Radiator® 3GPP AAA Server

Installation and reference manual for Radiator® 3GPP AAA Server 2.7. Last revised on November 18, 2021

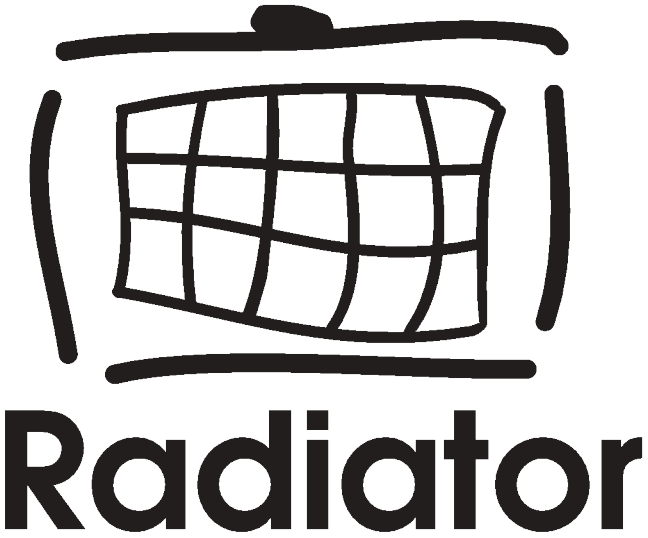

Table of Contents

1. Introduction to Radiator 3GPP AAA Server

This document describes how to install and configure the Radiator 3GPP AAA Server.

For more information about the 3GPP AAA Server in general, see Radiator 3GPP AAA Server whitepaper [https://files.radiatorsoftware.com/radiator/whitepapers/radiator-3gpp-aaa-server-whitepaper.pdf].

2. Installing Radiator 3GPP AAA Server

This section the describes how to install Radiator 3GPP AAA Server.

For installing Radiator 3GPP AAA Server, you need a database with sample tables created from `goodies/3gpp-aaa-server-mysql.sql`. Other types of databases and alternative database schemas are also supported.

2.1. Prerequisites

To be able to install Radiator 3GPP AAA Server, your system must meet these prerequisites:

- Radiator 4.26 or later.
- Radiator Carrier module 1.7 or later.
- `Radius::UtilXS` 2.2 or later is recommended. Otherwise additional modules from CPAN, as shown below, are needed.
- SCTP multihoming requires Radiator `Radius::UtilXS` module.
- IMSI encryption requires Radiator `Radius::UtilXS` module 2.0 or later.
- The following Perl modules:
  - `Radius::UtilXS` 2.2 or later - From Radiator downloads, or `Crypt::Rijndael`
  - `DBI`
  - `Radius::UtilXS` 2.0 or later - From Radiator downloads, or `Digest::SHA1`
  - `Digest::SHA` - Typically already installed with system Perl

2.2. Installation

Procedure

The recommended method is to install Radiator, Radiator Carrier and Radiator SIM Module from operating system specific packages. If required, source code installation is also possible. Operating system specific packages are available as stand-alone downloads and as repositories that integrate with operating system package management tools, such as **`yum`** and **`apt`**. See Radiator download site for detailed repository instructions.

1. Download the Radiator SIM Module distribution.
2. Unpack the file into a separate working directory.
3. Move to the distribution directory.
4. Prepare the distribution for installation.

   ```
   perl Makefile.PL
   ```

5. Run the installation. You may need the root access rights for running this command.

   ```
   make install
   ```

6. Build a Radiator configuration file based on `goodies/3gpp-aaa-server.cfg`.
7. Configure Radiator HSS (Home Subscriber Server) emulator file (`goodies/server_hss.cfg`) or connect to the HSS over SWx.
8. Run Radiator with the configuration file developed in the step 6.
9. Test and refine the configuration file. For more information, see Testing Radiator 3GPP AAA Server on page 2.
10. Set Radiator to start automatically when booting. For more information, see Radiator reference manual [https://files.radiatorsoftware.com/radiator/ref.pdf].

3. Testing Radiator 3GPP AAA Server

This section describes the test scenarios for Radiator 3GPP AAA Server.

To prepare the test setup:

1. Download the latest versions of Radiator, Radiator Carrier Pack, and Radiator SIM Module. Radiator 3GPP AAA Server is part of the Radiator SIM Module.
2. Install the downloaded softwares.
3. Create a separate directory for testing.
4. Copy `goodies/simcards.dat` from Radiator SIM Pack directory into the testing directory.
5. Copy all configuration files (`*.cfg`) into the testing directory.
6. Start the *`radiusd`* processes in the following order:
   a. 3GPP AAA Server
   b. HSS for processing MAR (Multimedia-Auth-Request) and SAR (Server-Assignment-Request)
   c. S6b for processing ASR (Abort-Session-Request)
   d. RADIUS/EAP-AKA (Extensible Authentication Protocol - Authentication and Key Agreement) to Diameter/SWm conversion. This requires Radiator 4.16 with patches or newer.

   To start these *`radiusd`* processes, execute the following commands in the testing directory:

```
radiusd -dictionary /etc/radiator/dictionary -log_stdout -foreground -trace 4
      -config 3gpp-aaa-server.cfg

radiusd -dictionary /etc/radiator/dictionary -log_stdout -foreground -trace 4
      -config server-hss.cfg

radiusd -dictionary /etc/radiator/dictionary -log_stdout -foreground -trace 4
      -config server-s6b.cfg

radiusd -dictionary /etc/radiator/dictionary -log_stdout -foreground -trace 4
      -config server-swm.cfg
```

7. If you need a Radius to Diameter conversion, start the process with the following command in the testing directory:

```
radiusd -dictionary /etc/radiator/dictionary -log_stdout -foreground -trace 4
      -config radius-eap-convert.cfg
```

3.1. Compiling `eapol_test`

`eapol_test` is a part of wpa_supplicant suite [http://w1.fi/wpa_supplicant/]. It is a tool for testing Radiator EAP-SIM (Extensible Authentication Protocol - Subscriber Identity Module), EAP-AKA, and EAP-AKA' (Extensible Authentication Protocol - Authentication and Key Agreement Prime) protocols. You can configure it to act as a supplicant to generate RADIUS requests which are sent directly to the RADIUS server. With `eapol_test`, you can test the system without the client, supplicant, and wireless access point.

Note

The `eapol_test` configuration `.config` file located in `wpa_supplicant2.4/wpa_supplicant/`. After configuring it, always rerun `make eapol_test` because the `eapol_test` target is not a part of the default `make` target.

Enabling AKA methods and USIM simulator

For EAP-AKA and EAP-AKA' the Milenage parameters are defined in format **`password="Ki:OPc:SQN"`** in `eapol_test .config` file.

To enable the AKA (Authentication and Key Agreement) methods and USIM (Universal Subscriber Identity Module) simulator:

```
echo CONFIG_EAP_AKA=y >> .config
```

```
echo CONFIG_EAP_AKA_PRIME=y >> .config
```

```
echo CONFIG_USIM_SIMULATOR=y >> .config
```

```
make eapol_test
```

3.2. Testing SWm requests sent by ePDG

About this task

This test scenario tests the functionality of Diameter SWm connection between the ePDG (Evolved Packet Data Gateway) and Radiator 3GPP AAA Server.

Procedure

To test EAP authentication initiated by an end user:

1. Open `aka-simulator.conf`, located in Radiator SIM Pack's `/goodies` directory, and check that identity is *`0232010000000000@nai.epc.mnc001.mcc232.3gppnetwork.org`*.
2. Run `eapol_test` to make sure the authentication works correctly. This sends a RADIUS EAP-AKA request to a Radiator radiusd instance started with `radius-eap-convert.cfg`. The request is converted to a DER (Diameter EAP Request) and sent over SWm.

To test other requests sent by ePDG:

1. Send an AAR (AA Request) over SWm. Use the correct *`-session_id`* from a previous DER. To do this, execute the following command in the Radiator SIM Pack directory:

```
perl goodies/diapwtst-3gpp
        -trace 4 -swm aar -originhost epdg3.epc.mnc001.mcc232.3gppnetwork.org
        -originrealm epc.mnc001.mcc232.3gppnetwork.org -user
        232010000000000@nai.epc.mnc001.mcc232.3gppnetwork.org -session_id
```

```
        'epdg.epc.mnc001.mcc232.3gppnetwork.org;1450276781;831118;0'
```

Note

Use *epdg3.epc.mnc001.mcc232.3gppnetwork.org* because *epdg.epc.mnc001.mcc232.3gppnetwork.org* already has a session with Radiator 3GPP AAA Server.

2. Send an STR (Session-Termination-Request) over SWm to terminate the session started with EAP-AKA. Use the correct *-session_id*. To do this, execute the following command in the Radiator SIM Pack directory:

```
perl
        goodies/diapwtst-3gpp -trace 4 -swm str -originhost
        epdg3.epc.mnc001.mcc232.3gppnetwork.org -originrealm
        epc.mnc001.mcc232.3gppnetwork.org -user
        232010000000000@nai.epc.mnc001.mcc232.3gppnetwork.org
        Termination-Cause=DIAMETER_LOGOUT -session_id
        'epdg.epc.mnc001.mcc232.3gppnetwork.org;1450276781;831118;0'
```

3.3. Testing S6b requests sent by PDG

About this task

This test scenario tests if the user session is created to the PDN GW (Packet Data Network Gateway).

Before you begin

You need to have SWm session for the tested IMSI (International mobile subscriber identity). For more information, see Testing SWm requests sent by ePDG on page 3.

Procedure

To execute the test:

1. Send an AAR over S6b to create a process. To do this, execute the following command in the Radiator SIM Pack directory:

```
perl goodies/diapwtst-3gpp -trace 4 -s6b aar
     -originhost pgw.epc.mnc001.mcc232.3gppnetwork.org
     -originrealm epc.mnc001.mcc232.3gppnetwork.org
     -user 232010000000000@nai.epc.mnc001.mcc232.3gppnetwork.org
```

2. Send an STR over S6b to terminate the process created in the previous step. Use the correct *-session_id*. To do this, execute the following command in the Radiator SIM Pack directory:

```
perl goodies/diapwtst-3gpp -trace 4 -s6b str
     -originhost pgw.epc.mnc001.mcc232.3gppnetwork.org
     -originrealm epc.mnc001.mcc232.3gppnetwork.org
     -user 232010000000000@nai.epc.mnc001.mcc232.3gppnetwork.org
     Termination-Cause=DIAMETER_LOGOUT
     -session_id 'pgw.epc.mnc001.mcc232.3gppnetwork.org;1234;1'
```

3.4. Testing SWx requests sent by HSS

About this task

This test scenario tests the user session termination.

Before you begin

You need to have SWm session for the tested IMSI. For more information, see Testing SWm requests sent by ePDG on page 3.

Procedure

To execute the test:

1. Send an RTR (Registration-Termination-Request) over SWx to terminate all SWm, S6b, STa, and SWa sessions Radiator 3GPP AAA Server has for a specific IMSI. To do this, execute the following command in the Radiator SIM Pack directory:

```
perl goodies/diapwtst-3gpp -trace 4
     -originhost hss2.aaa.mnc001.mcc232.3gppnetwork.org
     -originrealm aaa.mnc001.mcc232.3gppnetwork.org
     -desthost radiator-3gpp.aaa.mnc001.mcc232.3gppnetwork.org
     -swx rtr -user 232010000000000
```

2. Send an PPR (Push-Profile-Request) over SWx to replace the profile and trigger reauthentication for SWm, STA, and SWa sessions Radiator 3GPP AAA Server has for the IMSI. To do this, execute the following command in the Radiator SIM Pack directory:

```
perl goodies/diapwtst-3gpp -trace 4
     -originhost hss2.aaa.mnc001.mcc232.3gppnetwork.org
     -originrealm aaa.mnc001.mcc232.3gppnetwork.org
     -desthost radiator-3gpp.aaa.mnc001.mcc232.3gppnetwork.org
     -swx ppr -user 232010000000000
```

Note

You can add *PPR-Flags=1* to the command to send *PPR-Flags* parameter with the desired value.

Note

If *Origin-Host hss.aaa.mnc001.mcc232.3gppnetwork.org* is used, `3gpp-aaa-server.cfg` instance must be restarted after RTR or other request over SWx.

4. Configuring Radiator 3GPP AAA Server

This section describes the configurable parameters of Radiator 3GPP AAA Server.

4.1. <3GPPAuthHSS>

This section describes the configuring parameters of *<3GPPAuthHSS>*.

4.1.1. AAAServerSWx

This object list defines the AAA (Authentication, Authorisation, Accounting) Server SWx clause to be used.

4.1.2. HSSRealm

This string defines the Diameter realm that is used as Destination-Realm for SWx messages. If *HSSRealm* is not defined, *DiaPeerDef* entries need to be configured and DestinationRealm from Diameter Peer's DiaPeerDef configuration is used as the Destination-Realm. If no *HSSRealm* or Destination-Realm for the chosen DiaPeerDef is configured, Destination-Realm in the outgoing SWx request is empty.

Starting with Radiator 4.24, the recommended configuration is to define *HSSRealm* and leave *DiaPeerDef* parameters undefined. This allows Radiator to use Diameter routing to resolve the peer to send SWx requests to.

```
# Always use this as SWx Destination-Realm in requests
HSSRealm aaa.mnc001.mcc001.3gppnetwork.org
```

4.1.3. DiaPeerDef

DiaPeerDef defines how to select the peer to use when sending SWx requests to HSS. The HSS may be a directly connected peer, or reachable with Diameter Routing Agent or some other type of agent that routes the messages towards the HSS. *DiaPeerDef* parameter value can be a configuration file *Identifier* value or Diameter Auth-Application-Id or Vendor-Specific-Application-Id/Auth-Application-Id that the peer has advertised.

Multiple instances of *DiaPeerDef* are allowed. The first entry is the primary peer to use. Entries will be tried in the order they appear in the configuration file.

Starting with Radiator 4.24, the recommended configuration is to define *HSSRealm* and leave *DiaPeerDef* parameters undefined. This allows Radiator to use Diameter routing to resolve the peer to send SWx requests to.

```
# We peer directly with HSS
<DiaPeerDef ...>
    Identifier diapeer-hss
</DiaPeerDef>

# The configuration file Identifier to locate DiaPeerDef to use with HSS requests
DiaPeerDef DiaPeerDef-Identifier=diapeer-hss
#DiaPeerDef DiaPeerDef-Identifier=diapeer-hss-secondary

# An alternative is to use a peer that has advertised SWx
#DiaPeerDef Peer-Auth-Application-Id=3GPP:3GPP SWx
```

4.2. <DiaPeerDef>

This section describes the configuration parameters for <*DiaPeerDef*>. <*DiaPeerDef*> defines the Diameter peer this Radiator instance connects to. Both Radiator instance and the Diameter peer can initiate the connection.

A minimal Radiator 3GPP AAA Server configuration requires one <*DiaPeerDef*> clause for all used Diameter-based AuthBys. If there is no <*ServerDIAMETERTelco*> clause defined, <*DiaPeerDef*> clauses must have the *Initiator* flag set to connect to the Diameter peers.

A <*ServerDIAMETERTelco*> clause allows accepting incoming Diameter connections. When the <*ServerDIAMETERTelco*> is configured, Radiator acts as a Diameter responser. The settings for the

connecting peers are fetched from the <DiaPeerDef> clauses.The clauses are matched against the incoming CER (Capabilities Exchange Request) from the peer.

Note

At least one <DiaPeerDef> clause is always required.

If the <ServerDIAMETERTelco> clause is configured but there are no <DiaPeerDef> clauses, the incoming CER messages are rejected by Radiator. A <DiaPeerDef> is required to form a successful CEA (Capabilities Exchange Answer) back to the peer.

Note

A <DiaPeerDef> with an empty parameter list matches to any Diameter peer. This is useful when defining default settings for incoming connections from any Diameter peer.

4.2.1. Identifier

This is an optional parameter, which defines the name of the specific <DiaPeerDef> clause and its configuration. When defined, this allows you to choose the correct Diameter peer when configuring Diameter-relaying support.

4.2.2. AddToRequestFromDia

This parameter defines the Diameter attributes, which are added to a request object in addition with *OriginHost* on page 8 and *OriginRealm* on page 8. The request object is created when a Diameter request message is received. The request object is then sent to the handler with the correct application AuthBy for this request.

The request object contains reference to the incoming Diameter request. The chosen Diameter application adds the reference to the Diameter answer. <AuthBy DiaRelay> relays the request to the correct peer and processes the answer, which is returned from the relay peer.

4.2.3. PreHandlerHook

This is an optional parameter, which defines the Perl function that is called before the request object is sent to the handlers. The only passed argument is the reference to the current request object.

4.2.4. NoReplyHook

This is an optional parameter, which defines the Perl function that is called if no reply is received from any Diameter peer.

4.2.5. NoreplyTimeout

This integer defines how soon, in seconds, *NoReplyHook* on page 7 is called if the request stored in proxy does not receive a reply. The default value is **5**.

4.2.6. ProductName

This is an optional parameter, which defines the name of the specific Diameter peer. If defined, it is sent to the other Diameter peers within the CER and CEA messages. The default value is **Radiator**.

4.2.7. OriginHost

This string defines the name that *<ServerDIAMETERTelco>* uses to identify itself to the Diameter peers. It is sent to the Diameter peers in the Diameter CER and CEA messages. The Diameter peers use *OriginHost* to determine whether they have connected to the correct peer. *OriginHost* must be specified.

4.2.8. OriginRealm

This string defines the name of the Realm the *<ServerDIAMETERTelco>* uses. It is sent to the Diameter peers in the CER and CEA messages. The peer uses it to determine which requests are routed to this Radiator instance. *OriginRealm* must be specified.

4.2.9. DestinationHost

This string defines the value for *Destination-Host* for Diameter requests. The usage of this parameter depends on the Diameter application that uses this *<DiaPeerDef>*. This is an optional parameter.

4.2.10. DestinationRealm

This string defines the value for *Destination-Realm* for Diameter requests. The usage of this parameter depends on the Diameter application that uses this *<DiaPeerDef>*. This is an optional parameter.

4.2.11. SupportedVendorIds

This is an optional parameter, which defines the supported vendor IDs announced in CER and CEA messages. This has no default value and the supported vendor ID is not announced by default. The default dictionary or the configured dictionary file consist an alias group *DictVendors* for all supported vendors.

Example

```
# Advertise Open System Consultants and 3GPP
SupportedVendorIds 9048, 3GPP
```

4.2.12. AuthApplicationIds

This is an optional parameter, which defines the *Auth-Application-Id* attributes announced in the CER and CEA messages. The *Auth-Application-Id* is not announced by default.

Example

```
# Advertise Diameter Credit Control and EAP applications
AuthApplicationIds 4, Diameter-EAP
```

4.2.13. AcctApplicationIds

This is an optional parameter, which defines the *Acct-Application-Id* attributes announced in the CER and CEA messages. The *Acct-Application-Id* is not announced by default.

Example

```
AcctApplicationIds Base Accounting
```

4.2.14. VendorAuthApplicationIds

This is an optional parameter, which defines the authentication *Vendor-Specific-Application-Id* attributes announced in the CER and CEA messages. The *Vendor-Specific-Application-Id* is not

announced by default. The parameter value is a comma-separated list of **`vendor:application`** values. Both names and direct numeric values are accepted.

Example

```
VendorAuthApplicationIds 3GPP:3GPP-Rx, 3GPP:3GPP-Gx
```

4.2.15. VendorAcctApplicationIds

This is an optional parameter, which defines the accounting *`Vendor-Specific-Application-Id`* attributes announced in the CER and CEA messages. The *`Vendor-Specific-Application-Id`* is not announced by default. The parameter value is a comma-separated list of **`vendor:application`** values. Both names and direct numeric values are accepted.

Example

```
VendorAcctApplicationIds OSC:Example accounting app
```

4.2.16. Initiator

This is an optional flag, which defines if the Radiator instance can act as a connection initiator. It is not set by default.

`Initiator` must be set if Radiator instance has to act as an initiator and create a connection to the Diameter peer defined by this <*`DiaPeerDef`*>. If *`Initiator`* is not set, the Radiator instance does not initiate connections but other instances, such as ePDG, must act as a initiator.

4.2.17. Peer

This parameter defines the name or IP address of the Diameter peer. Both IPv4 and IPv6 addresses are supported. This parameter is required when <*`DiaPeerDef`*> is configured to act as an initiator.

4.2.18. Port

This is an optional parameter, which defines the network port <*`ServerDIAMETERTelco`*> listens to for connections from Diameter peers. For more information, see Radiator reference manual [https://files.radiatorsoftware.com/radiator/ref.pdf] under section <ServerDIAMETER>.

4.2.19. SCTPPeer

This parameter specifies one host name or address of an SCTP peer to connect to. An address can be an IPv4 or IPv6 address. Multiple *`SCTPPeer`* parameters are supported. When *`SCTPPeer`* is defined, it is used instead of *`Host`* or *`Peer`* parameters. Special formatting characters are supported. If SCTP multihoming is not supported, connection is attempted to each peer at a time.

When SCTP multihoming is supported, connection is attempted to all peers at once. In this case, all addresses defined with *`SCTPPeer`* must be either IPv4 or IPv6 addresses

Here is an example of using *`SCTPPeer`*:

```
# Peer has multiple IPv6 addresses
SCTPPeer 2001:db8:1500:1::a100
SCTPPeer 2001:db8:1500:2::a100
```

4.2.20. LocalAddress and LocalPort

These parameters control the address and optionally the port number used for the client source port, although this is usually not necessary. *`LocalPort`* is a string, it can be a port number or name. It binds the local port if *`LocalAddress`* is defined. If *`LocalPort`* is not specified or if it is set to **`0`**, a port number is allocated in the usual way.

When SCTP multihoming is supported, multiple comma separated addresses can be configured. All addresses defined with *`LocalAddress`* must be either IPv4 or IPv6 addresses.

```
LocalAddress 203.63.154.29
LocalPort 12345
```

4.2.21. Protocol

This is an optional parameter, which allows choosing transport layer protocol, TCP or SCTP, for carrying Diameter messages. For more information, see Radiator reference manual [https://files.radiatorsoftware.com/radiator/ref.pdf] under section <ServerDIAMETER>.

4.2.22. DisconnectTraceLevel

This optional parameter specifies log trace level for peer initiated disconnects. The default value is error level **`0`**. When connections are known to be short-lived, a non-default value may be useful. This parameter is available for all Stream based modules, such as *`<ServerDIAMETER>`* and *`<AuthBy RADSEC>`*.

```
# Debug logging is enough for peer disconnects
DisconnectTraceLevel 4
```

4.2.23. TLS_*

These parameters enable and configure of TLS (Transport Layer Security) authentication and encryption. For more information, see Radiator reference manual [https://files.radiatorsoftware.com/radiator/ref.pdf] under section "TLS configuration". To enable TLS, you need to define *`TLS_Protocols`* configuration parameter with the other TLS related parameters, such as certificates, that depend on your operating environment.

Note

Old configuration parameters *`UseTLS`* and *`UseSSL`* are obsolete and should not be used. Use *`TLS_Protocols`* instead.

4.3. <3GPPAuthMAP>

This section describes the configuring parameters of *`<3GPPAuthMAP>`*.

4.3.1. MAP

This string identifies the MAP (Mobile Application Part) used by a certain AuthBy.

4.4. <AAAServerSWx>

`<AAAServerSWx>` does not have any configurable parameters at the moment except *`Identifier`*.

4.5. <AAAServerSWm>

<AAAServerSWm> does not have any configurable parameters at the moment except *Identifier*.

4.6. <AAAServerS6b>

<AAAServerS6b> does not have any configurable parameters at the moment except *Identifier*.

4.7. <EAPContextInternal>

This section describes the configuring parameters of *<EAPContextInternal>*.

4.7.1. EAPContextTimeout

This integer defines the maximum time period, in seconds, how long EAP (Extensible Authentication Protocol) context is retained. The default value is **3**. Usually there is no need to change this value.

4.8. <EAPContextGossip>

This section describes the configuring parameters of *<EAPContextGossip>*.

4.8.1. EAPContextTimeout

This integer defines the maximum time period, in seconds, how long EAP context is retained. The default value is **3**. Usually there is no need to change this value.

4.9. <AAASessionInternal>

<AAASessionInternal> does not have any configurable parameters at the moment except *Identifier*. It keeps the session information of active SWm and S6b sessions and profiles fetched from HSS. The information is stored in internal memory.

4.10. <AAASessionGossip>

<AAASessionGossip> keeps the session information of active SWm and S6b sessions and profiles fetched from HSS. The information is stored in Gossip. The Gossip framework is documented in Radiator reference manual [https://files.radiatorsoftware.com/radiator/ref.pdf] under section <GossipRedis> and Gossip framework. *<AAASessionGossip>* supports also *Identifier*.

4.10.1. CloseAction

CloseAction defines how to update Gossip when the session is closed. This is not set by default and the session is deleted when closed. The functionality is similar as when the value is set to **delete**. When set to **timestamp**, the session is not deleted but the stopping time timestamp is marked when the session is closed.

4.11. <AAASessionSQL>

This section describes the configuring parameters of *<AAASessionSQL>*. It keeps the session information of active SWm and S6b sessions and profiles fetched from HSS. The information is stored in SQL database.

4.11.1. AddSessionQuery

This string contains the SQL query for saving the SWm and S6b session information. Most of the parameters are required to save. Some parameters are conditional depending on the features the non-3GPP access system requires.

The following bind variables are available in *AddSessionQuery*:

- *%0*

  This is the IMSI.

- *%1*

  This is the value of Diameter *Session-Id* AVP (Attribute-Value Pair).

- *%2*

  This is the value of Diameter *Origin-Host* AVP.

- *%3*

  This is the value of Diameter *Origin-Realm* AVP.

- *%4*

  This is the Diameter *Application Id* value.

- *%5*

  This is the Diameter application name, which corresponds to the *Application Id*.

- *%6*

  This is the Diameter *Service-Selection* attribute value, for example, SSID (Service Set Identifier) or NAI (Network Access Identifier).

- *%7*

  This is the permanent user identity represented as NAI without leading digit in front of IMSI.

- *%8*

  This is the session start time.

- *%9*

  Conditional: This is the value of *Emergency-Services* attribute. There is no need to store this attribute if emergency services are not enabled.

- *%10*

  Conditional: This is the value of *User-Name* attribute. Storing this is only needed when IMSI privacy is enabled and the ePDG requires anonymous username instead of permanent user identity.

For more information about SQL bind variables, see Radiator reference manual [https://files.radiatorsoftware.com/radiator/ref.pdf] under section SQL Bind Variables.

Example: *AddSessionQuery*

In the following example query, bind variables marked with question marks are used with *AddSessionQueryParam*.

```
AddSessionQuery INSERT INTO sessions ( \
      imsi, session_id, origin_host, origin_realm, \
      application_id, application_name, \
      service_selection, permanent_user_id, start_time) \
      VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)
AddSessionQueryParam %0
AddSessionQueryParam %1
AddSessionQueryParam %2
```

```
AddSessionQueryParam %3
AddSessionQueryParam %4
AddSessionQueryParam %5
AddSessionQueryParam %6
AddSessionQueryParam %7
AddSessionQueryParam %8
```

The following example shows how to store additional information required by emergency services and IMSI encryption. It's also possible to enable just emergency services or IMSI encryption. These two features do not depend on each other.

```
AddSessionQuery INSERT INTO sessions ( \
      imsi, session_id, origin_host, origin_realm, \
      application_id, application_name, \
      service_selection, permanent_user_id, start_time, \
      emergency_services, user_name ) \
      VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)
AddSessionQueryParam %0
AddSessionQueryParam %1
AddSessionQueryParam %2
AddSessionQueryParam %3
AddSessionQueryParam %4
AddSessionQueryParam %5
AddSessionQueryParam %6
AddSessionQueryParam %7
AddSessionQueryParam %8
AddSessionQueryParam %9
AddSessionQueryParam %10
```

4.11.2. AddSessionQueryParam

This string array defines the bound variables to be used with *`AddSessionQuery`*. See *`AddProfileQuery`* on page 11 for more information about the available bind variables.

4.11.3. GetSessionQuery

This string contains the SQL query for getting information about a single session for a specific IMSI.

The following bind variable is available in *`GetSessionQuery`*:

- *`%0`*

  This is the value of Diameter *`Session-Id`* AVP.

For more information about SQL bind variables, see Radiator reference manual [https://files.radiatorsoftware.com/radiator/ref.pdf] under section SQL Bind Variables.

Example: *`GetSessionQuery`*

This example shows the values that must be fetched from the database. Bind variables marked with question marks are used with *`GetSessionQueryParam`*, and *`GetSessionColumnDef`* parameters map SQL result columns to session variables. Session variable names are fixed as shown below, and they must correspond to AddSessionQuery example on page 11.

```
GetSessionQuery SELECT id, start_time, imsi, \
      session_id, origin_host, origin_realm, \
```

```
      application_id, application_name, \
      service_selection, permanent_user_id \
      FROM sessions WHERE stop_time IS NULL AND session_id=?
GetSessionQueryParam %0
GetSessionColumnDef  0, id
GetSessionColumnDef  1, start_time
GetSessionColumnDef  2, imsi
GetSessionColumnDef  3, session_id
GetSessionColumnDef  4, origin_host
GetSessionColumnDef  5, origin_realm
GetSessionColumnDef  6, application_id
GetSessionColumnDef  7, application_name
GetSessionColumnDef  8, service_selection
GetSessionColumnDef  9, permanent_user_id
```

The following example shows two additional columns, **`emergency_services`** and **`user_name`**, required by emergency services and IMSI encryption.

```
GetSessionQuery SELECT id, start_time, imsi, \
      session_id, origin_host, origin_realm, \
      application_id, application_name, \
      service_selection, permanent_user_id, \
      emergency_services, user_name \
      FROM sessions WHERE stop_time IS NULL AND session_id=?
GetSessionQueryParam %0
GetSessionColumnDef  0, id
GetSessionColumnDef  1, start_time
GetSessionColumnDef  2, imsi
GetSessionColumnDef  3, session_id
GetSessionColumnDef  4, origin_host
GetSessionColumnDef  5, origin_realm
GetSessionColumnDef  6, application_id
GetSessionColumnDef  7, application_name
GetSessionColumnDef  8, service_selection
GetSessionColumnDef  9, permanent_user_id
GetSessionColumnDef 11, emergency_services
GetSessionColumnDef 11, user_name
```

4.11.4. GetSessionQueryParam

This string array defines the bind variables to be used with *`GetSessionQuery`*. See *`GetSessionQuery`* on page 13 for more information about the available bind variables.

4.11.5. GetSessionColumnDef

This string hash defines how Radiator interprets the result of the *`GetSessionQuery`* statement. The format is **`'GetSessionColumnDef n, item'`**, where **`n`** is the index of the column in the *`GetSessionQuery`* on page 13 or *`GetAllSessionsQuery`* on page 16 result and **`item`** is the name of the value used in later processing. See *`GetSessionQuery`* on page 13 for an example.

4.11.6. CloseSessionQuery

This string contains the SQL query for closing a session.

The following bind variable is available in *CloseSessionQuery*:

- *%0*

  This is the ID which is fetched with *GetSessionSelect*.

For more information about SQL bind variables, see Radiator reference manual [https://files.radiatorsoftware.com/radiator/ref.pdf] under section SQL Bind Variables.

Example: *CloseSessionQuery*

In the following example query, bind variables marked with question marks are used with *CloseSessionQueryParam*s listed below the query.

```
CloseSessionQuery UPDATE sessions SET stop_time=? WHERE id=?
CloseSessionQueryParam %t
CloseSessionQueryParam %0
```

Instead of the updating session stop timestamp, it's also possible to remove old sessions.

```
CloseSessionQuery DELETE FROM sessions WHERE id=?
CloseSessionQueryParam %0
```

4.11.7. CloseSessionQueryParam

This string array defines the bound variables to be used with *CloseSessionQuery*. See *CloseSessionQuery* on page 14 for more information about the available bind variables.

4.11.8. CloseAllSessionsQuery

This string contains the SQL query for closing all open sessions of a specific IMSI.

The following bind variable is available in *CloseAllSessionsQuery*:

- *%0*

  This is the IMSI.

For more information about SQL bind variables, see Radiator reference manual [https://files.radiatorsoftware.com/radiator/ref.pdf] under section SQL Bind Variables.

Example: *CloseAllSessionsQuery*

In the following example query, bind variables marked with question marks are used with *CloseAllSessionsQueryParam*s listed below the query.

```
CloseAllSessionsQuery UPDATE sessions SET \
      stop_time=? WHERE imsi=? \
      AND stop_time IS NULL
CloseAllSessionsQueryParam %t
CloseAllSessionsQueryParam %0
```

Instead of the updating session stop timestamp, it's also possible to remove the old sessions.

```
CloseAllSessionsQuery DELETE FROM sessions WHERE imsi=? AND stop_time IS NULL
CloseAllSessionsQueryParam %0
```

4.11.9. CloseAllSessionsQueryParam

This string array defines the bound variables to be used with *CloseAllSessionsQuery*. See *CloseAllSessionsQuery* on page 15 for more information about the available bind variables.

4.11.10. CountSessionsQuery

This string contains the SQL query for counting all active sessions for one IMSI. The query must return one row where the first column is the session count.

The following bind variable is available in *CountSessionsQuery*:

- *%0*

  This is the IMSI.

For more information about SQL bind variables, see Radiator reference manual [https://files.radiatorsoftware.com/radiator/ref.pdf] under section SQL Bind Variables.

Example: *CountSessionsQuery*

In the following example query, bind variables marked with question marks are used with *CountSessionsQueryParam* listed below the query.

```
CountSessionsQuery SELECT COUNT(id) FROM sessions \
      WHERE stop_time IS NULL AND imsi=?
CountSessionsQueryParam %0
```

4.11.11. CountSessionsQueryParam

This string array defines the bind variables to be used with *CountSessionsQuery*. See *CountSessionsQuery* on page 16 for more information about the available bind variables.

4.11.12. GetAllSessionsQuery

This string contains the SQL query for getting information of all active sessions for a specific IMSI.

The following bind variable is available in *GetAllSessionsQuery*:

- *%0*

  This is the IMSI.

For more information about SQL bind variables, see Radiator reference manual [https://files.radiatorsoftware.com/radiator/ref.pdf] under section SQL Bind Variables.

Example: *GetAllSessionsQuery*

GetAllSessionsQuery is similar to GetSessionsQuery. The main difference is that the query must return all active sessions for an IMSI instead of just single session information. Session variable names are fixed as shown below, and they must correspond to AddSessionQuery example on page 11.

This example shows the values that must be fetched from the database.

```
GetAllSessionsQuery SELECT id, start_time, imsi, \
      session_id, origin_host, origin_realm, \
      application_id, application_name, \
      service_selection, permanent_user_id \
      FROM sessions WHERE stop_time IS NULL AND imsi=?
```

```
GetAllSessionsQueryParam %0
GetSessionColumnDef  0, id
GetSessionColumnDef  1, start_time
GetSessionColumnDef  2, imsi
GetSessionColumnDef  3, session_id
GetSessionColumnDef  4, origin_host
GetSessionColumnDef  5, origin_realm
GetSessionColumnDef  6, application_id
GetSessionColumnDef  7, application_name
GetSessionColumnDef  8, service_selection
GetSessionColumnDef  9, permanent_user_id
```

The following example shows two additional columns, **`emergency_services`** and **`user_name`**, required by emergency services and IMSI encryption.

```
GetAllSessionsQuery SELECT id, start_time, imsi, \
      session_id, origin_host, origin_realm, \
      application_id, application_name, \
      service_selection, permanent_user_id, \
      emergency_services, user_name \
      FROM sessions WHERE stop_time IS NULL AND imsi=?
GetSessionQueryParam %0
GetSessionColumnDef  0, id
GetSessionColumnDef  1, start_time
GetSessionColumnDef  2, imsi
GetSessionColumnDef  3, session_id
GetSessionColumnDef  4, origin_host
GetSessionColumnDef  5, origin_realm
GetSessionColumnDef  6, application_id
GetSessionColumnDef  7, application_name
GetSessionColumnDef  8, service_selection
GetSessionColumnDef  9, permanent_user_id
GetSessionColumnDef 10, emergency_services
GetSessionColumnDef 11, user_name
```

4.11.13. GetAllSessionsQueryParam

This string array defines the bind variables to be used with *`GetAllSessionsQuery`*. See *`GetAllSessionsQuery` on page 16* for more information about the available bind variables.

4.11.14. SaveProfileQuery

This string contains the SQL query for saving the subscriber's profile.

The following bind variables are available in *`SaveProfileQuery`*:

- *`%0`*

  This is the IMSI.

- *`%1`*

  This is the profile received from the HSS over SWx.

For more information about SQL bind variables, see Radiator reference manual [https://files.radiatorsoftware.com/radiator/ref.pdf] under section SQL Bind Variables.

Example: *SaveProfileQuery*

In the following example query, bind variables marked with question marks are used with *SaveProfileQueryParam*s listed below the query.

```
SaveProfileQuery INSERT INTO profiles (imsi, insert_time, profile) VALUES (?,?,?)
SaveProfileQueryParam %0
SaveProfileQueryParam %t
SaveProfileQueryParam %1
```

4.11.15. SaveProfileQueryParam

This string array defines the bind variables to be used with *SaveProfileQuery*. See *SaveProfileQuery* on page 17 for more information about the available bind variables.

4.11.16. GetProfileQuery

This string contains the SQL query for fetching the subscriber's profile. The query returns one row where the first column is the session count.

The following bind variable is available in *GetProfileQuery*:

- %0

  This is the IMSI.

For more information about SQL bind variables, see Radiator reference manual [https://files.radiatorsoftware.com/radiator/ref.pdf] under section SQL Bind Variables.

Example: *GetProfileQuery*

In the following example query, bind variables marked with question marks are used with *GetProfileQueryParam* listed below the query.

```
GetProfileQuery SELECT profile FROM profiles WHERE imsi=?
GetProfileQueryParam %0
```

4.11.17. GetProfileQueryParam

This string array defines the bind variables to be used with *GetProfileQuery*. See *GetProfileQuery* on page 18 for more information about the available bind variables.

4.11.18. DeleteProfileQuery

This string contains the SQL query for deleting subscriber's profile.

The following bind variables are available in *DeleteProfileQuery*:

- *%0*

  This is the IMSI.

For more information about SQL bind variables, see Radiator reference manual [https://files.radiatorsoftware.com/radiator/ref.pdf] under section SQL Bind Variables.

Example: *DeleteProfileQuery*

In the following example query, bind variables marked with question marks are used with *DeleteProfileQueryParam* listed below the query.

```
DeleteProfileQuery DELETE FROM profiles WHERE imsi=?
DeleteProfileQueryParam %0
```

4.11.19. DeleteProfileQueryParam

This string array defines the bind variables to be used with *DeleteProfileQuery* .See *DeleteProfileQuery* on page 18 for more information about the available bind variables.

4.12. <AuthBy Dia3GPPAAAServer>

This section describes the configuring parameters of *<AuthBy Dia3GPPAAAServer>*. Apart from the parameters listed here, *<AuthBy Dia3GPPAAAServer>* inherits other parameters from *<AuthBy AKA>*. These parameters are documented in Radiator SIM Module reference manual.

4.12.1. AAAServerS6b

This object list defines the AAA Server S6b clause to be used.

4.12.2. AAAServerSWm

This object list defines the AAA Server SWm clause to be used.

4.12.3. AAAServerSWx

This object list defines the AAA Server SWx clause to be used.

4.12.4. AAASession

This object list defines the identifier of AKA Identity clause to be used as the 3GPP AAA Server session database.

4.12.5. AKAIdentity

This string defines the identifier of AKA Identity clause for mapping temporary AKA IDs (TMSI (Temporary Mobile Subscriber Identity) and reauthentication ID) to IMSI.

4.12.6. EAPContext

This object list defines the identifier of EAP context clause to be used.

4.12.7. SWmAuth

This string defines the identifier of 3GPP Auth clause for communicating with the remote AKA authentication and authorisation peer (HSS or MAP) for SWm messages.

4.12.8. EmergencyServices

This flag parameter enables support for emergency services. When *EmergencyServcies* is not enabled, SWm or S6b requests with *Emergency-Services* AVP that have *Emergency-Indication* bit set are logged and rejected. Defaults to not set.

4.12.9. IMSICrypt

This string defines the identifier of *IMSICrypt* clause to use for IMSI decryption. For more about IMSI encryption, see Section 4.13. <IMSICrypt> on page 20.

4.12.10. DiaEIR

This object list enables EIR (Equipment Identity Register) check and identifies the used *DiaEIR* clause. This is an optional parameter.

4.12.11. EIR_SWm_UnknownAction

This defines how the 3GPP AAA Server handles the SWm requests if the EIR check does not recognise the connecting equipment. Unrecognised equipment is accepted by default. Allowed values are **accept** and **reject**.

4.12.12. StripMACFromUserName

This optional string parameters defines how 3GPP AAA Server strips MAC address of wireless LAN access point if it is embedded in the username. Some user equipment sends username in format such as `0234031234567890@00-11-22-33-44-55:nai.epc.mnc003.mcc234.3gppnetwork.org` where `00-11-22-33-44-55:` is not part of the username as specified by 3GPP TS 29.273. If ePDG or other network device can not be configured to remove the MAC address, 3GPP AAA Server can be configured to do so.

This option is not set by default and nothing is stripped. The only currently allowed value is **colon**, which removes everything starting after @ and ending with the first :.

```
# Our ePDG can not strip MAC addresses
StripMACFromUserName colon
```

4.13. <IMSICrypt>

This section describes the configuration parameters of an *<IMSICrypt>* clause. This clause provides support for Permanent Identity encryption, sometimes also called IMSI encryption or IMSI privacy. IMSI encryption is specified in 3GPP document *S3-170116* and Wireless Broadband Alliance technical specification *IMSI Privacy Protection for Wi-Fi*.

IMSI encryption is supported by all EAP-SIM, EAP-AKA, EAP-AKA' and 3GPP AAA Server configuration clauses. To enable IMSI encryption, you need to modify Radiator configuration as follows:

- First define an *<IMSICrypt>* clause with an *Identifier* parameter.
- Then add *IMSICrypt* configuration parameter in *AuthBy* clauses.

For required software versions and modules, see Section 2.1. Prerequisites on page 1. A full configuration example is in file `goodies/imsicrypt.cfg`

Example: *IMSICrypt*

For a full example, see `goodies/simcrypt.cfg`. Key configured with *DefaultPrivateKey* is used when **Key Identifier AVP**, also known as **certificate identifier attribute**, is not present. Key configured with *PrivateKeyFile* is used identifier is not present.

For more information about **Key Identifier AVP** and **certificate identifier attribute**, see 3GPP document *S3-170116* and Wireless Broadband Alliance technical specification *IMSI Privacy Protection for Wi-Fi*.

```
<IMSICrypt>
   # Identifier is used by AKA and SIM clauses to refer to this
   # clause for identity decryption.
   Identifier imsi-decrypter
```

```
   # DefaultPrivateKeyFile and DefaultPrivateKeyPassword work as
   # pairs.
   DefaultPrivateKeyFile %D/certificates/server-key.pem
   DefaultPrivateKeyPassword whatever

   #DefaultPrivateKeyFile %D/private-keys/default-key1.pem
   #DefaultPrivateKeyPassword password-for-default-key1

   #DefaultPrivateKeyFile %D/private-keys/default-key2.pem
   ## Key in file default-key2.pem is not password protected

   #PrivateKeyFile     CertificateSerialNumber=12345,%D/private-keys/key-12345.pem
   #PrivateKeyPassword CertificateSerialNumber=12345,password-for-key-12345

   #PrivateKeyFile     CertificateSerialNumber=23456,%D/private-keys/key-23456.pem
   ## Key in file key-23456.pem is not password proteced

   #PrivateKeyFile     CertificateSerialNumber=34567,%D/private-keys/key-34567.pem
   #PrivateKeyPassword CertificateSerialNumber,password-for-key-34567
</IMSICrypt>

<AuthBy AKAWX>
   # Other AKAWX configuration parameters
   IMSICrypt imsi-decrypter
</AuthBy>
```

4.13.2. DefaultPrivateKeyFile

DefaultPrivateKeyFile defines a private key file name for a key that is used when an encrypted permanent identity does not have key identifier. You can configure multiple key files to support key roll over. Decryption is attempted with all key files until the first one succeeds. If no key is able to correctly decrypt an encrypted identity, an error is returned to the client and the authentication fails.

See the configuration example on page 20 for more information.

4.13.3. DefaultPrivateKeyPassword

DefaultPrivateKeyPassword defines the password for decrypting a default private key defined with *DefaultPrivateKey*. Key encryption is optional. If a key is stored without encryption, this parameter is not needed. An encrypted key file and its respective password must be configured in pairs.

See the configuration example on page 20 for more information.

4.13.4. PrivateKeyFile

PrivateKeyFile defines a private key file name in **name=value,filename** format. This key is used when an encrypted permanent identity sent by the client has a key identifier. Decryption is attempted only with the key that matches the key identifier the client sends. If the key is not able to correctly decrypt the encrypted identity, an error is returned to the client and the authentication fails. You should not configure more than one *PrivateKeyFile* parameter with the same **name=value** because only the latest parameter is used.

See the configuration example on page 20 for more information.

4.13.5. PrivateKeyPassword

PrivateKeyPassword defines the password for decrypting a private key defined with *PrivateKey*. The format for this parameter is **name=value,password** where **name** and **value** must match the respective values of a *PrivateKeyFile* parameter. Key encryption is optional. If a key is stored without encryption, this parameter is not needed.

See the configuration example on page 20 for more information.

4.14. Configuring EIR

This section describes how to configure EIR parameters.

The EIR is a database that contains information on mobile devices that are banned from using the network or need to be tracked for some purpose. The devices are listed by their IMEI (International Mobile Equipment Identity).

You can find example configuration files in the Carrier Pack contents, `goodies/eir-client.cfg` and `goodies/eir-server.cfg`.

4.14.1. <DiaEIR>

This section describes the configuring parameters of <*DiaEIR*>. <*DiaEIR*> implements the interface for querying EIR.

4.14.1.1. Identifier

This parameter defines the name of the specific EIR clause in the configuration. This must be defined, otherwise you cannot refer to this EIR clause.

4.14.1.2. DiaPeerDef

This parameter defines the Diameter Peer which the this clause connects to.

4.14.1.3. EIRCache

EIRCache is Identifier of the EIRCache clause. If this is not set, no caching is done. This is not set by default.

4.14.2. <EIRCacheInternal>

This section describes the configuring parameters of <*EIRCacheInternal*>. <*EIRCacheInternal*> is an optional module for caching EIR responses.

4.14.2.1. Identifier

This parameter defines the name of the specific EIR clause in the configuration. This must be defined, otherwise you cannot refer to this EIR clause.

4.14.2.2. CacheTimeout

CacheTimeout defines (in seconds) for how long the successful EIR responses are cached. The default value is **1800** (30 minutes).

4.14.2.3. NegativeCacheTimeout

If EIR cannot be connected or it returns an answer that cannot be successfully processed, *NegativeCacheTimeout* defines the time (in seconds) for how long time the answer is cached. Using this feature gives EIR time to recover from the possible error condition. The default value is **300** (5 minutes).

4.15. <Server3GPPTest>

This section describes the configuring parameters of *<Server3GPPTest>*.

4.15.1. 3GPPCardDatabaseFilename

This string defines the file path and name where the 2G SIM (Subscriber Identity Module) or 3G USIM card details are stored. Radiator requires read and write access to this file. When defined, this parameter is used to find the Milenage algorithm parameters for SIM and USIM cards. See `goodies/simcards.dat` for a sample file.

The file contains the following information coded to hexadecimal:

- IMSI for the SIM/USIM card
- Ki (Authentication key)
- Encrypted OPc (Operator Code)
- AMF (Authentication Management Field)
- SQN (Sequence Number)

4.15.2. IndLength

This integer defines the length of the IND part in bits in the AKA authentication vector SQN. The default value is **5**.

4.15.3. VendorAuthApplicationIds

This string defines the values of *Auth-Application ID* AVPs in CER. This is an optional parameter with no default value.

4.16. <ServerDIAMETERTelco>

This section describes the configuring parameters of *<ServerDIAMETERTelco>*.

4.16.1. Peer

This parameter defines the name or IP address of the Diameter peer. Both IPv4 and IPv6 addresses are supported. This parameter is required when *<DiaPeerDef>* is configured to act as an initiator.

4.16.2. Port

This is an optional parameter, which defines the network port *<ServerDIAMETERTelco>* listens to for connections from Diameter peers. For more information, see Radiator reference manual [https://files.radiatorsoftware.com/radiator/ref.pdf] under section <ServerDIAMETER>.

4.16.3. BindAddress

This is an optional parameter, which defines one or more network interface addresses that are listened to for incoming Diameter connections. For more information, see Radiator reference manual [https://www.open.com.au/radiator/ref.pdf] under section <ServerDIAMETER>.

4.16.4. MaxBufferSize

This is an optional parameter, which defines the maximum number of octets buffered in output. For more information, see Radiator reference manual [https://www.open.com.au/radiator/ref.pdf] under section <ServerDIAMETER>.

4.16.5. Protocol

This is an optional parameter, which allows choosing transport layer protocol, TCP or SCTP, for carrying Diameter messages. For more information, see Radiator reference manual [https://files.radiatorsoftware.com/radiator/ref.pdf] under section <ServerDIAMETER>.

4.16.6. ReadTimeOut

This is an optional parameter, which defines the maximum time, in seconds, to wait for incoming Diameter connection to complete the initial handshaking. The default value is **`10`**. For more information, see Radiator reference manual [https://www.open.com.au/radiator/ref.pdf] under section <ServerDIAMETER>.

4.16.7. DisconnectTraceLevel

This optional parameter specifies log trace level for peer initiated disconnects. The default value is error level **`0`**. When connections are known to be short-lived, a non-default value may be useful. This parameter is available for all Stream based modules, such as *`<ServerDIAMETER>`* and *`<AuthBy RADSEC>`*.

```
# Debug logging is enough for peer disconnects
DisconnectTraceLevel 4
```

4.16.8. TLS_*

These parameters enable and configure of TLS authentication and encryption. For more information, see Radiator reference manual [https://files.radiatorsoftware.com/radiator/ref.pdf] under section "TLS configuration". To enable TLS, you need to define *`TLS_Protocols`* configuration parameter with the other TLS related parameters, such as certificates, that depend on your operating environment.

Note

Old configuration parameters *`UseTLS`* and *`UseSSL`* are obsolete and should not be used. Use *`TLS_Protocols`* instead.

---

5. Abbreviations

Authentication, Authorisation, Accounting

AAA (Authentication, Authorisation, Accounting)

Acronym: **AAA**

AA Request

AAR (AA Request)

Acronym: **AAR**

Authentication and Key Agreement

AKA (Authentication and Key Agreement)

Acronym: **AKA**

Authentication Management Field

AMF (Authentication Management Field)

Acronym: **AMF**

Abort-Session-Request

ASR (Abort-Session-Request)

Acronym: **ASR**

Attribute-Value Pair

AVP (Attribute-Value Pair)

Acronym: **AVP**

Capabilities Exchange Answer

CEA (Capabilities Exchange Answer)

Acronym: **CEA**

Capabilities Exchange Request

CER (Capabilities Exchange Request)

Acronym: **CER**

Diameter EAP Request

DER (Diameter EAP Request)

Acronym: **DER**

Extensible Authentication Protocol

EAP (Extensible Authentication Protocol)

Acronym: **EAP**

Extensible Authentication Protocol - Authentication and Key Agreement

EAP-AKA (Extensible Authentication Protocol - Authentication and Key Agreement)

Acronym: **EAP-AKA**

Extensible Authentication Protocol - Authentication and Key Agreement Prime

EAP-AKA' (Extensible Authentication Protocol - Authentication and Key Agreement Prime)

Acronym: **EAP-AKA'**

Extensible Authentication Protocol - Subscriber Identity Module

EAP-SIM (Extensible Authentication Protocol - Subscriber Identity Module)

Acronym: **EAP-SIM**

Equipment Identity Register

EIR (Equipment Identity Register)

Acronym: **EIR**

Evolved Packet Data Gateway

ePDG (Evolved Packet Data Gateway)

Acronym: **ePDG**

Home Location Register

HLR (Home Location Register)

Acronym: **HLR**

Home Subscriber Server

HSS (Home Subscriber Server)

Acronym: **HSS**

International Mobile Equipment Identity

IMEI (International Mobile Equipment Identity)

Acronym: **IMEI**

International mobile subscriber identity

IMSI (International mobile subscriber identity)

Acronym: **IMSI**

Authentication key

Ki (Authentication key)

Acronym: **Ki**

Mobile Application Part

MAP (Mobile Application Part)

Acronym: **MAP**

Multimedia-Auth-Request

MAR (Multimedia-Auth-Request)

Acronym: **MAR**

Network Access Identifier

NAI (Network Access Identifier)

Acronym: **NAI**

Operator Code

OPc (Operator Code)

Acronym: **OPc**

Packet Data Network Gateway

PDN GW (Packet Data Network Gateway)

Acronym: **PDN GW**

Note

Sometimes acronym PGW is used.

Push-Profile-Request

PPR (Push-Profile-Request)

Acronym: **PPR**

Reauthentication Request

RAR (Reauthentication Request)

Acronym: **RAR**

Registration-Termination-Request

RTR (Registration-Termination-Request)

Acronym: **RTR**

Server-Assignment-Request

SAR (Server-Assignment-Request)

Acronym: **SAR**

Subscriber Identity Module

SIM (Subscriber Identity Module)

Acronym: **SIM**

Sequence Number

SQN (Sequence Number)

Acronym: **SQN**

Service Set Identifier

SSID (Service Set Identifier)

Acronym: **SSID**

Session-Termination-Request

STR (Session-Termination-Request)

Acronym: **STR**

Transport Layer Security

TLS (Transport Layer Security)

Acronym: **TLS**

Temporary Mobile Subscriber Identity

TMSI (Temporary Mobile Subscriber Identity)

Acronym: **TMSI**

Universal Subscriber Identity Module

USIM (Universal Subscriber Identity Module)

Acronym: **USIM**