

# Seagull - Radius protocol

## Table of contents

1 Radius protocol details.....	2
2 Getting started with Radius.....	2
2.1 First try.....	2
2.2 First try explained.....	3
3 Configuration files.....	7
3.1 Generic configuration.....	7
3.2 Radius dictionary.....	8
3.2.1 Types.....	8
3.2.2 Header.....	8
3.2.3 Body.....	8
3.2.4 Dictionary.....	8
3.2.5 Message.....	9
3.3 Actions in scenario commands for Radius.....	9
4 Radius authentication.....	10
4.1 Seagull implementation.....	10
4.2 Authentication dictionary.....	10
4.3 Authentication scenario.....	11

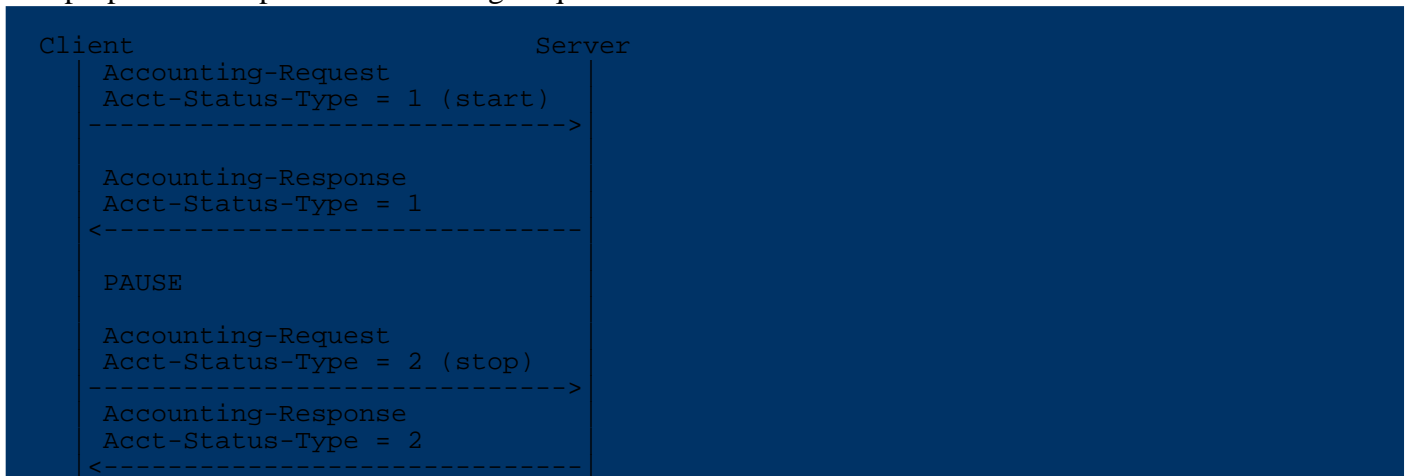
## 1. Radius protocol details

The Radius implementation in Seagull conforms to the RADIUS authentication RFC: [RFC 2865](http://tools.ietf.org/html/rfc2865) (<http://tools.ietf.org/html/rfc2865>) and the RADIUS accounting RFC : [RFC 2866](http://tools.ietf.org/html/rfc2866) (<http://tools.ietf.org/html/rfc2866>)

## 2. Getting started with Radius

### 2.1. First try

The proposed example is a "Accounting Request" scenario:



#### Note:

This scenario is included when you install Seagull. It is located in /usr/local/share/seagull/radius/ directory.

Open two terminal sessions. Terminal 2 will be the server and Terminal 1 the client. Examples are located in the "run" directory. So the first thing you need to do is to go in this directory (in both terminal windows):

```
cd run
```

In Terminal 2 window type:

```
./start_server.ksh
```

In Terminal 1 window type:

```
./start_client.ksh
```

On Terminal 2 (server side), you will see:

Start/Current Time	2007-07-18 14:37:43	2007-07-18 14:38:13
Counter Name	Periodic value	Cumulative value
Elapsed Time	00:00:01:009	00:00:30:300
Call rate (/s)	0.991	0.132
Incoming calls	1	4
Outgoing calls	0	0
Msg Recv/s	2.973	0.231
Msg Sent/s	2.973	0.231
Unexpected msg	0	0
Current calls	1	0.033

Successful calls	2	3
Failed calls	0	0
Refused calls	0	0
Aborted calls	0	0
Timeout calls	0	0
Last Info	Incomming traffic	
Last Error	No error	
--- Next screen : Press key 1 ----- [h]: Display help -----		

## 2.2. First try explained

Here is the script (start\_client.ksh) that launched the client in our example:

```
#!/bin/ksh
export LD_LIBRARY_PATH=/usr/local/bin
seagull -conf ../config/conf.client.xml -dico ../config/radius-accounting.xml -scen
../scenario/radius-accounting.client.xml -log ../logs/radius-accounting.client.log
-llevel ET
```

**Note:**

On some systems, you might need to include the following export in your Seagull script: "export SHLIB\_PATH=/usr/local/bin".

Both sides are relying on the Radius Accounting dictionary provided with Seagull: radius-accounting.xml to encode Radius messages. Refer to [dictionary configuration](#) section for more information on the format of this dictionary. The dictionary is specified using the -dico parameter on the [command line](#) (core.html#cli\_help) .

The generic configuration (including network and other parameters) is different for the client and the server. The client uses conf.client.xml and the server uses conf.server.xml. The configuration file is specified using the -conf parameter on the [command line](#) (core.html#cli\_help) .

Here are both files:

conf.client.xml	conf.server.xml
<pre>&lt;?xml version="1.0" encoding="ISO-8859-1"?&gt; &lt;configuration name="Simple Radius Accounting TCP/IP Client Conf"&gt;    &lt;define entity="transport"     name="trans-ip-v4"     file="libtrans_ip.so"     create_function="create_cipio_instanc delete_function="delete_cipio_instanc     init-args="type=tcp"&gt;   &lt;/define&gt;    &lt;define entity="channel"     name="trans-ip-v4"     protocol="radius-accounting-v1"     transport="trans-ip-v4"     open-args="mode=client;dest=127.0.0.1   &lt;/define&gt;    &lt;define entity="traffic-param"     name="call-rate" value="1"&gt;&lt;/define&gt;   &lt;define entity="traffic-param"</pre>	<pre>&lt;?xml version="1.0" encoding="ISO-8859-1"?&gt; &lt;configuration name="Simple Radius TCP/IP Server Conf"&gt;   &lt;define entity="transport"     name="trans-ip-v4"     file="libtrans_ip.so"     create_function="create_cipio_instanc"     delete_function="delete_cipio_instanc"     init-args="type=tcp"&gt;   &lt;/define&gt;    &lt;define entity="channel"     name="trans-ip-v4"     protocol="radius-accounting-v1"     transport="trans-ip-v4"     open-args="mode=server;source=127.0.0.1:1813"&gt;   &lt;/define&gt;    &lt;define entity="traffic-param"     name="display-period"     value="1"&gt;&lt;/define&gt;</pre>

<pre> name="display-period" value="1"&gt;&lt;/define&gt;   &lt;define entity="traffic-param" name="log-stat-period" value="1"&gt;&lt;/define&gt;   &lt;define entity="traffic-param" name="log-stat-file" value=" ../logs/client-stat.csv"&gt;&lt;/define&gt;   &lt;define entity="traffic-param" name="call-timeout-ms" value="10000"&gt;&lt;/define&gt;   &lt;define entity="traffic-param" name="display-scenario-stat" value="true"&gt;&lt;/define&gt;   &lt;define entity="traffic-param" name="display-protocol-stat" value="true"&gt;&lt;/define&gt;   &lt;define entity="traffic-param" name="log-protocol-stat-period" value="5"&gt;&lt;/define&gt;   &lt;define entity="traffic-param" name="log-protocol-stat-name" value="all"&gt;&lt;/define&gt;   &lt;define entity="traffic-param" name="log-protocol-stat-file" value=" ../logs/client-protocol-stat.csv"&gt;&lt;/define&gt;    &lt;define entity="traffic-param" name="max-send" value="200"&gt;&lt;/define&gt;   &lt;define entity="traffic-param" name="max-receive" value="200"&gt;&lt;/define&gt;   &lt;define entity="traffic-param" name="max-simultaneous-calls" value="2000"&gt;&lt;/define&gt;   &lt;define entity="traffic-param" name="select-timeout-ms" value="50"&gt;&lt;/define&gt;    &lt;define entity="traffic-param" name="external-data-file" value=" ../scenario/external_client_data.csv"&gt;&lt;/define&gt;   &lt;define entity="traffic-param" name="external-data-select" value="sequential"&gt;&lt;/define&gt; &lt;/configuration&gt; </pre>	<pre>   &lt;define entity="traffic-param" name="log-stat-period" value="1"&gt;&lt;/define&gt;   &lt;define entity="traffic-param" name="log-stat-file" value=" ../logs/server-stat.csv"&gt;&lt;/define&gt;    &lt;define entity="traffic-param" name="call-timeout-ms" value="10000"&gt;&lt;/define&gt;   &lt;define entity="traffic-param" name="display-scenario-stat" value="true"&gt;&lt;/define&gt;   &lt;define entity="traffic-param" name="display-protocol-stat" value="true"&gt;&lt;/define&gt;   &lt;define entity="traffic-param" name="log-protocol-stat-period" value="5"&gt;&lt;/define&gt;   &lt;define entity="traffic-param" name="log-protocol-stat-name" value="all"&gt;&lt;/define&gt;   &lt;define entity="traffic-param" name="log-protocol-stat-file" value=" ../logs/server-protocol-stat.csv"&gt;&lt;/define&gt;    &lt;define entity="traffic-param" name="max-send" value="2000"&gt;&lt;/define&gt;   &lt;define entity="traffic-param" name="max-receive" value="2000"&gt;&lt;/define&gt;   &lt;define entity="traffic-param" name="max-simultaneous-calls" value="2000"&gt;&lt;/define&gt;   &lt;define entity="traffic-param" name="select-timeout-ms" value="50"&gt;&lt;/define&gt;    &lt;define entity="traffic-param" name="external-data-file" value=" ../scenario/external_server_data.csv"&gt;&lt;/define&gt;   &lt;define entity="traffic-param" name="external-data-select" value="sequential"&gt;&lt;/define&gt; &lt;/configuration&gt; </pre>
---	---

**Table 1: Example client and server configuration**

Now comes the real stuff: the scenario.

First, the scenario source: [radius-accounting.client.xml](#) (radius-accounting.client.xml.html)

And now the commented version:

Scenario	Comments
<pre> &lt;?xml version="1.0" encoding="ISO-8859-1" ?&gt;  &lt;scenario name="Simple Radius Accounting Client scenario"&gt; &lt;counter&gt; </pre>	<pre> XML header Scenario name Counters definitions In this section are initialized the counters that will </pre>

<pre> &lt;counterdef name="session-counter" init="0"&gt; &lt;/counterdef&gt; &lt;counterdef name="identifier-counter" init="0"&gt; &lt;/counterdef&gt; &lt;/counter&gt;  &lt;!-- Initialisation Scenario &lt;init&gt; &lt;/init&gt; --&gt;  &lt;!-- Traffic Scenario --&gt; &lt;traffic&gt;    &lt;!-- Start Accounting Request --&gt;   &lt;send channel="trans-ip-v4"&gt;     &lt;action&gt;       &lt;!-- For each new call, increment the Acct-Session-Id counter --&gt;       &lt;inc-counter name="session-counter"&gt; &lt;/inc-counter&gt;       &lt;inc-counter name="identifier-counter"&gt; &lt;/inc-counter&gt;       &lt;set-value name="Acct-Session-Id" format="\$(session-counter)"&gt;&lt;/set-val       &lt;set-value name="Identifier" format="\$(identifier-counter)"&gt;&lt;/set-        &lt;!-- This field must not be use with a true value --&gt;       &lt;restore-from-external field="0" entity="Authenticator"&gt;&lt;/restore-from      &lt;/action&gt;     &lt;Message name="Accounting-Request"&gt;       &lt;Attribute name="Acct-Session-Id" value="value_is_replaced"&gt; &lt;/Attribute&gt;       &lt;!-- 1 stand for Start --&gt;       &lt;Attribute name="Acct-Status-Type" value="1"&gt; &lt;/Attribute&gt;       &lt;Attribute name="NAS-Identifier" value="ims.hpintelco.org"&gt; &lt;/Attribute&gt;     &lt;/Message&gt;     &lt;action&gt;       &lt;store name="ACCT_SES_ID" entity="Acct-Session-Id"&gt;&lt;/store&gt;       &lt;start-timer&gt;&lt;/start-timer&gt;     &lt;/action&gt;   &lt;/send&gt;    &lt;receive channel="trans-ip-v4"&gt;     &lt;action&gt;       &lt;stop-timer&gt;&lt;/stop-timer&gt;     &lt;/action&gt; </pre>	<p>then be used during the calls. For example here: we initialize the two counter that will be incremented every time a call is started.</p> <p>No init section</p> <p>Traffic section</p> <p>Actions before sending the first Accounting-Request message: Increment the session-counter counter. Increment the identifier-counter counter. Set the field Acct-Session-Id with the counter session-counter Set the field Acct-Session-Id with the counter session-counter</p> <p>restore-from-external gets some data from an external file into the Authenticator field.</p> <p>Send the Accounting-Request message with 3 attributes The Acct-Session-Id attribute is added to the message, the value had been changed with set-value action</p> <p>The Acct-Status-Type attribute is added to the message with the value 1 The NAS-Identifier attribute is added to the message with the value "ims.hpintelco.org"</p> <p>Actions after sending the first Accounting-Request message: store the value of the field Acct-Session-Id into the variable ACCT_SES_ID start the timer (to measure response time)</p> <p>Actions before receiving Accounting-Response stop the timer</p> <p>Receive the Accounting-Response message with 2 attributes The real values of attributes are</p>
--	--

<pre> &lt;Message name="Accounting-Response"&gt;   &lt;Attribute name="Acct-Session-Id" value="na"&gt; &lt;/Attribute&gt;   &lt;!-- 1 stand for Start --&gt;   &lt;Attribute name="Acct-Status-Type" value="1"&gt; &lt;/Attribute&gt; &lt;/Message&gt; &lt;/receive&gt;  &lt;wait-ms value="500"&gt;&lt;/wait-ms&gt;  &lt;!-- Stop Accounting Request --&gt; &lt;send channel="trans-ip-v4"&gt;   &lt;action&gt;     &lt;!-- For each new call, increment the Acct-Session-Id counter --&gt;     &lt;inc-counter name="identifier-counter"&gt; &lt;/inc-counter&gt;     &lt;set-value name="Identifier" format="\$(identifier-counter)"&gt;&lt;/set- value&gt;     &lt;!-- This field must not be use with a true value --&gt;     &lt;restore-from-external field="1" entity="Authenticator"&gt;&lt;/restore-from- external&gt;     &lt;restore name="ACCT_SES_ID" entity="Acct-Session-Id"&gt;&lt;/restore&gt;   &lt;/action&gt;   &lt;Message name="Accounting-Request"&gt;     &lt;Attribute name="Acct-Session-Id" value="value_is_replaced"&gt; &lt;/Attribute&gt;     &lt;!-- 2 stand for Stop --&gt;     &lt;Attribute name="Acct-Status-Type" value="2"&gt; &lt;/Attribute&gt;     &lt;Attribute name="NAS-Identifier" value="ims.hpintelco.org"&gt; &lt;/Attribute&gt;   &lt;/Message&gt;   &lt;action&gt;     &lt;start-timer&gt;&lt;/start-timer&gt;   &lt;/action&gt; &lt;/send&gt;  &lt;receive channel="trans-ip-v4"&gt;   &lt;action&gt;     &lt;stop-timer&gt;&lt;/stop-timer&gt;   &lt;/action&gt;   &lt;Message name="Accounting-Response"&gt;     &lt;Attribute name="Acct-Session-Id" value="na"&gt; &lt;/Attribute&gt;     &lt;!-- 2 stand for Stop --&gt;     &lt;Attribute </pre>	<pre> set by the remote.  Make a pause in scenario (of 500ms)  Actions before sending the Accounting-Request message  Increment the identifier-counter set the value of the field Identifier with the identifier-counter  restore an external value into the field authenticator  restore the value of the variable ACCT_SES_ID into the field Acct-Session-Id  Send the message Accounting-Request The Acct-Session-Id attribute is added to the message, the value had been changed with restore action  The Acct-Status-Type attribute is added to the message with the value 2 The NAS-Identifier attribute is added to the message with the value "ims.hpintelco.org"  Actions after sending the first Accounting-Request message: start the timer (to measure response time)  Actions before receiving Accounting-Response stop the timer  Receive the Accounting-Response message with 2 attributes The real values of attributes are set by the remote </pre>
---	---

```

name="Acct-Status-Type" value="2">
</Attribute>
</Message>
</receive>

</traffic>

</scenario>

```

### 3. Configuration files

There are 3 different configuration files:

- [Generic](#) configuration file - describing traffic and network parameters
- [Protocol dictionary](#) configuration file - rarely to be edited
- [Scenario](#) file - description of the message exchanges

#### 3.1. Generic configuration

The generic configuration file describes the network environment as well as traffic parameters.

The network environment is described through "[transport channel entities](#) (core.html#ref\_transport) ". The transport entity is then used as an attribute of [send](#) (core.html#cmd\_send) and [receive](#) (core.html#cmd\_receive) scenario commands, as well as during the opening of the transport channel (see below).

```

<!-- Radius example -->
<?xml version="1.0" encoding="ISO-8859-1"?>
<configuration name="Simple Radius Accounting TCP/IP Client Conf">

<!-- Define the transport to be used: -->
  <define entity="transport"
    name="trans-ip-v4"
    file="libtrans_ip.so"
    create_function="create_cipio_instance"
    delete_function="delete_cipio_instance"
    init-args="type=tcp">
  </define>

<!-- Then you specify the opening of the channel, on the transport previously
described. -->

<!-- For a server listening to port 1813 on interface "127.0.0.1", it will look like
this: -->
  <define entity="channel"
    name="trans-ip-v4"
    protocol="radius-accounting-v1"
    transport="trans-ip-v4"
    open-args="mode=server;source=127.0.0.1:1813">
  </define>

<!-- For a client sending messages to port 1813 on interface "127.0.0.1", it will look
like this: -->
  <define entity="channel"
    name="trans-ip-v4"
    protocol="radius-accounting-v1"
    transport="trans-ip-v4"
    open-args="mode=client;dest=127.0.0.1:1813">
  </define>

```

## 3.2. Radius dictionary

Radius messages and parameters are described in XML dictionaries. The tool comes with a complete Radius dictionary.

A dictionary contains several XML sections

### 3.2.1. Types

"types" section contains all types needed for the protocol. For Diameter, these are:

```
<types>
  <!-- Types defined for the Attributes -->
  <typedef name="Octet" type="number" size="1" unit="octet"> </typedef>
  <typedef name="Integer32" type="signed" size="4" unit="octet"> </typedef>
  <typedef name="Unsigned32" type="number" size="4" unit="octet"> </typedef>
  <typedef name="Address" type="number" size="4" unit="octet"> </typedef>
  <typedef name="Time" type="number" size="4" unit="octet"> </typedef>
  <typedef name="String" type="string" unit="octet"> </typedef>
  <typedef name="Text" type="string" unit="octet"> </typedef>
  <typedef name="Grouped" type="grouped" > </typedef>

  <typedef name="Octet_16" type="string" size="16" unit="octet"> </typedef>
  <typedef name="AddressV6" type="string" size="16" unit="octet"> </typedef>
  <typedef name="InterfaceId" type="string" size="8" unit="octet"> </typedef>
</types>
```

### 3.2.2. Header

"header" section contains the description of message header. For Radius, this is:

```
<header name="Message" length="Length" type="Code">
  <fielddef name="Code" size="1" unit="octet"> </fielddef>
  <fielddef name="Identifier" size="1" unit="octet"> </fielddef>
  <fielddef name="Length" size="2" unit="octet"> </fielddef>
  <fielddef name="Authenticator" type="Octet_16"> </fielddef>
</header>
```

### 3.2.3. Body

"body" section contains the description of message body (which naturally comes after the header). For Radius, this is:

```
<body>
  <header name="Attribute" length="Attr-Length" type="Attr-Type">
    <fielddef name="Attr-Type" size="1" unit="octet"> </fielddef>
    <fielddef name="Attr-Length" size="1" unit="octet"> </fielddef>
  </header>
</body>
```

### 3.2.4. Dictionary

"dictionary" section contains all possible parameters that a message can contain. Here is a description for some Radius attributes:

```
<dictionary>
  <!-- RADIUS Attribute definitions -->
  <Attribute>
```

```

<!-- RADIUS Base Attribute definitions see RFC 2865 -->
<define name="User-Name" type="String">
  <setfield name="Attr-Type" value="1"></setfield>
</define>
<define name="User-Password" type="String">
  <setfield name="Attr-Type" value="2"></setfield>
</define>
<define name="CHAP-Password" type="String">
  <setfield name="Attr-Type" value="3"></setfield>
  <!-- composed by CHAP-Id Octect and String -->
</define>
<define name="NAS-IP-Address" type="Address">
  <setfield name="Attr-Type" value="4"></setfield>
</define>
<define name="NAS-Port" type="Integer32">
  <setfield name="Attr-Type" value="5"></setfield>
</define>
<define name="Service-Type" type="Integer32">
  <setfield name="Attr-Type" value="6"></setfield>
</define>
</Attribute>
</dictionary>

```

### 3.2.5. Message

"Message" section contains all possible Diameter messages (from RFC 2865 and 2866). Here is an example of Radius messages:

```

<define name="Access-Request">
  <setfield name="Code" value="1"></setfield>
</define>
<define name="Access-Accept">
  <setfield name="Code" value="2"></setfield>
</define>

```

### 3.3. Actions in scenario commands for Radius

The <send> and <receive> scenario commands include an <action> and <message> sections.

The <action> section can be placed before or after the <message> section.

Actions placed before the message (called "**pre-actions**") are executed before the message is actually sent or received. Actions placed after the message (called "**post-actions**") are executed after the message is sent or received.

For example, actions that can be placed before a message are actions to set the values of some ids or to set some fields using external values, before sending a message. Example:

```

<send channel="trans-ip-v4">
  <action>
    <!-- For each new call, increment the Acct-Session-Id counter -->
    <inc-counter name="session-counter"> </inc-counter>
    <inc-counter name="identifier-counter"> </inc-counter>
    <set-value name="Acct-Session-Id"
      format="$(session-counter)"></set-value>
    <set-value name="Identifier"
      format="$(identifier-counter)"></set-value>

    <!-- This field must not be use with a true value -->
    <restore-from-external field="0" entity="Authenticator"></restore-from-external>

  </action>

```

Actions that can be placed after a message are actions to store some values after the message has been received. Example:

```
<action>
  <store name="ACCT_SES_ID" entity="Acct-Session-Id"></store>
</action>
</send>
```

The list of [possible actions](#) is available in the reference section.

## 4. Radius authentication

The authentication implementation in Radius protocol is to be used as an example. The "crypto\_method" method, used for the authentication, is defined in ["library-crypto/CryptExternalMethods.hpp"](http://gull.svn.sourceforge.net/viewvc/gull/seagull/trunk/src/library-crypto/CryptExternalMethods.hpp?view=markup) (<http://gull.svn.sourceforge.net/viewvc/gull/seagull/trunk/src/library-crypto/CryptExternalMethods.hpp?view=markup>) and implemented in ["library-crypto/CryptExternalMethods.cpp"](http://gull.svn.sourceforge.net/viewvc/gull/seagull/trunk/src/library-crypto/CryptExternalMethods.cpp?view=markup) (<http://gull.svn.sourceforge.net/viewvc/gull/seagull/trunk/src/library-crypto/CryptExternalMethods.cpp?view=markup>) in the sources.

It can be modified to match the exact authentication algorithm.

### 4.1. Seagull implementation

At first, the message is built with the "Authenticator" field filled with '0000000000000000' in the scenario:

```
<Message name="Accounting-Request">
  <setfield name="Authenticator" value="0000000000000000"> </setfield>
  ...
```

In the pre-action part of the scenario, the "Authenticator" field is set with the result of the authentication method:

```
<set-value name="Authenticator" method="authentication" message_part="all"
  format="shared_secret=${SH_SE}"></set-value>
```

The "method" field selects the function to be used for the authentication from the ones defined in the dictionary.

The "message\_part" defines which part of the message is used for the authentication (in this case, the entire message is used).

The "format" defines additional parameters if the method needs some (In this case, the "shared\_secret" parameter is passed to the method with the value of the memory zone "SH\_SE". The memory zone "SH\_SE" can be filled by a "restore from\_external" action for example).

### 4.2. Authentication dictionary

The authentication method must be defined in the dictionary with its name, the name of the function to be used and the name of the library in which the function is defined.

Example:

```
<!-- external methods for fields modifications -->
<external-method>
  <defmethod name="authentication"
    param="lib=lib_crypto.so;function=crypto_method_radius">
  </defmethod>
</external-method>
```

The authentication method "name" is used in the scenario to call the authentication function.

The "param" defines the name of the library "lib=lib\_crypto.so" and the name of the function "function=crypto\_method\_radius".

### 4.3. Authentication scenario

Here is the commented version of the Radius client using the "authentication" method defined in "library-crypto/CryptExternalMethods.hpp" and implemented in "library-crypto/CryptExternalMethods.cpp" in the sources

Client XML scenario.	Comments
<pre> &lt;?xml version="1.0" encoding="ISO-8859-1" ?&gt;  &lt;scenario name="Simple Radius Accounting Client scenario"&gt;  &lt;counter&gt;   &lt;counterdef name="session-counter" init="0"&gt; &lt;/counterdef&gt;   &lt;counterdef name="identifier-counter" init="0"&gt; &lt;/counterdef&gt; &lt;/counter&gt;  &lt;!-- Traffic Scenario --&gt; &lt;traffic&gt;    &lt;!-- Start Accounting Request --&gt;   &lt;send channel="trans-ip-v4"&gt;     &lt;action&gt;       &lt;!-- For each new call, increment the Acct-Session-Id counter --&gt;       &lt;inc-counter name="session-counter"&gt; &lt;/inc-counter&gt;       &lt;inc-counter name="identifier-counter"&gt; &lt;/inc-counter&gt;       &lt;set-value name="Acct-Session-Id" format="\$(session-counter)"&gt;&lt;/set-val       &lt;set-value name="Identifier" format="\$(identifier-counter)"&gt;&lt;/set-       &lt;set-value name="Authenticator" method="authentication" message_part="all" format=""&gt;&lt;/set-value&gt;      &lt;/action&gt;     &lt;Message name="Accounting-Request"&gt;       &lt;setfield name="Authenticator" value="0000000000000000"&gt; &lt;/setfield&gt;       &lt;Attribute name="Acct-Session-Id" value="value_is_replaced"&gt; &lt;/Attribute&gt;       &lt;!-- 1 stand for Start --&gt;       &lt;Attribute name="Acct-Status-Type" value="1"&gt; &lt;/Attribute&gt; </pre>	<pre> XML header  The counters are declared ("session-counter" and "identifier-counter")  Send on "trans-ip-v4" channel as defined in the config file Pre-actions for each Accounting-Request message to be sent...  For each new call, increment the session-counter and identifier-counter counters  Set the field "Acct-Session-Id" with "session-counter"  Set the field "Identifier" with "identifier-counter"  Set the field "Authenticator". The "authentication" method is defined in the dictionary. It uses the entire message as defined in message_part="all". The result of the method is placed in the field after the generation of the rest of the message.  Send an "Accounting-Request" message that is initialized as follows before the above pre-actions are applied: Set the field "Authenticator" with a default value "0000000000000000". The value is replaced by the "set-value" action at the execution of the scenario. Set the attribute "Acct-Session-Id" with a default value "value_is_replaced". The value is replaced by the "set-value" action </pre>

```
<Attribute
name="NAS-Identifier"
value="ims.hpintelco.org">
</Attribute>
</Message>
<action>
  <store name="ACCT_SES_ID"
entity="Acct-Session-Id"></store>
  <start-timer></start-timer>
</action>
...
</traffic>
</scenario>
```

at the execution of the scenario.

Set the attribute "Acct-Status-Type" with a default value "1".  
Set the attribute "NAS-Identifier" with a default value "ims.hpintelco.org".

Post-actions for each Accounting-Request message that is sent...

Store the value of the field "Acct-Session-Id" in the memory zone "ACCT\_SES\_ID". It is used to identify the session.