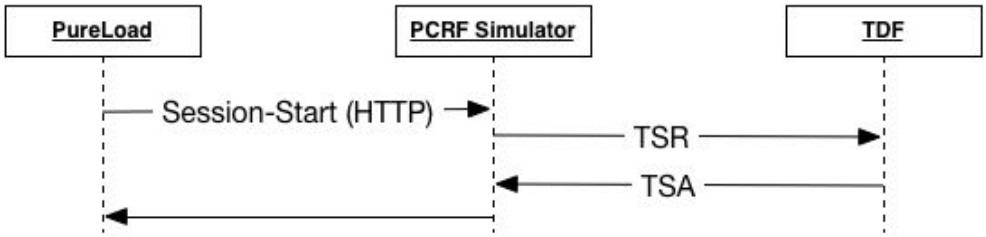


The “Stop” message is a custom HTTP DELETE based message (described later) that must be sent to remove a Sd session from the PCRF simulator. The “Session-Stop” message is generated by using a standard HTTP task: HttpDeleteTask.

TDF Call Flows

Session Establishment

Session establishment will be simulated by PureLoad sending a “Session-Start” custom HTTP POST message to the PCRF simulator:



The “Session-Start” message must be generated by PureLoad using a standard HTTP task: HttpPostContentTask.

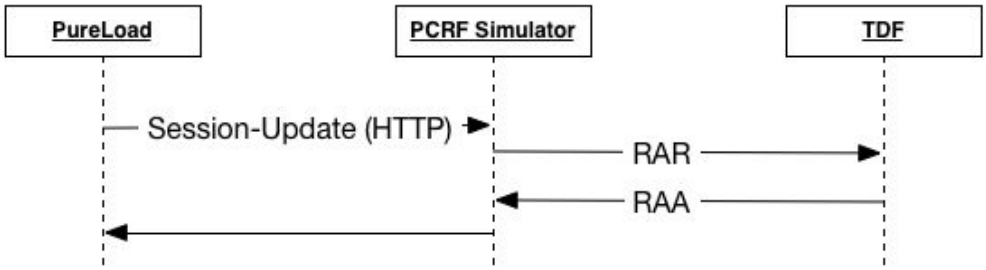
The HTTP POST request includes information in XML format that is used by the PCRF simulator to generate the TSR being sent. The XML data includes information about AVPs to be generated and added to the TSR.

When the simulator receives a TSA message information session information is stored (in memory) for the session (one per simulated UE).

Note: The only difference from BYOD Session Establishment is that the PCRF Simulator will not wait for a CCR-U before the HTTP response is returned. This is controlled by setting the XML element “Wait-For-CCR” to “false”.

Session Update - PCRF Initiated

PCRF initiated session updates will be simulated by PureLoad sending a “Session-Update” custom HTTP PUT message to the PCRF simulator:

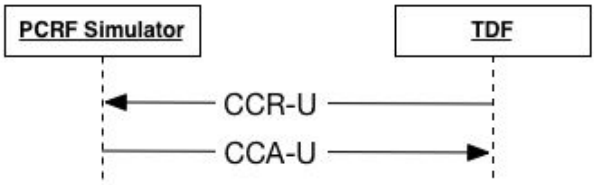


The “Session-Update” message must be generated by PureLoad using a standard HTTP task: HttpPutTask.

The HTTP PUT request includes information in XML format that is used by the PCRF simulator to generate the RAR being sent. The XML data includes information about AVPs to be generated and added to the RAR.

Session Update - TDF Initiated

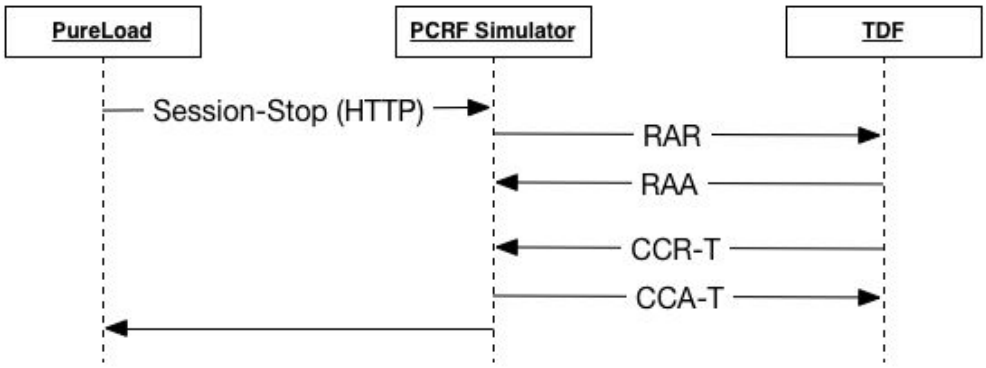
TDF session updates (CCR-U) will result in a (CCA-U) with optionally updated ADC rules as specified in configuration per MSISDN (or range of MSISDN).



The generated CCA-U can include AVPs that is controlled by the configuration of the PCRF simulator. Different AVPs can be specified, based on subscription IDs.

Session Termination

Session termination will be simulated by PureLoad sending a “Session-Stop” custom HTTP DELETE message to the PCRF simulator:

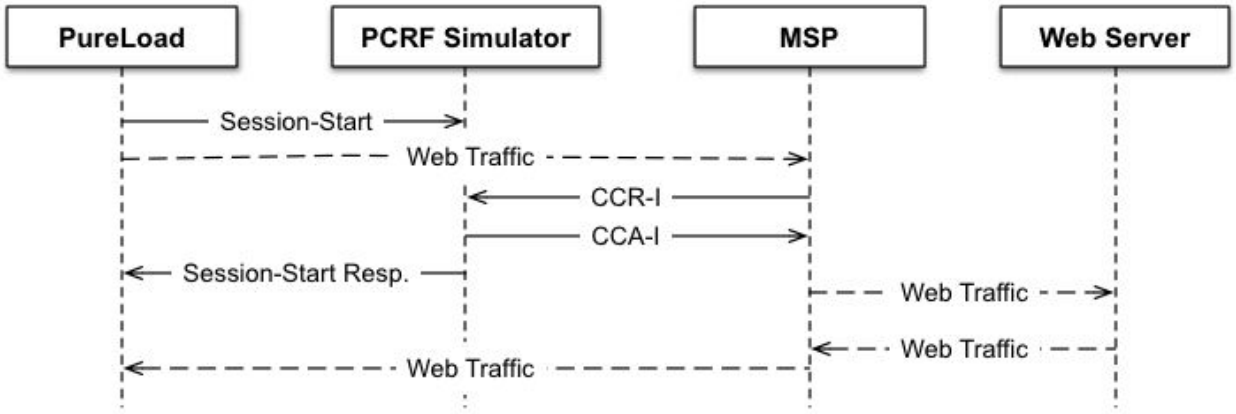


Note: this is the same asf or BYOD, but provided here for reference.

Gx Call Flows

Session Establishment

Session establishment will be simulated by PureLoad sending a “Session-Start” custom HTTP POST message to the PCRF simulator:



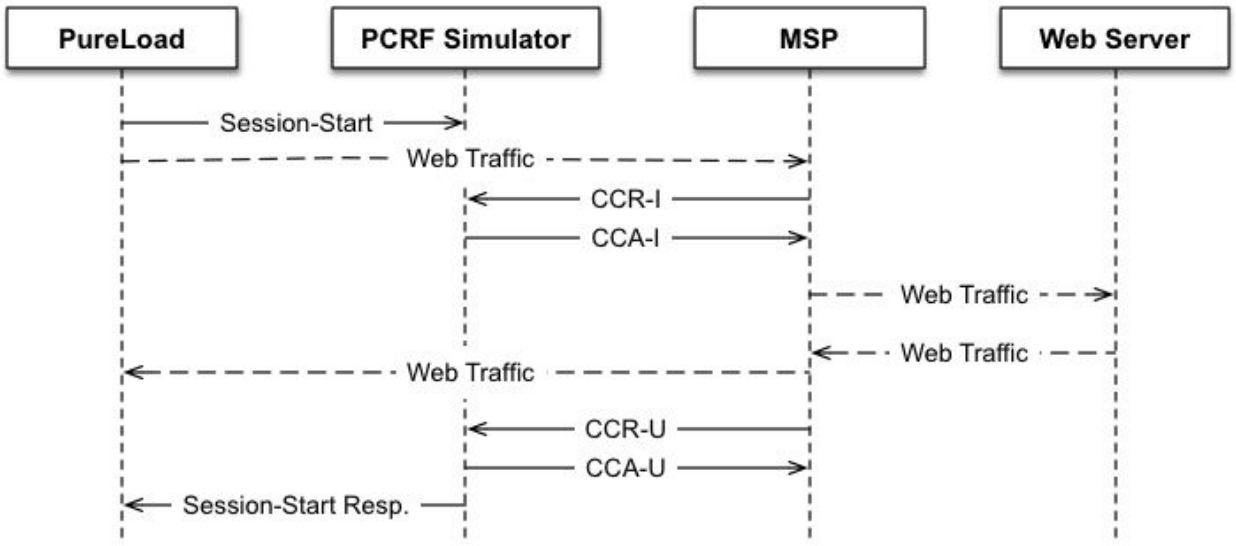
The “Session-Start” message must be generated by PureLoad using an asynchronous HTTP task (to be able to send web traffic before the response is received)

The HTTP POST request includes information in XML format that is used by the PCRF simulator to generate the CCA-I being sent. The XML data includes information about AVPs (mainly Charging-Rule-Install) to be generated and added to the CCA-I.

Once PCRF Simulator sent the CCA-I to establish GX session, 200 OK is sent as response to Pureload. The received CCR-I is returned in the HTTP response body as text.

Session Establishment, with PCC Rule Pull

Optionally session establishment including an initial PPC Rule Pull can be simulated by PureLoad sending a “Session-Start” custom HTTP POST message with a “wait-for-ccr-u” query parameter to the PCRF simulator:



The “Session-Start” message must be generated by PureLoad using an asynchronous HTTP task (to be able to send web traffic before the response is received)

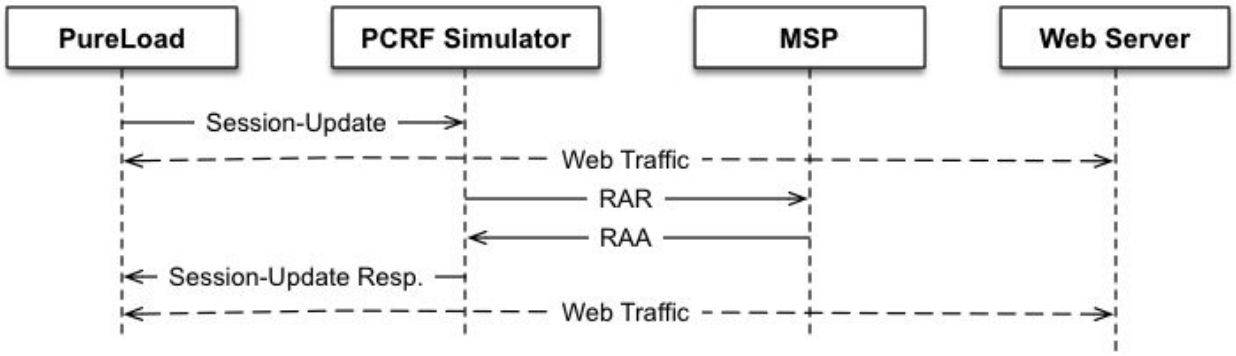
The HTTP POST request includes information in XML format that is used by the PCRF simulator to generate the CCA-I being sent. The XML data includes information about AVPs (mainly Charging-Rule-Install) to be generated and added to the CCA-I.

The generated CCA-U can include AVPs (mainly Charging-Rule-Install and Charging-Rule-Remove) that is controlled by the configuration of the PCRF simulator. Different AVPs can be specified, based on subscription IDs.

Once PCRF Simulator receives the CCR-U ant replies with CCA-U 200 OK is sent as response to Pureload. The received CCR-U is returned in the HTTP response body as text.

Session Update - PCC Rule Push

PCRF initiated session updates will be simulated by PureLoad sending a “Session-Update” custom HTTP PUT message to the PCRF simulator:



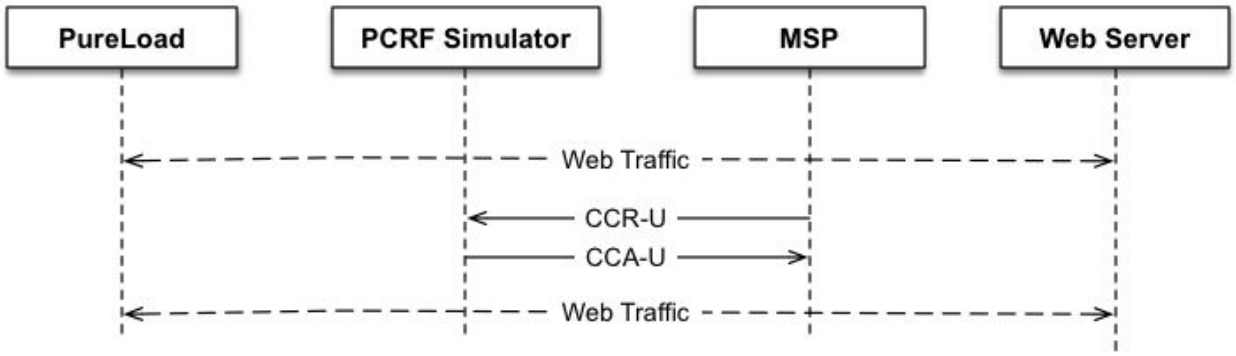
The “Session-Update” message must be generated by PureLoad using a HTTP task.

The HTTP PUT request includes information in XML format that is used by the PCRF simulator to generate the RAR being sent for the specified MSISDN. The XML data includes information about AVPs to be generated (mainly Charging-Rule-Install and Charging-Rule-Remove) and added to the RAR.

When PCRF Simulator receives RAA, 200 OK is sent as response to Pureload. The received RAA is returned in the HTTP response body as text.

Session Update - PCC Rule Pull

PCC Rule Pull is triggered by CCR-U from MSP, which will result in a CCA-U with optionally updated PCC rules as specified in configuration per MSISDN (or range of MSISDN).

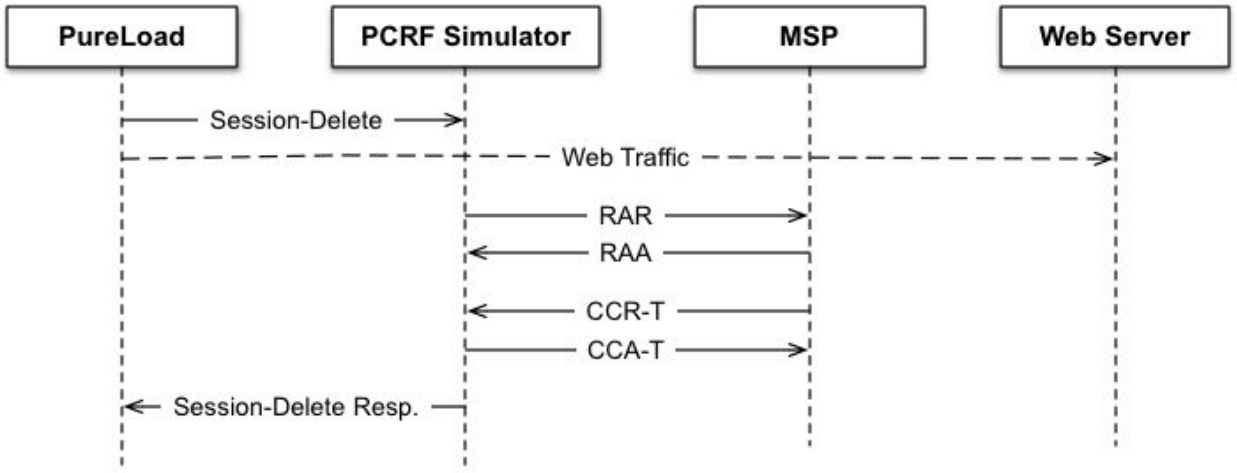


The generated CCA-U can include AVPs (mainly Charging-Rule-Install and Charging-Rule-Remove) that is controlled by the configuration of the PCRF simulator. Different AVPs can be specified, based on subscription IDs.

Any errors during the session update,such as invalid CCA-U received for unknown session, is logged and indicated in statistics log.

Session Termination Initiated By PCRF Simulator

Session termination will be simulated by PureLoad sending a “Session-Delete” custom HTTP DELETE message to the PCRF simulator:

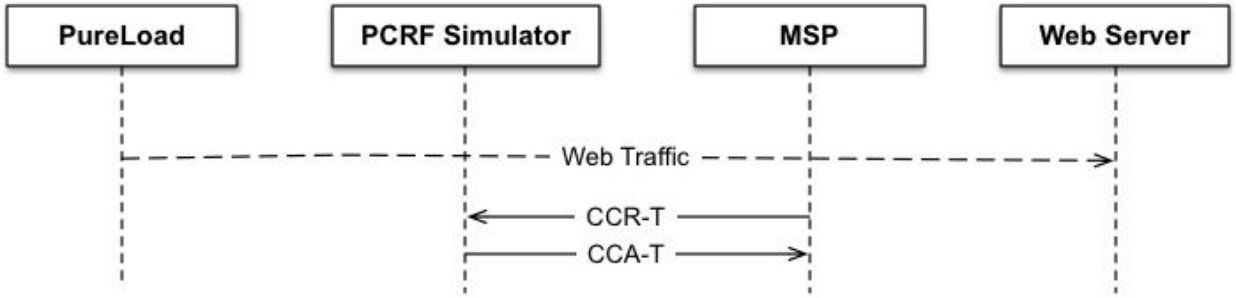


The HTTP DELETE message includes information in XML format that is used by PCRF simulator to generate the RAR being sent for the specific MSISDN. The XML data includes information about the AVPS to be generated and added to the RAR.

When PCRF Simulator receives CCA-T, 200 OK is sent as response to Pureload. The received CCA-T is returned in the HTTP response body as text.

Session Termination Initiated By MSP

PCRF Simulator will response to CCA-T for PCEF-initiated session termination.

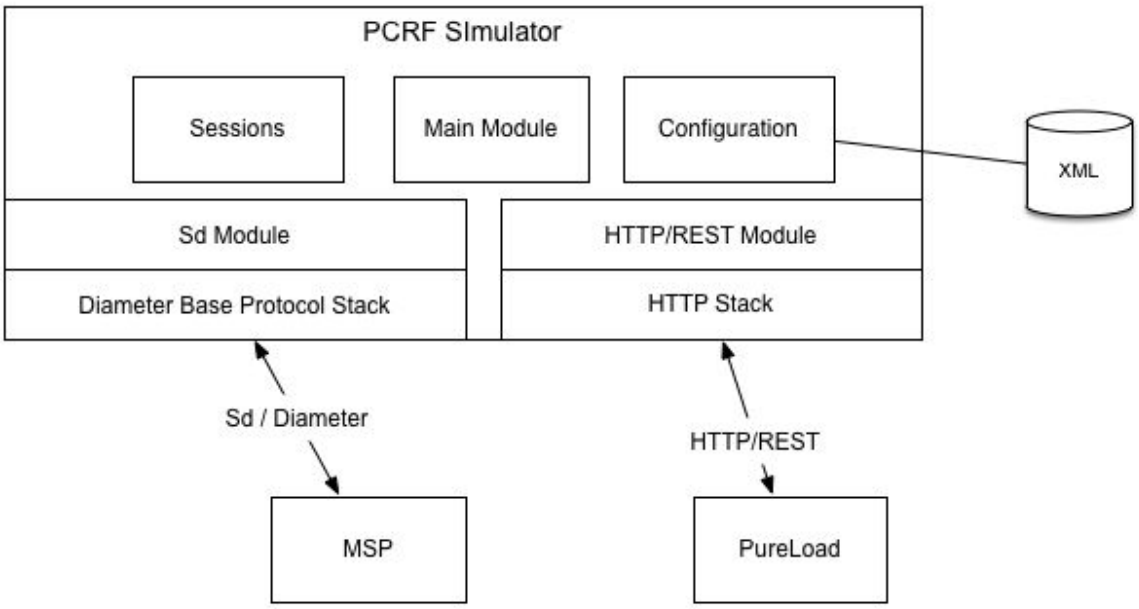


Any errors during the session termination, such as invalid CCA-U received for unknown session, is logged and indicated in statistics log.

PCRF Simulator

Overview

The PCRF simulator is designed as a Java based server application. The following is an overview of the implementation



Established sessions are stored in memory. Information stored are session id as well as information provided when starting the session.

The Diameter part of the PCRF simulator supports Diameter Sd and Diameter Gx. See previous section regarding Diameter support.

To support initiating, updating and terminating session from PureLoad a simple HTTP based RESTful web service API is used.

Simple statistics are written to file

Installation

To use the PCRF simulator official Java from Oracle is required. The latest Java 7 version is recommended.

The PCRF simulator is provided as a tar file that is extracted in an appropriate directory. The content of the tar file:

- README.txt
  General information.
- bin/
  Scripts to start the simulator
- config/
  Configuration files
- lib/
  Used JAR files
- log/
  Directory to store log files.
- doc/
  Directory with documentation (this file)

The script to start the HSS Simulator bin/pcrf-simulator.sh must be configured to define where Java is installed. Change the following line:

```
JAVA_HOME=/opt/java/jdk1.8
```

to where Java, version 8 is installed.

In addition the size of the Java Heap might need to be increased. If so change the line:

```
JAVA_OPTS="-Xms1g -Xmx2g -XX:+UseG1GC $JAVA_OPTS"
```

as needed. Default is 2GB max heap.

Configuration

Main properties of the server is controlled by the XML file config/hss-server.xml. By default this file looks like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<pcrf-simulator>
    <!-- Diameter server properties -->
    <diameter>
        <!-- Host name or IP address -->
        <host>localhost</host>
        <!-- Origin host name -->
        <origin-host></origin-host>
        <!-- Port -->
        <port>3868</port>
        <!-- Server realm -->
        <realm>pureload.com</realm>
        <!-- Default vendor id -->
        <vendor-id>10415</vendor-id>
        <product-name>PureLoad HSS Simulator</product-name>
        <firmware>1</firmware>
        <!-- Request timeout (milliseconds)-->
        <request-timeout>4000</request-timeout>
        <custom-avps>
          <avp name="TDF-Flow-Identifier" code="9898" vendor-id="10415" type="Unsigned32"/>
        </custom-avps>
    </diameter>
    <!-- Statistics configuration -->
    <statistics>
        <!-- Interval in seconds -->
        <interval>30</interval>
        <!-- Report Sd statistics -->
        <type>Sd</type>
        <!-- Report Gx statistics -->
        <type>Gx</type>
    </statistics>
    <!-- HTTP server host/port -->
    <http>
        <!-- Host name (or IP address) and port. -->
        <host>localhost</host>
        <port>8080</port>
    </http>
    <!-- Session update response APVs (CCA-U AVPs) per subscription id's →
    <session-update>
       <update-avp subscription-id="4670924*">
            <![CDATA[
             <Session-Data>...</Session-Data>
            ]]>
       </update-avp>
    </session-update>
    <!-- Gx session update response APVs (Gx CCA-U AVPs) per subscription id's -->
    <gx-session-update>
       <update-avp subscription-id="4670924*">
            <![CDATA[
             <Session-Data>...</Session-Data>
            ]]>
       </update-avp>
    </gx-session-update>
</pcrf-simulator>
```

Diameter

Configuration of various properties controlling the Diameter server. Most elements are self explanatory, but a few notes:

- diameter/host element
  Hostname or IP address for local peer
- diameter/origin-host element
  Used to set Origin-Host AVP to different from what's' specified by host element
- diameter/port element
  Port for the local peer(s).
- diameter/request-timeout
  The internal timeout used by PCRF simulator for Diameter requests

Custom AVPs

Basic, non-grouped custom AVPs can be added using the custom-avps element.
See the example in the provided default configuration or contact PureLoad support for details.

Statistics

Basic statistics is produced and stored in the log/stats.log file. The interval when to write statistics are specified using the statistics/interval element. If less than or equals to zero statistics are disabled.
It is also possible to control if Sd and/or Gx statistics to be produced.

HTTP

Configuration of various properties controlling the HTTP (Web Service) server.

- http/host element
  Hostname or IP address of HTTP server.
- http/port element
  Port for the HTTP server.

Session Update

Configuration of what Sd CCA-U AVPs to be set per subscription ID for TDF initiated session updates (CCR-U) is defined using the session-update element.

Configuration of what Gx CCA-U AVPs to be set per subscription ID for PCC Rule Pull updates (CCR-U) is defined using the gx-session-update element.

The update-avp specifies the APVs to be added. One set of AVPs can be specified per subscription ID, which can be specified using wildcards (to mach a range of subscription IDs).

For example:

```
<session-update>
    <update-avp subscription-id="343730">
        <![CDATA[
        <Session-Data>
          <ADC-Rule-Install>
            <ADC-Rule-Definition>
              <ADC-Rule-Name>dummy</ADC-Rule-Name>
              <TDF-Application-Identifier>entds</TDF-Application-Identifier>
            </ADC-Rule-Definition>
          </ADC-Rule-Install>
        </Session-Data>
        ]]>
    </update-avp>
    <update-avp subscription-id="4670924*">
        <![CDATA[
        <Session-Data>
          <ADC-Rule-Remove>
            <ADC-Rule-Definition>
               <ADC-Rule-Name>dummy</ADC-Rule-Name>
               <TDF-Application-Identifier>entds</TDF-Application-Identifier>
             </ADC-Rule-Definition>
          </ADC-Rule-Remove>
        </Session-Data>
        ]]>
    </update-avp>
</session-update>
```

The first update-avp element will be used for subscription ID 343730. The second update-avp element will be used for any subscription ID starting with 4670924.

The XML syntax to specify AVPs is the same as used in the HTTP (REST) interface.

Logging

Logging is controlled by the file config/log4j.properties. To change logging from the default level (INFO) to detailed logging (DEBUG) for stdout, change the following lines:

```
log4j.rootLogger = INFO, CONSOLE, FILE
…
log4j.appender.CONSOLE.Threshold=INFO
```

Change INFO to DEBUG and a lot of details will be logged to stdout.

Web Services (REST) Interface

The HTTP based API to HSS Simulator is designed as a RESTful web service. Session can be created, updated and terminated:

- Session are created using HTTP POST
- Session updates using HTTP PUT
- Sessions are terminated using HTTP DELETE

Request URLs and Response Status

The URL path used is:

- /sd-session/*subscription-id* or /session/*subscription-id*
  Control TDF/Sd sessions.
- /gx-session/*subscription-id*
  Control Gx sessions

where *subscription-id* (MSISDN) identifies the session.

To create or update a session the entity body of the HTTP request must include request bodys using XML session data as described below.

The following response status codes are used:

- 200 Ok
  The Request was successful
- 400 Bad Request
  The request could not be completed
- 404 Not Found
  Session not found
- 500 Internal Server Error
  Undefined server errors

XML Session Data

Sd based sessions

The XML Session Data that must be specified when creating a new Sd based session in the format, shown by the following example:

```
<?xml version="1.0" encoding="UTF-8"?>
<Session-Data>
    <Subscription-Id>46709242010</Subscription-Id>
    <Destination-Host>msp.ericsson.com</Destination-Host>
    <Framed-IP-Address>192.168.1.100</Framed-IP-Address>
    <Event-Trigger>APPLICATION_START</Event-Trigger>
    <Event-Trigger>APPLICATION_STOP</Event-Trigger>
    <Sd-MSP-Specific-Attr-N
code="131506">www.sl.se/byod/sl.json</Sd-MSP-Specific-Attr-N>
    <ADC-Rule-Install>
       <ADC-Rule-Definition>
          <ADC-Rule-Name>e-byod</ADC-Rule-Name>
          <TDF-Application-Identifier>entds</TDF-Application-Identifier>
       </ADC-Rule-Definition>
   </ADC-Rule-Install>
</Session-Data>
```

This is a simple example with the minimum set of elements. All elements are mapped to Diameter Sd AVPs with the corresponding names. Not all possible AVPs are listed below, but all Sd AVPs are supported.

Session-Data
This is the root element that must be used. The most important child elements are:

Subscription-Id
The subscription id (MSISDN) to be used.

Destination-Host
Destination host of remote peer. Optional and only required if PCRF simulator have more than one remote peer.

Framed-IP-Address or Framed-IPv6-Prefix
The UE IP address.

Event-Trigger
Event to be enabled. Typically both APPLICATION_START and APPLICATION_STOP is used.

Wait-For-CCR

This is used to control if we should wait for a CCR when to establish a new session. Default is true. **Note**: this *must* be set to false for TDF session establishment!

Sd-MSP-Specific-Attr-N
This element requires that a code attribute is specified to set the actual AVP code to be generated. For example to generate a Sd-MSP-Specific-Attr-6 with AVP code 131506:

```
<Sd-MSP-Specific-Attr-N code="131506">
www.sl.se/byod/sl.json
</Sd-MSP-Specific-Attr-N>
```

ADC-Rule-Install
Optional element to specify a rule to be installed. Child elements are used to define the AVPs to be included in the nester and grouped AVPs to be generated.

The XML structure is the same as corresponding ADC-Rule-Install AVP. This means (for example) that the following is a valid example:

```
<ADC-Rule-Install>
    <ADC-Rule-Definition>
        <ADC-Rule-Name>rule-fat-00</ADC-Rule-Name>
        <TDF-Application-Identifier>app1</TDF-Application-Identifier>
        <Flow-Status>0</Flow-Status>
        <QoS-Information>
            <QoS-Class-Identifier>0</QoS-Class-Identifier>
            <Max-Requested-Bandwidth-DL>22</Max-Requested-Bandwidth-DL>
            <Guaranteed-Bitrate-DL>44</Guaranteed-Bitrate-DL>
            <Bearer-Identifier>1</Bearer-Identifier>
            <Allocation-Retention-Priority>
                <Priority-Level>1</Priority-Level>
                <Pre-emption-Capability>0</Pre-emption-Capability>
                <Pre-emption-Vulnerability>0</Pre-emption-Vulnerability>
            </Allocation-Retention-Priority>
            <APN-Aggregate-Max-Bitrate-UL>-1</APN-Aggregate-Max-Bitrate-UL>
            <APN-Aggregate-Max-Bitrate-DL>-1</APN-Aggregate-Max-Bitrate-DL>
        </QoS-Information>
        <Monitoring-Key>test</Monitoring-Key>
        <Redirect-Information>
            <Redirect-Support>0</Redirect-Support>
        </Redirect-Information>
        <Mute-Notification>0</Mute-Notification>
    </ADC-Rule-Definition>
    <Rule-Activation-Time>12:00:00</Rule-Activation-Time>
    <Rule-Deactivation-Time>12:00:00</Rule-Deactivation-Time>
</ADC-Rule-Install>
```

Gx Based Sessions

The XML Session Data that should be specified when updating Gx based session in the format, shown by the following example:

```
<?xml version="1.0" encoding="UTF-8"?>
<Session-Data>
   <Subscription-Id>46709242010</Subscription-Id>
   <Charging-Rule-Remove>
      <Charging-Rule-Base-Name>rule1</Charging-Rule-Base-Name>
   </Charging-Rule-Remove>
   <Charging-Rule-Install>
      <Charging-Rule-Base-Name>rule2</Charging-Rule-Base-Name>
   </Charging-Rule-Install>
</Session-Data>
```

This is a simple example with the minimum set of elements. All elements are mapped to Diameter Gx AVPs with the corresponding names.

Note that not all Gx AVPs are supported.

Another example to be used when terminating a Gx based session:

```
<?xml version="1.0" encoding="UTF-8"?>
<Session-Data>
   <Subscription-Id>46709242010</Subscription-Id>
   <Session-Release-Cause>IP_CAN_SESSION_TERMINATION</Session-Release-Cause>
</Session-Data>
```