PureLoad®
extreme performance testing

Pure HSS Simulator

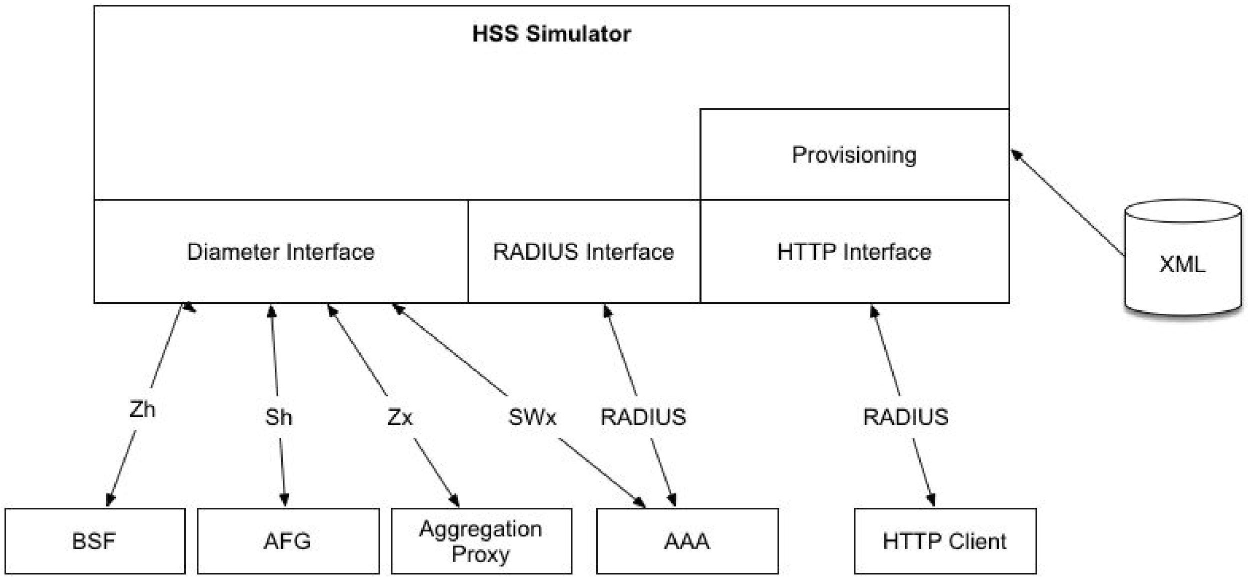
Table of Content

Introduction

This document describes the PureLoad HSS Simulator. The Simulator is designed to be used for performance and stability tests, using a subset of the Sh, Zh and Zx Diameter interfaces.

The HSS Simulator is implemented in Java and provides a Diameter interface and a simple HTTP based web services interface. All provisioned data (user identities) are handled in-memory and no persistent storage is provided.

The following picture shows an overview of the implementation and provided interfaces:



Configuration settings are controlled by an external XML file. Provisioning is also provided by a set of XML-files that are read during startup. In addition, changes may be made using the REST based web services interface.

Installation

Prerequisites

To use the HSS Simulator, official Java 8 from Oracle is required. The latest Java 8 version is recommended.
SCTP is supported on Linux only. See the section about SCTP below.

Distribution

The HSS 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)

Diameter over SCTP

SCTP is supported on Linux only and is based on the Linux kernel built-is support. In addition the Linux user space library for SCTP (libsctp) is required. How to install libsctp depends on the Linux distribution used:

- Ubuntu
  sudo apt-get install lksctp-tools
- SuSE
  sudo zypper install lksctp-tools

Start Script Modification

The script to start the HSS Simulator bin/hss-simulator.sh must be configured to define where Java 8 is installed.

Change the following line:

```
JAVA_HOME=/home/pureload/jre1.8
```

to where Java 8 is installed.

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

```
JAVA_OPTS="-Xms64m -Xmx1024m $JAVA_OPTS"
```

Configuration

Server

The server is configured by the XML file config/hss-server.xml. The file consist of 5 child configuration elements:

```
<?xml version="1.0" encoding="UTF-8"?>
<hss-simulator>

   <!-- Diameter server properties -->
   <diameter>
    ...
   </diameter>

   <!-- Radius server properties -->
   <radius>
    ...
   </radius>

   <!-- Provisioning -->
   <provisioning>
    ...
   </provisioning>

   <!-- Statistics configuration -->
   <statistics>
    ...
   </statistics>

   <!-- HTTP server host/port -->
   <http>
    ...
   </http>

</hss-simulator>
```

- Diameter element
  Defines the diameter interface
- Radius element
  Defines the radius interface
- Provisioning element
  Provisioning of User Identities
- Statistics element
  Define if statistics should be logged
- Http element
  Define the HTTP interface

Diameter

```
    <!-- Diameter server properties -->
    <diameter>
        <!-- Hostname or IP address. Multiple comma separated hosts can be specified -->
        <host>192.168.1.100</host>
        <!-- Origin host names(s). Multiple comma separated names can be specified -->
        <!-- (Used to set Origin-Host AVP to override value specified by host element) -->
        <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>
        <!-- Overload control (RFC 7683). Disabled by default -->
        <overload>
            <enabled>true</enabled>
            <threshold-tps>10</threshold-tps>
            <validity-duration>30</validity-duration>
            <reduction-percentage>10</reduction-percentage>
        </overload>
        <!-- Request timeout for diameter requests sent (milliseconds)-->
        <request-timeout>3000</request-timeout>
        <!-- Simulated delay (milliseconds) -->
        <delay>0</delay>
        <!-- Protocol (TCP or SCTP) -->
        <protocol>TCP</protocol>
        <sctp>
            ...
        </sctp>
    </diameter>
```

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

- host element
  Host name(s) or IP address(es) for local peer(s). Multiple IP addresses or hostnames can be specified, separated by comma. For SCTP this is the local primary addresses.
- port element
  Port for the local peer.
- overload
  Diameter overload configuration
- request-timeout
  Used in cases where the HSS simulator send outgoing Diameter messages
- delay
  Used to add a simulated delay before Diameter answer messages is sent
- protocol (TCP or SCTP), controls which transport protocol to use
  If set to SCTP, the SCTP subsection can be used to configure SCTP specific settings.

Diameter SCTP

If Diameter protocol is set to SCTP, the sctp child element is used to configure details of the SCTP implementation.

```
<!-- Diameter server properties -->
<diameter>
    ...
    <!-- Protocol (TCP or SCTP) -->
    <protocol>SCTP</protocol>
    <sctp>
        <!-- Associated hosts. 1'st host must the primary address, specified in
          <host> element above -->
        <associated-hosts>192.168.1.100, 192.168.0.101, 192.168.0.102</associated-hosts>
        <!-- Time to live, 0 for no timeout  -->
        <ttl>0</ttl>
        <!-- Max number of inbound streams -->
        <max-in>10</max-in>
        <!-- Max number of outbound streams -->
        <max-out>4</max-out>
    </sctp>
</diameter>
```

The associated-hosts element is used to specify the host name(s) or IP address(es) to bind to (multi-homing). Hosts are be specified, separated by comma, where the first host must be one specified in the diameter host element (the primary host).

Radius

The Radius interface is configured using the radius child element:

```
<!-- Radius server properties -->
<radius>
    <!-- Hostname or IP address. Multiple comma separated hosts can be specified -->
    <host>localhost</host>
    <!-- Ports (-1 to disable) -->
    <auth-port>1812</auth-port>
    <acct-port>1813</acct-port>
    <!-- RADIUS secret -->
    <secret>pureload</secret>
    <!-- Add Message-Authenticator AVP to response packets (true/false) -->
    <add-message-authenticator>true</add-message-authenticator>
</radius>
```

Most are self explanatory, but a few notes:

- host element
  Host name(s) or IP address(es) for local peer(s). Multiple IP addresses or hostnames can be specified, separated by comma.
- auth-port element
  Port used for RADIUS authentication messages.
- acct-port element
  Port used for RADIUS accounting messages.

Provisioning

Provisioning of User Identities is done by specifying path(s) to additional XML file(s). Multiple path elements may be used. If a path is relative it is supposed to be located in the same directory as where the main server configuration is located.
In addition provisioning of InitialFilterCriteria tied to public user identities, is done by specifiying path to an additional file.

```
<!-- Provisioning -->
<provisioning>
    <!-- Basic/default provisioning -->
    <path>/hss-users.xml</path>
    <!-- Additional provisioning -->
    <path>/function-test-users.xml</path>
    <!-- Sh IFC provisioning -->
    <sh-ifc-path>/sh-ifc.xml</sh-ifc-path>
</provisioning>
```

For details on the XML format see the “Provisioning of User Identities” section.

Statistics

Basic statistics is produced and stored in the log/stats.log file. The interval when to write statistics are specified in seconds using the statistics/interval element.

```
<!-- Statistics configuration -->
<statistics>
    <!-- Interval in seconds -->
    <interval>0</interval>
</statistics>
```

HTTP

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

```
<!-- HTTP server host/port -->
<http>
    <!-- Hostname or IP address. Multiple comma separated hosts can be specified -->
    <host>localhost</host>
    <!-- Port -->
    <port>8080</port>
</http>
```

- http/host element
  Host name(s) or IP address(es) for HTTP server. Multiple IP addresses or hostnames can be specified, separated by comma.
- http/port element
  Port for the HTTP server.

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.

Starting the Simulator

The HSS Simulator is started using the start script in the bin directory which takes the configuration path and library path as optional arguments:

```
%> ./hss-simulator.sh [<config path> [<lib path>]]
```

The configuration path is by default the config/ directory and library path the lib/ directory where the simulator is installed.

Provisioning

Provisioning of User Identities

Data tied to user identities is provisioned using one or more XML files containing private / public identity user data.

The main structure of the data should be as follows:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<users>
  <private-identity>
     <!-- Private identity elements -->
  </private-identity>
  <public-identity>
     <!-- Private identity elements -->
  </public-identity>
</users>
```

Multiple private-identity and public-identity elements may be specified. The defined identities are used by the Diameter interfaces as follows:

- Private Identities
  Used by Zh and SWx
- Public Identities
  Used by Sh and Zx

Both private-identity and public-identity elements have a mandatory attribute type that can be:

- default
  Default settings. All identities will use these settings, if not overridden.
- range
  Used to specify settings for a range of user identities. Ranges are specified using regular expressions.
- user
  Used to specify settings for an individual single user identity.

Incoming requests are first matched against single user definitions and if no match, the range definitions will be tried in the order they are defined in the XML file.

Private Identities

A typical definition of a Private Identity (where type is default) could look like:

```
<private-identity type="default">
  <identity>default</identity>
  <!-- Authentication settings -->
  <!-- secret key as string, required length 16 -->
  <auth-key>1234567890123456</auth-key>
  <!-- Amf id as hex bytes, required length 2 bytes, e.g. 0A5F -->
  <auth-amf>0000</auth-amf>
  <!-- Operator secret as hex bytes, required length 16 bytes -->
  <auth-op>00000000000000000000000000000000</auth-op>
  <!-- Sequence number as hex bytes, required length 6 bytes -->
  <auth-sqn>000000000000</auth-sqn>
  <!-- GUSS settings -->
  <guss-lifeTime>3600</guss-lifeTime>
  <guss-uiccType>GBA</guss-uiccType>
</private-identity>
```

This can then be overridden to define a single user identity:

```
<private-identity type="user">
  <identity>alice@ims.test</identity>
  <auth-key>1234567890123456</auth-key>
  <guss-ussList>
     <uss id="0" type="0">
       <uids>
         <uid>sip:alice@ims.test</uid>
         <uid>sip:alice2@ims.test</uid>
       </uids>
     </uss>
  </guss-ussList>
</private-identity>
```

Here we define a private identity “alice@ims.test”. All settings in the “default” specification will be used for this user, with the exception of “auth-key” and the “guss-ussList”.

An example where we specify a range of private identities:

```
<private-identity type="range">
  <identity>user(\d\d\d)@ims.test</identity>
  <guss-ussList>
     <uss id="0" type="0">
       <uids>
         <uid>sip:user%s@ims.test</uid>
         <uid>sip:user%s@alt.ims.test</uid>
       </uids>
     </uss>
     <extension><![CDATA[<mbmsserviceidlist><mbmsserviceid>
       <keydomainid>domain%$1s</keydomainid><mskid>mskid%$1s</mskid>
       </mbmsserviceid></mbmsserviceidlist>]]>
     </extension>
  </guss-ussList>
</private-identity>
```

I.e Private Identities “user000@ims.test” to “user999@ims.test” is defined.

SWx Support

To support SWx additional elements can be specified:

```
<private-identity type="default">
  <identity>default</identity>
  …
  <allowed-networks>pureload.com</allowed-networks>
  <allowed-access-types>WLAN,VIRTUAL</allowed-access-types>
  <Non-3GPP-User-Data>
     <Subscription-Id>
       <Subscription-Id-Type>END_USER_E164</Subscription-Id-Type>
       <Subscription-Id-Data>016646997167</Subscription-Id-Data>
     </Subscription-Id>
     <Non-3GPP-IP-Access>NON_3GPP_SUBSCRIPTION_ALLOWED</Non-3GPP-IP-Access>
     <Non-3GPP-IP-Access-APN>Non_3GPP_APNS_DISABLE</Non-3GPP-IP-Access-APN>
 </Non-3GPP-User-Data>
</private-identity>
```

The support SWx related elements are:

- allowed-networks
  Comma separated list of networks allowed
- allowed-access-types
  Comma separated list of allowed access types (RAT-Type)
- Non-3GPP-User-Data
  Non-3GPP-User-Data AVP, specified in XML format

Public Identities

A typical definition of a Public Identity (where type is default) could look like:

```
<public-identity type="default">
  <identity>default</identity>
  <!-- Digest Password (used by Zx) -->
  <digest-password>123456</digest-password>
  <!-- SSO setting (used by Zx)-->
  <!-- The following values can be used: enabled, unavailable or blocked -->
  <sso>unavailable</sso>
  <!-- Optional flag indicating if Framed-IP-Address should be matched (Zx) -->
  <!-- Set to true or false. Default is false. -->
  <match-framed-ip>false</match-framed-ip>
  <!-- Optional Repository-Data -->
  <repository-data><![CDATA[<?xml version="1.0" encoding="UTF-8"?>
    <Sh-Data xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             xmlns="http://schemas.ericsson.com/HSS/app_service_data">
     <RepositoryData><ServiceIndication>NAFMEDIA</ServiceIndication>
     <SequenceNumber>101</SequenceNumber><ServiceData><mbmsserviceidlist>
     <mbmsserviceid><keydomainid>domain1</keydomainid><mskid>mskid1</mskid></mbmsserviceid>
     <mbmsserviceid><keydomainid>domain2</keydomainid><mskid>mskid2</mskid></mbmsserviceid>
     </mbmsserviceidlist></ServiceData></RepositoryData>
    </Sh-Data>]]>
  </repository-data>
</public-identity>
```

This can then be overridden to define a single user identity:

```
<public-identity type="user">
  <identity>sip:alice@ims.test</identity>
  <msisdn>+4563457900</msisdn>
  <ipv4-address>192.168.1.100</ipv4-address>
  <sso>enabled</sso>
</public-identity>
```

Similar to Private Identities we can specify a range of Public Identities:

```
<public-identity type="range">
  <identity>sip:user\d\d\d@ims.test</identity>
</public-identity>
```

I.e Public Identities “sip:user000@ims.test” to “sip:user999@ims.test” are defined.

More about Range definitions

Ranges of user identities are specified using regular expressions. Any regular expression can be used and as long as the incoming request matches the regular expression, the identity will be accepted and dynamically constructed in-memory for the duration of the request.

Each private identity requires a relationship to one or more public identities. For ranges of private identities, this has been solved by allowing references to capturing groups in the regular expression. The references are then used when specifying the related public identities using the marker ‘%s’. The first ‘%s’ refers to the first capturing group, the second to the second group and so on. It is also possible to explicitly say which group to use with the syntax ‘%$1s’ for the first group, ‘%$2s’ for the second group (and so on).

References to capturing groups can be used in the following fields:

- uid (in private-identity)
- GUSS-extension CDATA (in private-identity)
- Repository-Data CDATA (in public-identity)

The regular expression user(\d\d\d)@ims.test contains one capturing group with three digits. A public identity can then be defined as sip:user%s@ims.test where the ‘%s’ will be replaced by the matched three digits. E.g. an incoming request for private identity user017@ims.test would resolve the public identity reference to sip:user017@ims.test.

Provisioning of Sh InitialFilterCriteria

XML data to be returned by Sh UDR requests (Sh Pull) with Data-Reference AVP set to InitialFilterCriteria. The information is organized per group of public user identities specified using regular expression.

The main structure of the data should be as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Sh IFCs (InitialFilterCriteria's) configuration -->
<sh-ifcs>
  <!-- IFCs for public identities “sip:user000@ims.test” to “sip:user999@ims.test” -->
  <ifc identity="sip:user\d\d\d@ims.test">
    <![CDATA[
    ]]>
   </ifc>
</sh-ifcs>
```

Multiple ifc element may be specified and will be matched against actual public identity. If a match the CDATA (XML data) information is returned in the UDA answer message.
A more complete example with actual IFCs XML elements:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Sh IFCs (InitialFilterCriteria's) configuration -->
<sh-ifcs>
   <!-- IFCs for public identities “sip:user000@ims.test” to “sip:user999@ims.test” -->
   <ifc identity="sip:user\d\d\d@ims.test">
       <![CDATA[
       <IFCs>
           <InitialFilterCriteria>
               <Priority>0</Priority>
               <TriggerPoint>
                   <ConditionTypeCNF>0</ConditionTypeCNF>
                   <SPT>
                       <ConditionNegated>1</ConditionNegated>
                       <Group>0</Group>
                       <SIPHeader>
                           <Header>XDMS</Header>
                           <Content>VPN</Content>
                       </SIPHeader>
                   </SPT>
               </TriggerPoint>
               <ApplicationServer>
                   <ServerName>sip:AS1@homedomain.com</ServerName>
               </ApplicationServer>
           </InitialFilterCriteria>
```

```
        </IFCs>
        ]]>
    </ifc>
</sh-ifcs>
```

Diameter Support

The Diameter Base Protocol Stack support all parts of IETF RFC 3588 that is required to support the the Diameter Applications described below.

The Diameter Applications/Interfaces supported are Sh, Zh, Zx and SWx as described in Reference [1,2,3,4]. All command codes and AVPs described are supported.

Note that even if all Applications are supported on Diameter protocol level, not all are supported by the HSS Simulator.

Sh

The Sh module process Sh User-Data-Request UDR) commands and returns Sh User-Data-Answer (UDA) commands.

Three types of Data-Reference AVP values are supported:

- Repository-Data
  Public identity is looked up and User-Data AVP containing Repository-Data XML is returned.
- IMSPublicIdentity
  Public identity is looked up and User-Data AVP is returned including the public identifier defined (in XML format).
- IPAddressSecureBindingInformation
  Public identity is looked up and defined IP information is returned using the Framed-IP-Address, Framed-IPv6-Prefix or Framed-Interface-Id AVPs
- InitialFilterCriteria
  InitialFilterCriteria (IFCs) XML are looked up per received Server-Name AVP (see previous “Provisioning of Sh InitialFilterCriteria” section). The IFCs are returned using User-Data AVP.

Zh

The Zh module process Zh Multimedia-Authentication-Request (MAR) commands and returns Zh Multimedia-Authentication-Answer (MAA) commands.

Zh is used for providing services in the 3GPP Generic Bootstrapping Architecture (GBA) and performs the retrieval of an authentication vector and GBA User Security Settings (GUSS).

A private identity is looked up based on the user name in the incoming MAR. Depending on requested authentication scheme, an authentication vector is then generated (AKA or SIP Digest).

The GUSS contains references to public identities. An example GUSS XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<guss id="user017@ims.test">
 <bsfInfo>
   <uiccType>GBA_U</uiccType>
   <lifeTime>3600</lifeTime>
 </bsfInfo>
 <ussList>
   <uss id="2" type="2" nafGroup="0">
     <uids>
       <uid>sip:user017@ims.test</uid>
     </uids>
     <flags></flags>
  </uss>
</ussList>
</guss>
```

Zx

The Zx module process Zx Multimedia-Authentication-Request (MAR) commands and returns Zx Multimedia-Authentication-Answer (MAA) commands.

A public identity is looked up based on the incoming MAR. If the flag ‘match-framed-ip’ is set for the public identity, the framed IP address is matched against the MAR.

Depending on requested authentication scheme (SSO or Digest), an authenticate message is then returned in the answer.

SWx

The SWx module process incoming MAR and SAR commands for a AAA server as described in Reference [4]. In addition HSS initiated RTR and PPR commands can be triggered using the REST API.

To identify a subscriber the User-Name AVP received in Diameter messages are used and consist of a user IMSI. Corresponding private user identities are assumed to be defined as: *imsi@destination-realm*.

The following describes an overview of the procedures supported (not all details are described).

Authentication Procedure

When a MAR command is received the following is done:

1. check that the user exists
2. check that the subscriber has Non-3GPP-User-Data defined
3. check Visited-Network-Identifier (if present)
4. check the access type
5. check if there is an existing 3GPP AAA Server already assigned the subscriber
6. check the request type and possibly authenticate

If any step fails an error is returned.
AAA Server identity is defined using the Origin-Host AVP.

Registration/DeRegistration Notification

When a SAR command is received the following is done:

1. check that the user exists
2. check if there is an existing 3GPP AAA Server already
3. check Server-Assignment-Type AVP
    a. REGISTRATION - check/set subscriber status
    b. DEREGISTRATION - clear assigned 3GPP AAA Server
    c. AAA_USER_DATA_REQUEST - return Non-3GPP-User-Data
    d. PGW_UPDATE - update APN id, Content id, etc.
    e. RESTORATION - update the registration data

If any step fails an error is returned.
AAA Server identity is defined using the Origin-Host AVP.

Network Initiated DeRegistration by HSS

Sending RTR messages is triggered using the REST API or done as a result from processing a MAR command (when a new AAA server is assigned).

The provided REST API call will include information required to generate all AVPs in generated RTA (Destination-Host and Deregistration-Reason).

If Deregistration-Reason is PERMANENT_TERMINATION current AAA server for the subscriber is cleared and user status is set to NOT_REGISTERED.

HSS Initiated Update of User Profile Procedure

Sending PPR messages is triggered using the REST API.

The provided API will include all information required to generate all required PPR AVPs (Non-3GPP-User-Data and PPR-Flags).

Diameter overload control

Diameter overload control as specified in Reference [5] is supported and is enabled is specified in the server configuration.

If overload control is enabled, the HSS simulator will check the diameter TPS threshold and if overload is detected OC-Supported-Features and OC-OLR AVPs are added to Diameter answer messages.

To have a certain inertia when TPS is calculated actual TPS is calculated using an average of transactions for the last 30 seconds.

RADIUS Support

The RADIUS interface supports RADIUS authentication and accounting as as described in Reference [6]. Limitations applies as follows.

Authentication

Only authentication types PAP and CHAP are supported.
When AccessRequests packets are received the RADIUS secret is verified. If not verified the request is silently ignored. If verified, received authentication key and password are authenticated. If authentication is successful an AccessAccept response packet is returned, else AccessReject is returned.

Accounting

Received AccountingRequests is processed and simple AccountingResponse message is returned.

In addition the MessageAuthenticator attribute is supported. If this AVP is included in received packet it is used to verify the integrity of the AccessRequest.
If configured to be included in responses it is also added to response packets.

Web Services (REST) Interface

The HTTP based API to HSS Simulator is designed as a RESTful web service.
All identities can be manipulated using HTTP GET, POST, PUT and DELETE methods.

Use GET to read identities, POST to create new ones, PUT to edit and DELETE to delete existing ones.

Two different URL paths are used to specify the identity to be manipulated:

- /private-identity
- /public-identity

The path must be followed by the identity to be manipulated. To create or change an identity the entity body of the HTTP request must include the XML data used to manipulate the identity. The format is the same as used for provisioning of identities.

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
  Identity not found
- 500 Internal Server Error
  Undefined server errors.

Read Identity

Example request reading a private identity:

```
GET /private-identity/user@ims.test
Accept: application/xml
```

Returned response body consists of XML representing identity.

Create New Identity

Example request creating a private identity:

```
POST /private-identity/user@ims.test
Content-Type: application/xml
<?xml version="1.0"?>
<private-identity type="user">
   <identity>user@ims.test</identity>
</private-identity>
```

Update Identity

Example request updating a public identity:

```
PUT /public-identity/user@ims.test
Content-Type: application/xml
<?xml version="1.0"?>
<public-identity type="user">
   <identity>sip:alice@ims.test</identity>
   <ipv4-address>192.168.1.100</ipv4-address>
   <sso>enabled</sso>
</public-identity>
```

The existing public identity will be completely replaced with the new one.

Delete Identity

Example deleting an private identity:

```
DELETE /private-identity/user@ims.test
```

SWx Procedures

By using a query parameter with a HTTP PUT method SWx procedures can be triggered. Two query parameters are supported:

- /private-identity?swx-rtr
  To trigger sending SWx RTR Diameter message to an AAA server
- /private-identity?swx-ppr
  To trigger sending SWx PPR Diameter message to an AAA server

Response body is set to the received Diameter message AVP's, in XML format.
AVPs of the Diameter messages to be sent is specified in XML format as follows:

SWx RTR

Example request that will trigger the simulator to send a RTR message:

```
PUT /private-identity/001010999999001@pureload.com?swx-rtr
Content-Type: application/xml
<?xml version="1.0" encoding="UTF-8"?>
<RTR>
    <Deregistration-Reason>
       <Reason-Code>PERMANENT_TERMINATION</Reason-Code>
     </Deregistration-Reason>
    <Destination-Host>192.168.0.100</Destination-Host>
</RTR>
```

The Deregistration-Reason must be specified. If Destination-Host is unspecified, the RTR message is sent to the currently associated AAA server for the user.

SWx PPR

Example request that will trigger the simulator to send a PPR message:

```
PUT /private-identity/001010999999001@pureload.com?swx-ppr
Content-Type: application/xml
<?xml version="1.0" encoding="UTF-8"?>
<PPR>
    <PPR-Flags>0</PPR-Flags>
    <Destination-Host>192.168.0.100</Destination-Host>
</PPR>
```

If Destination-Host is unspecified, the PPR message is sent to the currently associated AAA server for the user.

PureLoad Diameter Tasks

PureLoad Diameter Tasks, supporting Zh, Sh and Zx is provided as a separate task extension package. This package can be used for testing and verification of the HSS Simulator, and when simulation of Diameter client is needed.

References

[1] Zh Interface Description in HSS - 7/155 19-CRA 119 0246/11 Uen A

[2] Sh Interface Description in HSS - 2/15519-CRA 119 0246/11 Uen A

[3] Zx Interface Description in HSS - 3/155 19-CRA 119 0248/11 Uen A

[4] Diameter SWx application as described in
3GPP TS 29.273 "3GPP EPS AAA interfaces" release 13.

[5] Diameter Overload Indication Conveyance, RFC 7683

[6] RADIUS as described in RFC 2865, RFC 2866 and RFC 2869