Radiator® Service Provider Module

Installation and reference manual for Radiator® Service Provider Module 1.8. Last revised on November 3, 2022

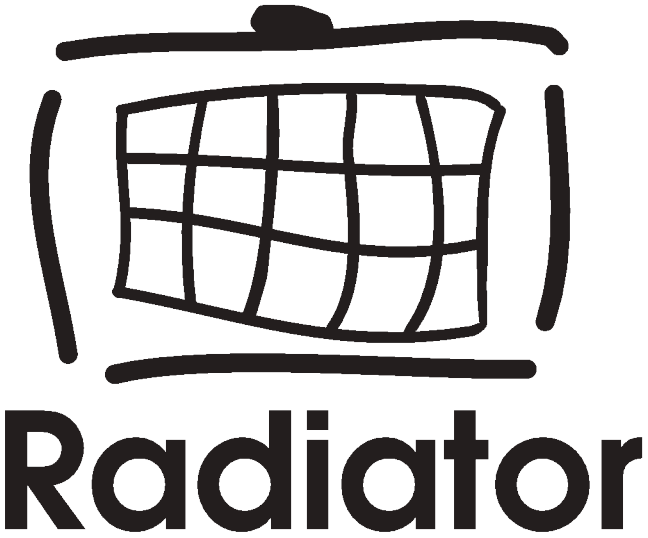

Table of Contents

1. Introduction to Radiator Service Provider Module 1
2. Installing Radiator Service Provider Module 1
    2.1. Prerequisites 1
    2.2. Installing and upgrading Radiator Service Provider Module 1
3. Configuring Diameter peers 2
    3.1. <DiaPeerDef attribute=value,attribute=value,...> 2
        3.1.1. Identifier 3
        3.1.2. Initiator 3
        3.1.3. AddToRequestFromDia 4
        3.1.4. PreHandlerHook 4
        3.1.5. NoReplyHook 4
        3.1.6. NoreplyTimeout 4
        3.1.7. ProductName 4
        3.1.8. OriginHost 4
        3.1.9. OriginRealm 4
        3.1.10. DestinationHost 4
        3.1.11. DestinationRealm 4
        3.1.12. SupportedVendorIds 5
        3.1.13. AcctApplicationIds 5
        3.1.14. VendorAuthApplicationIds 5
        3.1.15. VendorAcctApplicationIds 5
        3.1.16. Peer 5
        3.1.17. Port 5
        3.1.18. LocalAddress and LocalPort 6
        3.1.19. Protocol 6
        3.1.20. TLS_* 6
4. Configuring DIAMETER server 6
    4.1. <ServerDIAMETERTelco> 6
        4.1.1. Peer 6
        4.1.2. Port 6
        4.1.3. BindAddress 7
        4.1.4. MaxBufferSize 7
        4.1.5. Protocol 7
        4.1.6. ReadTimeOut 7
        4.1.7. TLS_* 7
5. Configuring Diameter relay 7
    5.1. <AuthBy DiaRelay> 7
        5.1.1. DiaPeerDef 7
        5.1.2. RelayAlgorithm 8
6. Configuring <MessageLog FILECARRIER> 8

6.1. Filename 8
6.2. Format 9
7. Configuring EIR 9
7.1. <AuthBy DiaEIR> 9
7.1.1. DiaEIR 9
7.1.2. MakeAnswer 9
7.1.3. EIR_UnknownAction 9
7.1.4. EIR_AttributesHook 9
7.1.5. IMEIAttribute 10
7.1.6. SoftwareVersionAttribute 10
7.1.7. IMSIAttribute 10
7.2. <DiaEIR> 10
7.2.1. Identifier 10
7.2.2. DiaPeerDef 10
7.2.3. EIRCache 10
7.3. <EIRCacheInternal> 10
7.3.1. Identifier 10
7.3.2. CacheTimeout 10
7.3.3. NegativeCacheTimeout 10
7.4. <ServerDiaEIR> 11
7.4.1. VendorAuthApplicationIds 11
8. Configuring <ServerDHCP> 11
8.1. UsernameOption 11
8.2. DefaultUsername 11
8.3. DHCPyiaddrAttr 11
8.4. DHCPSubnetMaskAttr 11
8.5. DefaultDHCPSubnetMask 11
8.6. DHCPRouterAttr 12
8.7. DHCPDomainServerAttr 12
8.8. DHCPAddressTimeAttr 12
8.9. DefaultDHCPAddressTime 12
8.10. DHCPClientIdentifier 12
9. <AuthBy DiaINTERNAL> 12
9.1. DefaultResult 12
9.2. AuthResult 12
9.3. AcctResult 13
10. Using VSA framework for customised attributes 13
11. Abbreviations 14

1. Introduction to Radiator Service Provider Module

This document describes how to install and configure Radiator Service Provider Module. This module was previously known as Radiator Carrier Module.

Radiator Service Provider Module is designed especially for service provider and carrier use. It includes the following functionalities:

- Radiator AAA Server Software, for more information, see Product specifications [https://radiatorsoftware.com/products/radiator/]
- Advanced Radiator Diameter Client and Server Software with `<DiaPeerDef>` and `<ServerDIAMETERTelco>`
- Diameter relay application with `<AuthBy DiaRelay>`
- General support for EIR (Equipment Identity Register), EIR client with `<AuthBy DiaEIR>` and test EIR server with `<ServerDiaEIR>`

With these features, Radiator Service Provider Module provides all the essential functionalities needed for carrier-grade AAA server software. Radiator Service Provider Module provides also the base platform for other Radiator components that can be added on the top of Radiator Service Provider Module. These components provide functionalities for several different use cases, such as Wi-Fi offloading and VoWiFi with Radiator SIM Module or VoLTE authentication with Radiator GBA/BSF Module.

2. Installing Radiator Service Provider Module

This section the describes how to install Radiator Service Provider Module.

2.1. Prerequisites

Radiator Service Provider Module requires Radiator 4.26 or later. SCTP multihoming support requires Radiator Radius::UtilXS Perl module available from Radiator downloads. Possible updates to the prerequisites are listed on Radiator Service Provider Module downloads [https://radiatorsoftware.com/products/radiator-service-provider-pack/] (login required). For more information on Radiator, see Radiator reference manual [https://radiatorsoftware.com/products/radiator/].

2.2. Installing and upgrading Radiator Service Provider Module

The recommended method is to install Radiator and Radiator Service Provider 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.

To install stand-alone Radiator Service Provider Module operating system specific packages:

1. Download Radiator Service Provider Module distribution package for your operating system.
2. Install it with the package manager. For example with Red Hat Enterprise Linux and CentOS:

   ```
   rpm -Uvh radiator-carrier-1.7-1.el7.noarch.rpm
   ```

To install Radiator Service Provider Module using source code package:

1. Download the Radiator Service Provider 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. Set Radiator to start automatically when booting. For more information, see Radiator reference manual [https://radiatorsoftware.com/products/radiator/].

3. Configuring Diameter peers

Diameter peer configuration is required by Diameter applications Radiator supports. Diameter applications and `<ServerDIAMETERTelco>` need peer definitions to work properly

All Diameter peers a Radiator instance talks to, must have a matching `DiaPeerDef` clause. A `DiaPeerDef` clause defines what is advertised to the peer, which applications are supported for the peer and if Radiator should actively open connections to the peer instead of waiting for a connection from the peer.

You can configure a single instance of Radiator to support, for example, relaying requests from some peers and processing the requests locally by the other peers.

You can configure Radiator to advertise certain applications to certain peers. The peer definition also determines if Radiator should act as an initiator and establish a connection to the peer. Alternatively Radiator can act as a responder waiting for the peer connection. It is possible to configure Radiator as an initiator for some peer connections and responder for the others.

A `<ServerDIAMETERTelco>` clause is required to create a listen socket for the incoming Diameter peer connections.

3.1. <DiaPeerDef attribute=value,attribute=value,...>

A `DiaPeerDef` clause defines describes and defines a Diameter peer this Radiator instance can be connected to. A Diameter connection can be initiated by Radiator or by the peer.

A minimal Radiator configuration requires one `DiaPeerDef` clause in addition to any `AuthBy DiaPCRF`, `DiaPCEF`, `DiaRelay` or other Diameter based AuthBys. When there is no `ServerDIAMETERTelco` clause, the `DiaPeerDef` clauses must be configured with the `Initiator` flag to connect to the Diameter peers.

A `ServerDIAMETERTelco` clause allows accepting incoming Diameter connections. When a `ServerDIAMETERTelco` is configured, Radiator will act as a Diameter responder. The settings for the connecting peers are looked up from `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 a `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.

Here is an example of configuring *`DiaPeerDef`* clauses. The first clause defines a Diameter peer that Radiator connects to. The connection is made to the IP address and port configured within the clause. An IP address and port are only needed when `Initiator` flag parameter is set.

The second DiaPeerDef clause defines a peer that must initiate a connection. When a Diameter peer connects to Radiator, the transport layer responder parameters, such as use of TLS, are defined within

a `<ServerDIAMETERTelco>` clause. When the transport layer is successfully set up, the peer sends a Diameter `Capabilities-Exchange-Request (CER)`. If the CER has `Origin-Host` with value `epdg.epc.mnc001.mcc001.3gppnetwork.org` , it matches the first `DiaPeerDef` and defines how Radiator responds to the `CER`.

```
# Our DRA requires that Radiator initiates the connection
<DiaPeerDef Origin-Host=dra.mnc001.mcc001.3gppnetwork.org>
    Identifier example-dra

    SupportedVendorIds 3GPP
    AuthApplicationIds 3GPP SWx, 3GPP SWm, 3GPP S6b

    ProductName Radiator 3GPP AAA Server

    Peer 172.16.172.80
    Port 3868
    Initiator

    OriginHost radiator-3gpp.aaa.mnc001.mcc001.3gppnetwork.org
    OriginRealm aaa.mnc001.mcc001.3gppnetwork.org
</DiaPeerDef>

# Definition of direct peering our ePDG initiates
<DiaPeerDef Origin-Host=epdg.epc.mnc001.mcc001.3gppnetwork.org>
    Identifier example-epdg

    SupportedVendorIds 3GPP
    AuthApplicationIds 3GPP SWm

    ProductName Radiator 3GPP AAA Server

    OriginHost radiator-3gpp.aaa.mnc001.mcc001.3gppnetwork.org
    OriginRealm aaa.mnc001.mcc001.3gppnetwork.org
</DiaPeerDef>

# Listen to incoming Diameter connections
<ServerDIAMETERTelco>
    Identifier diameter-server

    Port 3880
    # TLS and other settings
</ServerDIAMETERTelco>
```

3.1.1. Identifier

This is an optional parameter, which defines the name of the specific *`<DiaPeerDef>`* clause and its configuration.

3.1.2. 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 (Evolved Packet Data Gateway), must act as a initiator.

3.1.3. AddToRequestFromDia

This parameter defines the Diameter attributes, which are added to a request object in addition with *OriginHost* on page 4 and *OriginRealm* on page 4. 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.

3.1.4. 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.

3.1.5. NoReplyHook

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

3.1.6. NoreplyTimeout

This integer defines how soon, in seconds, *NoReplyHook* on page 4 is called if the request stored in proxy does not receive a reply. The default value is **5**.

3.1.7. 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**.

3.1.8. 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.

3.1.9. 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.

3.1.10. 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.

3.1.11. 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.

3.1.12. 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
```

3.1.13. 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
```

3.1.14. 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
```

3.1.15. 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
```

3.1.16. 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.

3.1.17. 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>.

3.1.18. 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
```

3.1.19. 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>.

3.1.20. 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. Configuring DIAMETER server

A <ServerDIAMETERTelco> clause is required to create a listen socket for the incoming Diameter peer connections.

4.1. <ServerDIAMETERTelco>

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

4.1.1. 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.

4.1.2. 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>.

4.1.3. 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://files.radiatorsoftware.com/radiator/ref.pdf] under section <ServerDIAMETER>.

4.1.4. MaxBufferSize

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

4.1.5. 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>.

4.1.6. 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://files.radiatorsoftware.com/radiator/ref.pdf] under section <ServerDIAMETER>.

4.1.7. 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. Configuring Diameter relay

Radiator supports Diameter relay functionality with *`<AuthBy DiaRelay>`* clause. One or more Diameter relays can be supported by a single Radiator instance. The Diameter relays are independent from each other.

5.1. <AuthBy DiaRelay>

This section describes the configuration parameters of *`<AuthBy DiaRelay>`*. See the distribution package goodies directory for relay configuration samples.

5.1.1. DiaPeerDef

This parameter defines the Diameter Peer requests should be relayed to. At least one DiaPeerDef parameters is required and some load balancing relay algorithms support multiple destinations. Relay algorithms may also have restrictions on how which attributes the lookup is based on. For more information, see Section 5.1.2. RelayAlgorithm on page 8.

Example

```
# Relay the requests to peer defined by DiaPeerDef with
# Identifier radiator-dia-auth-server
DiaPeerDef DiaPeerDef-Identifier=radiator-dia-auth-server

# It is also possible to look up peer based on applications
# it advertises.
#DiaPeerDef Peer-Auth-Application-Id=3GPP:3GPP SWx
```

5.1.2. RelayAlgorithm

This parameter defines the how Diameters requests should be relayed to the peers. At least one DiaPeerDef parameters is required. Some load balancing relay algorithms support multiple destinations.

The following algorithms are currently supported:

- `FailOver`

  Messages are sent to one peer only. If the peer becomes unreachable, the next configured peer is used. When a failed peer becomes available, it's used again.

- `HashBalance`

  Messages are distributed between all configured peers. If a peer is unavailable, the messages are distributed between the remaining peers. Diameter `Session-Id` attribute is used as the distribution key.

`FailOver` can use run-time information about peers for selecting the next hop. For example when a peer advertises its supported applications, `Peer-Auth-Application-Id` can be used to select peers based on the applications they currently advertise.

`HashBalance` adds to its targets only those `DiaPeerDef` clauses that can be found during Radiator startup. Using `DiaPeerDef-Identifier` is recommended. Here's an example:

Example

```
# Balance load to multiple Diameter peers based on Diameter Session-Id attribute
RelayAlgorithm HashBalance

# Balance to peers defined with these Identifier values
DiaPeerDef DiaPeerDef-Identifier=aaa-server1
DiaPeerDef DiaPeerDef-Identifier=aaa-server2
DiaPeerDef DiaPeerDef-Identifier=aaa-server3
DiaPeerDef DiaPeerDef-Identifier=aaa-server4
```

6. Configuring <MessageLog FILECARRIER>

This section describes the configuring parameters of *<MessageLog FILECARRIER>*.

6.1. Filename

This string defines the file name into which the log statistics are saved. The default value is **`%L/messagelog`**. The file name can include special formatting characters.

6.2. Format

This enumeration is an optional parameter, which defines the file output format. The possible values are **`text`**, **`libpcap`**, and **`text2pcap`**. The default value is **`text`** and it is also applied when the value is not defined.

7. 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).

7.1. <AuthBy DiaEIR>

This section describes the configuring parameters of *`<AuthBy DiaEIR>`*.

`<AuthBy DiaEIR>` queries IMEI status from EIR using Diameter S13/S13' interface. It can also query the software version and IMSI (International mobile subscriber identity) status. The result from EIR determines if the request is accepted or rejected.

The distribution package contains an example configuration file `goodies/eir-client.cfg`.

7.1.1. DiaEIR

This object list enables EIR check and identifies the used *`DiaEIR`* clause. This has no default value but it must be set.

7.1.2. MakeAnswer

When *`MakeAnswer`* is set, a simple Diameter answer with either DIAMETER_SUCCESS or DIAMETER_UNABLE_TO_COMPLY is sent if *`<AuthBy DiaEIR>`* does not return ignore. This parameter is not set by default.

7.1.3. EIR_UnknownAction

`EIR_UnknownAction` defines the return value for *`<AuthBy DiaEIR>`* if the EIR request gets DIAMETER_UNKNOWN_EQUIPMENT. The value can be either **`accept`** or **`reject`**, **`accept`** is the default value.

7.1.4. EIR_AttributesHook

This parameter is a hook that allows customising how to set IMEI and other values are passed to EIR. This has not any default value.

When this is set, *`IMEIAttribute`*, *`IMSIAttribute`*, and *`SoftwareVersionAttribute`* are ignored.

Here is an example that shows how to get the values from current request attributes:

```
EIR_AttributesHook sub
      { my $p = $_[0]; my $imei = $_[1]; my $sv = $_[2]; my $imsi = $_[3]; \
      $$imei = $p->get_attr('X-OSC-SIM-IMEI'); \
      $$sv   = $p->get_attr('X-OSC-SIM-Software-Version'); \
      $$imsi = $p->get_attr('X-OSC-SIM-IMSI'); }
```

7.1.5. IMEIAttribute

This parameter defines the IMEI attribute in the current request that must have a value for IMEI. This is ignored when *EIR_AttributesHook* is defined. The default value is **OSC-SIM-IMEI**.

7.1.6. SoftwareVersionAttribute

This parameter defines the software version attribute in the current request that must have a value for software version. This is ignored when *EIR_AttributesHook* is defined. The default value is **OSC-SIM-Software-Version**.

7.1.7. IMSIAttribute

This parameter defines the IMSI attribute in the current request that must have a value for IMSI. This is ignored when *EIR_AttributesHook* is defined. The default value is **OSC-SIM-IMSI**.

7.2. <DiaEIR>

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

7.2.1. 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.

7.2.2. DiaPeerDef

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

7.2.3. EIRCache

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

7.3. <EIRCacheInternal>

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

7.3.1. 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.

7.3.2. CacheTimeout

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

7.3.3. 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).

7.4. <ServerDiaEIR>

This section describes the configuring parameters of *<ServerDiaEIR>*. Apart from the parameters listed here, *<ServerDiaEIR>* inherits other parameters from *<ServerDIAMETER>*. *<ServerDIAMETER>* is documented in Radiator reference manual [https://radiatorsoftware.com/products/radiator/].

The distribution package contains an example configuration file `goodies/eir-server.cfg`. The test IMEIs are currently listed in `ServerEIR.pm`. Different test IMEIs trigger different responses, including errors, from this example EIR server. You can add your test device's IMEI into `ServerEIR.pm`. Later it will be possible to create a separate configuration file for *<ServerDiaEIR>* that contains the IMEI list.

7.4.1. VendorAuthApplicationIds

VendorAuthApplicationIds is an optional parameter that defines the Vendor-Specific-Application-Ids announced in CEA. The default value is **S13/S13'**.

8. Configuring <ServerDHCP>

This section describes the configuration parameters of *<ServerDHCP>*. *<ServerDHCP>* handles DHCP (Dynamic Host Configuration Protocol) requests by converting them to RADIUS requests. The converted RADIUS requests can be used for authentication and dynamic address allocation.

To serve DHCP requests, Radiator must be able to bind to DHCP server port 67. This typically requires that Radiator is run as root. Because DHCP uses fixed port numbers 67 and 68, there cannot be other DHCP services or Radiator instances configured with *<AddressAllocator DHCP>* or *<ServerDHCP>* on the same host, unless separate bind IP addresses are available.

DHCP options in Radiator configuration use the names defined by IANA. For example, DHCP option 51 that is typically known as "lease time" or "IP Address Lease Time" is called "Address Time" in Radiator configuration. For a configuration example, see `goodies/server-dhcp.cfg`.

8.1. UsernameOption

This string parameter defines the name of the DHCP option that is used as a value for User-Name in the request dispatched to *Handler*. The default value is **Hostname**.

8.2. DefaultUsername

This string parameter defines the User-Name that is used in requests dispatched to *Handler* if *UsernameOption* does not define this option in the DHCP request. The default value is **Unknown-DHCP-User**.

8.3. DHCPyiaddrAttr

This string defines the attribute name in RADIUS reply for *yiaddr* in DHCP replies. The default value is **Framed-IP-Address**.

8.4. DHCPSubnetMaskAttr

This string defines the attribute name in RADIUS reply for Subnet Mask option in DHCP replies. If this is left empty, *DefaultDHCPSubnetMask* is used instead. The default value is **Framed-IP-Netmask**.

8.5. DefaultDHCPSubnetMask

This is a string parameter. If *DHCPSubnetMask* has no value, this is used instead. Special formatting characters are supported. This has no default value.

8.6. DHCPRouterAttr

This string defines the attribute name in RADIUS reply for Router option in DHCP replies. All instances in the reply are added. This has no default value.

8.7. DHCPDomainServerAttr

This string defines the attribute name in RADIUS reply for Domain Server option in DHCP replies. All instances in the reply are added. This has no default value.

8.8. DHCPAddressTimeAttr

This string defines the attribute name in RADIUS reply for Address Time, also known as lease time, option in DHCP replies. The default value is **`Session-Timeout`**.

8.9. DefaultDHCPAddressTime

This is an integer parameter. If *`DHCPAddressTime`* has no value, this parameter is used instead. Special formatting characters are supported. The default value is **`86400`**.

8.10. DHCPClientIdentifier

This string defines the value used for Client Id option in DHCP replies. Special formatting characters are supported. The default value is **`%{User-Name}`**.

9. <AuthBy DiaINTERNAL>

This section describes how to configure *`<AuthBy DiaINTERNAL>`* module.

`<AuthBy DiaINTERNAL>` provides similar functionality to Radiator's *`<AuthBy INTERNAL>`*. The distribution package contains an example configuration file `goodies/dia-internal.cfg`.

Here is an example of using *`<AuthBy DiaINTERNAL>`*:

```
<AuthBy DiaINTERNAL>
      Identifier authby-dia-internal

      # DefaultResult defaults to DIAMETER_UNABLE_TO_COMPLY. Other
      # results are not set by default.

      #AuthResult DIAMETER_SUCCESS
      #AcctResult DIAMETER_SUCCESS
      #DefaultResult DIAMETER_SUCCESS
</AuthBy>
```

9.1. DefaultResult

This string parameter defines the Diameter result to use if no request specific result code is specified. The default value is **`DIAMETER_UNABLE_TO_COMPLY`**.

9.2. AuthResult

This string defines the Diameter result that is used for NASREQ Authentication requests. This is not set by default.

9.3. AcctResult

This string defines the Diameter result that is used for NASREQ Accounting requests. This is not set by default.

10. Using VSA framework for customised attributes

This section describes how to enable advanced attribute encoding and decoding with Radiator's VSA (Vendor-Specific Attributes) framework.

Radiator's VSA framework supports creating vendor-specific modules for encoding and decoding attributes that have non-standard formats. The attribute type for these attributes is set to *`custom`* in Radiator's dictionary.

At this moment, the distribution package includes a customised dictionary file, `dictionary.custom_vsa`, and 2 vendor-specific modules:

- `Vendor_10415.pm`

  This module contains the routines for customised 3GPP VSAs.

- `Vendor_40808.pm`

  This module contains the routines for customised Wi-Fi Alliance VSAs.

The customised dictionary file, `dictionary.custom_vsa`, redefines some attributes present in the default dictionary file. The redefined attributes are passed the included vendor-specific modules.

Here is an example of decoding attribute *`3GPP-GPRS-QoS-Profile`* when `dictionary.custom_vsa` is first disabled and then enabled:

```
Wed Dec 13 17:48:50 2017: DEBUG: Packet dump:
*** Received from 127.0.0.1 port 59458 ....
Code:       Access-Request
Identifier: 193
Authentic:  <230>\<232><168><1><151><191>/OZ<171>k<203><164><199><206>
Attributes:
        User-Name = "mikem"
        Service-Type = Framed-User
        NAS-IP-Address = 203.63.154.1
        NAS-Identifier = "203.63.154.1"
        NAS-Port = 1234
        Called-Station-Id = "123456789"
        Calling-Station-Id = "987654321"
        NAS-Port-Type = Async
        User-Password = o<244><24><21><12>-<191>v<233><217>o<6><165>0<137><230>
        3GPP-GPRS-QoS-Profile = "99-23731f9301858574597878


Wed Dec 13 17:51:27 2017: DEBUG: Packet dump:
*** Received from 127.0.0.1 port 60790 ....
Code:       Access-Request
Identifier: 85
Authentic:  {<152><151>~<149>Z<157>@w<250><146>B<7><218><253><234>
Attributes:
        User-Name = "mikem"
        Service-Type = Framed-User
        NAS-IP-Address = 203.63.154.1
        NAS-Identifier = "203.63.154.1"
```

```
        NAS-Port = 1234
        Called-Station-Id = "123456789"
        Calling-Station-Id = "987654321"
        NAS-Port-Type = Async
        User-Password = <187><166><234><164>Q<150>bu<132><3>c<0><16>02<175>
        3GPP-GPRS-QoS-Profile = 99-23731f9301858574597878
        OSC-3GPP-GPRS-Release-Indicator = 99
        OSC-3GPP-GPRS-Peak-Throughput-Class = Up-to-64kBps
        OSC-3GPP-GPRS-Mean-Throughput-Class = Best-Effort-Mean-Throughput
        OSC-3GPP-GPRS-Traffic-Class = Background-Class
        OSC-3GPP-GPRS-Max-Uplink = 896
        OSC-3GPP-GPRS-Guaranteed-Uplink = 512
        OSC-3GPP-GPRS-Max-Downlink = 896
        OSC-3GPP-GPRS-Guaranteed-Downlink = 512
```

11. Abbreviations

Capabilities Exchange Answer

CEA (Capabilities Exchange Answer)

Acronym: **CEA**

Capabilities Exchange Request

CER (Capabilities Exchange Request)

Acronym: **CER**

Dynamic Host Configuration Protocol

DHCP (Dynamic Host Configuration Protocol)

Acronym: **DHCP**

Evolved Packet Data Gateway

ePDG (Evolved Packet Data Gateway)

Acronym: **ePDG**

Equipment Identity Register

EIR (Equipment Identity Register)

Acronym: **EIR**

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**

Transport Layer Security

TLS (Transport Layer Security)

Acronym: **TLS**

Virtual Routing and Forwarding

VRF (Virtual Routing and Forwarding)

Acronym: **VRF**

Vendor-Specific Attributes

VSA (Vendor-Specific Attributes)

Acronym: **VSA**