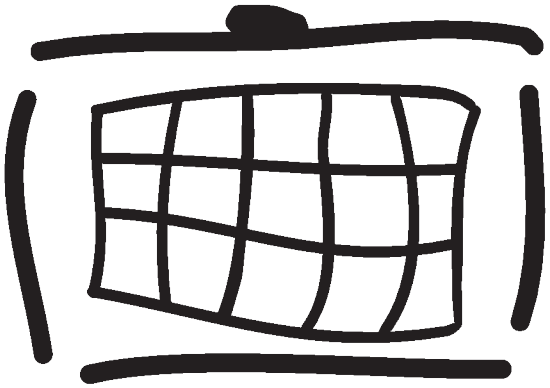
Radiator GBA/BSF Support module reference guide

Version: GBA/BSF Support module 1.3

Release date: May 18, 2016



Radiator

Table of Contents

1. Introduction to Radiator GBA/BSF Support module

This document describes how to install and configure the Radiator GBA/BSF Support module.

For more information about GBA/BSF (General Bootstrapping Architecture/Bootstrapping Server Functionality) architecture, see Radiator GBA/BSF whitepaper [https://www.open.com.au/radiator/GBA-BSF-whitepaper.pdf].

Radiator GBA/BSF Support module has 2 components, BSF (Bootstrapping Server Functionality) and NAF (Network Application Function)/AP (Application Proxy). The BSF component's hostname is *`bsf.ims.mncXXX.mccYYY.pub.3gppnetwork.org`* and it uses the following interfaces:

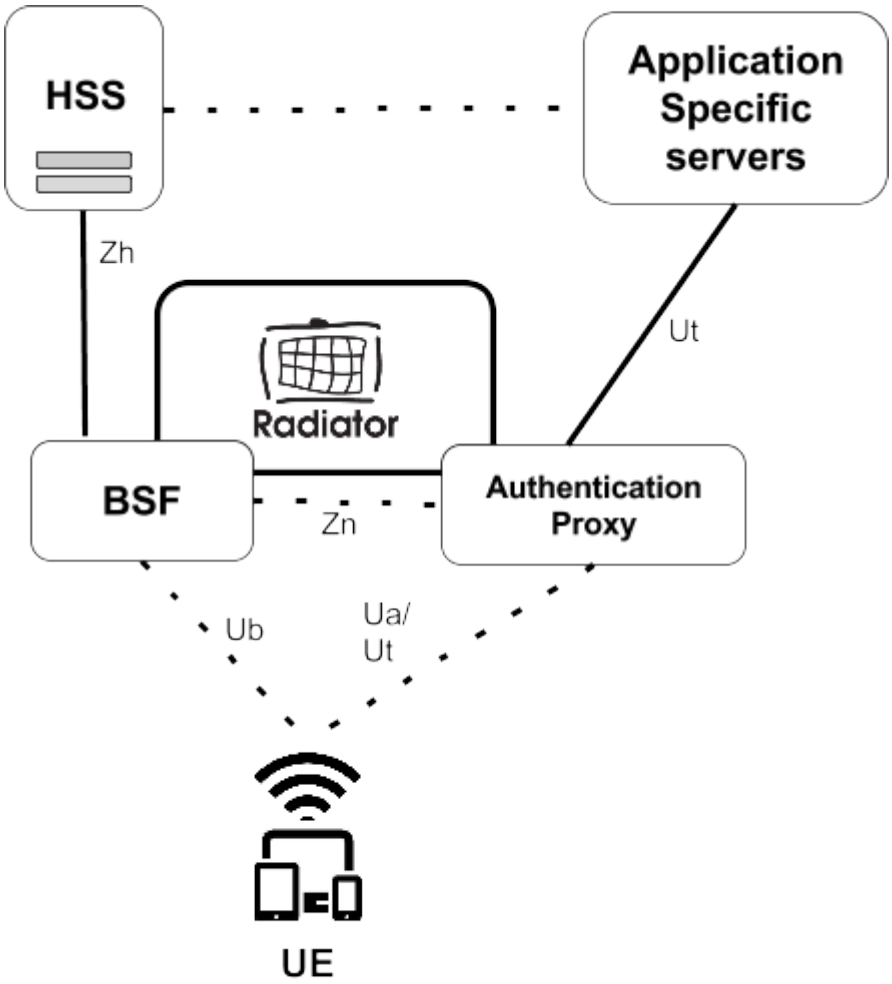
- HTTP/HTTPS Ub
- Diameter Zn
- Diameter Zh for connecting to the HSS (Home Subscriber Server) or DRA (Diameter Routing Agent)

The NAF/AP component's hostname is *`xcap.ims.mncXXX.mccYYY.pub.3gppnetwork.org`* and it uses the following interfaces:

- HTTP/HTTPS Ua
- Diameter Zn for connecting to the BSF

The following figure shows the basic architecture of GBA/BSF.

Figure 1. GBA/BSF architecture



2. Installing Radiator GBA/BSF Support module

Prerequisites

You need the following software components for installing Radiator GBA/BSF Support module:

- Radiator. For more information, see Radiator website [http://open.com.au/radiator/].
- Perl
- The following Perl libraries:
  - `XML::LibXML`
  - `Digest::SHA`
  - `MIME::Base64`
  - `Cache::FastMmap`
- OpenResty application server with NGINX, LuaJIT, and NGINX add-on modules. For more information, see OpenResty website [https://openresty.org/].

Procedure

To install Radiator GBA/BSF Support module:

1. Download the Radiator GBA/BSF Support module distribution.
2. Prepare the distribution for installation.

   ```
   perl Makefile.PL
   ```

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

   ```
   make install
   ```

   Instead of steps 1 and 2, you can copy the Radiator GBA/BSF Support module distribution's `Radius` directory into the `Radiator/Radius` directory.

4. Copy `lua/` directory into the `/opt/osc` directory.
5. Copy example configurations from `goodies/` directory into the `/etc/radiator` directory.
6. Edit the example configuration files to match your environment. If you are running BSF and AP components on the same host, combine the BSF and AP configurations into a single configuration file.
7. Start NGINX web server and Radiator.

   ```
   /etc/init.d/nginx start
   ```

   ```
   /etc/init.d/radiator start
   ```

8. To ensure your system is working correctly, check the log files. They are located in the following directories by default:

   - `/var/log/nginx`
   - `/var/log/radius`

3. Configuring Radiator GBA/BSF Support module

This section describes the configurable parameters of Radiator GBA/BSF Support module.

3.1. <AuthBy DiaBSF>

This section describes the configuring parameters of *`<AuthBy DiaBSF>`*. Apart from the parameters listed here, *`<AuthBy DiaBSF>`* inherits other parameters from *`AuthGeneric`*. These parameters are documented in Radiator reference manual [https://www.open.com.au/radiator/ref.pdf] under section <AuthBy xxxxxx>.

3.1.1. BSFZn

This object list lists the used BSFZn clauses.

3.2. <BSFZn>

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

3.2.1. CacheExpiry

This string defines the expiration time of entries in shared memory cache. The value **`0`** means there is no explicit expiration time, the least recently used value is expired first when needed. This value can be expressed in minutes (**`m`**), hours (**`h`**), or days (**`d`**). The default value is **`1d`**.

3.2.2. CacheSize

This string defines the size of a shared memory cache. This value can be expressed in kilobytes (**k**) or megabytes (**m**). The default value is **100m**.

3.2.3. OriginHost

This string defines the name that *<BSFZn>* uses to identify itself to the Diameter peers. It is sent to the Diameter peers in the Diameter BIR (Bootstrapping Info Request) and BIA (Bootstrapping Info Answer) messages. The Diameter peers use *OriginHost* to determine whether they have connected to the correct peer. *OriginHost* must be specified.

3.2.4. OriginRealm

This string defines the name of the realm the *<BSFZn>* uses. It is sent to the Diameter peers in the BIR and BIA messages. The peer uses *OriginRealm* to determine which requests are routed to this Radiator instance. *OriginRealm* must be specified.

3.2.5. RequireUSSForNaf

This flag defines whether the USS (User Security Settings) for NAF is required to exist for the subscriber. This is not set by default.

3.2.6. SendIMPI

This flag defines whether the subscriber's IMPI (IP Multimedia Private Identity) is sent back to NAF within Zn BIA. This is not set by default.

3.3. <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 GBA/BSF Support module 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.

3.3.1. Identifier

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

3.3.2. AddToRequestFromDia

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

3.3.3. 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.

3.3.4. NoReplyHook

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

3.3.5. NoReplyTimeout

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

3.3.6. 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**.

3.3.7. 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.

3.3.8. 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.

3.3.9. 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.

3.3.10. 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.

3.3.11. 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
```

3.3.12. 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
```

3.3.13. 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
```

3.3.14. 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
```

3.3.15. 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
```

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

3.3.17. 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.

3.3.18. 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://www.open.com.au/radiator/ref.pdf] under section <ServerDIAMETER>.

3.3.19. Protocol

This is an optional parameter, which specifies the connection protocol used for carrying the Diameter messages. For more information, see Radiator reference manual [https://www.open.com.au/radiator/ref.pdf] under section <ServerDIAMETER>.

3.3.20. UseTLS

This is an optional parameter, which defines if TLS (Transport Layer Security) encryption is used for authentication and encryption. For more information, see Radiator reference manual [https://www.open.com.au/radiator/ref.pdf] under section <ServerRADSEC>.

3.3.21. TLS_*

These parameters define the establishment of TLS authentication and encryption. For more information, see Radiator reference manual [https://www.open.com.au/radiator/ref.pdf] under section <ServerRADSEC>.

3.4. <Integrator>

This section describes the configuring parameters of <*Integrator*>. Integrator is an interface between NGINX web server and Radiator. HTTP Ub request is translated into Diameter Zh request and Ub to Zn correspondingly.

3.4.1. AddToRequest

This string defines any number of RADIUS attributes to the RADIUS requests generated by <*Integrator*>. The attributes can be used for tagging the authentication requests. This is an optional parameter.

3.4.2. BSFDefaultLifetime

This defines the default lifetime for BSF in seconds.

3.4.3. BSFServerName

This defines the names for used BSF servers. This parameter can have several values, separated by commas. If there are more than one value defined, the Zn request is sent to all BSF servers and the first successful reply is used to authenticate the Ua request.

3.4.4. HSSServerName

This defines the names for used HSS servers. This parameter can have several values, separated by commas. If more than one name is defined, the Zh request is sent to the first HSS server, to which Radiator has a Diameter peering up.

3.4.5. 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 MAR (Multimedia-Auth-Request) and MAA (Multimedia-Auth-Answer) messages. The Diameter peers use `OriginHost` to determine whether they have connected to the correct peer. `OriginHost` must be specified.

3.4.6. OriginRealm

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

3.4.7. 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.

3.4.8. 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.

3.4.9. SockPath

This defines the path to Unix domain socket that is used for communicating with NGINX.

3.5. <ServerDIAMETERTelco>

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

3.5.1. 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://www.open.com.au/radiator/ref.pdf] under section <ServerDIAMETER>.

3.5.2. Clients

This parameter defines the IP addresses of permitted clients. If not defined, all clients are permitted, subject to authentication. The parameter value is a list of comma- or space-separated IP addresses.

3.5.3. 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>.

3.5.4. 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>.

3.5.5. Protocol

This is an optional parameter, which specifies the connection protocol used for carrying the Diameter messages. For more information, see Radiator reference manual [https://www.open.com.au/radiator/ref.pdf] under section <ServerDIAMETER>.

3.5.6. 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>.

3.5.7. UseTLS

This is an optional parameter, which defines if TLS encryption is used for authentication and encryption. For more information, see Radiator reference manual [https://www.open.com.au/radiator/ref.pdf] under section <ServerRADSEC>.

3.5.8. TLS_*

These parameters define the establishment of TLS authentication and encryption. For more information, see Radiator reference manual [https://www.open.com.au/radiator/ref.pdf] under section <ServerRADSEC>.

4. Configuring NGINX

This section describes the relevant configuration parameters of NGINX. For more information about NGINX, see NGINX web site [https://www.nginx.com/].

4.1. ngx_ap_server_name

This parameter defines the AP name. This is used as the NAF name when calculating NAF keys. Therefore, it must match with the hostname the Ub clients are using.

Example:

```
set $ngx_ap_server_name 'xcap.ims.mncXXX.mccYYY.pub.3gppnetwork.org';
```

4.2. ngx_ap_socket_name

This parameter defines the name for the Unix domain socket, that is used for communicating with Radiator.

Example:

```
set $ngx_ap_socket_name 'unix:/tmp/radiator-naf.sock';
```

4.3. ngx_bsf_server_name

This parameter defines the BSF name. This is used as a realm in B-TID (Bootstrapping Transaction Identifier) user names.

Example:

```
set $ngx_bsf_server_name 'bsf.ims.mncXXX.mccYYY.pub.3gppnetwork.org';
```

4.4. ngx_bsf_socket_name

This parameter defines the name for the Unix domain socket, which is used for communicating with Radiator.

Example:

```
set $ngx_bsf_socket_name 'unix:/tmp/radiator-bsf.sock';
```

4.5. ngx_btid_username_realm

This parameter defines the BSF realm suffix.

Example:

```
set $ngx_btid_username_realm 'mncXXX.mccYYY.pub.3gppnetwork.org';
```

This setting allows the both types of realms to be used:

```
123456789012345@bsf.mncXXX.mccYYY.pub.3gppnetwork.org
123456789012345@bsf.ims.mncXXX.mccYYY.pub.3gppnetwork.org
```

4.6. ngx_default_lifetime

This parameter defines the default lifetime, in seconds, for BSF.

Example:

```
set $ngx_default_lifetime '86400';
```

4.7. ngx_gsid

This parameter defines the NAF type. The following options are available:

- **`0`**

  Unspecified service

- **`2`**

  AP (Authentication Proxy)

4.8. ngx_impi_username_realm

This parameter defines the IMPI realm suffix.

Example:

```
set $ngx_impi_username_realm 'mncXXX.mccYYY.3gppnetwork.org';
```

This setting allows the both types of realms to be used:

```
123456789012345@mncXXX.mccYYY.3gppnetwork.org
123456789012345@ims.mncXXX.mccYYY.3gppnetwork.org
```

5. Abbreviations

Authentication and Key Agreement

Acronym: **AKA**

Application Proxy

Acronym: **AP**

Authentication Proxy

Acronym: **AP**

Bootstrapping Info Answer

Acronym: **BIA**

Bootstrapping Info Request

Acronym: **BIR**

Bootstrapping Server Functionality

Acronym: **BSF**

Bootstrapping Transaction Identifier

Acronym: **B-TID**

Capabilities Exchange Answer

Acronym: **CEA**

Capabilities Exchange Request

Acronym: **CER**

Diameter Routing Agent

Acronym: **DRA**

Evolved Packet Data Gateway

Acronym: **ePDG**

General Bootstrapping Architecture

Acronym: **GBA**

General Bootstrapping Architecture/Bootstrapping Server Functionality

Acronym: **GBA/BSF**

Home Subscriber Server

Acronym: **HSS**

IP Multimedia Private Identity

Acronym: **IMPI**

Multimedia-Auth-Answer

Acronym: **MAA**

Multimedia-Auth-Request

Acronym: **MAR**

Network Application Function

Acronym: **NAF**

Transport Layer Security

Acronym: **TLS**

User Security Settings

Acronym: **USS**