Radiator® Policy and Charging Module

Installation and reference manual for Radiator® Policy and Charging Module 2.3. Last revised on January 25, 2023

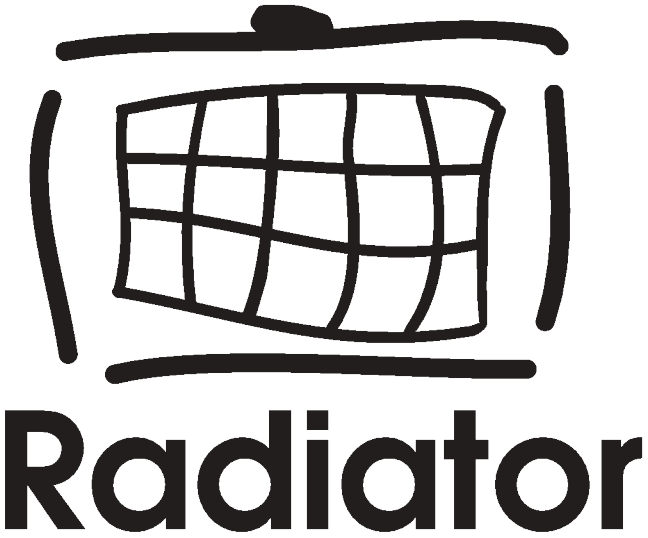

Table of Contents

1. Introduction to Radiator Policy and Charging Module

This document describes how to install and configure the Radiator policy and charging support for Radiator AAA Server from Radiator Software.

Diameter is an AAA (Authentication, Authorisation, Accounting) protocol commonly used by, but not limited to, telecommunication systems. Diameter protocols are called applications and use the Diameter base protocol. For example, Diameter Credit-Control Application defines the additional command codes and attributes required for supporting real-time credit-control. See RFC 6733 for the Diameter base protocol. Diameter applications supported by Radiator are introduced later in this reference manual.

RADIUS is the de facto AAA protocol for authenticating users and for recording accounting information. It is commonly used by Wireless Access Points (APs), Terminal Servers, or NAS (Network Access Server) whenever a user logs on and off network access devices or dial-up Internet service. It is supported and used by most vendors such as Cisco, Ericsson, Huawei, Juniper, Ruckus, Aruba, Alcatel-Lucent and others. See RFCs 2865 and 2866 for more details on the RADIUS protocol.

Radiator is a highly configurable and extensible AAA server that allows you to easily customize and control how you authenticate users and record accounting information.

Radiator runs on most Unix, Linux and Windows hosts. It is written entirely in Perl, and is therefore highly portable. Full source code is supplied, so you can alter the behaviour of Radiator's internals if you need to. Detailed list of supported systems and installation requirements are available in the Radiator reference manual https://files.radiatorsoftware.com/radiator/ref.pdf

2. Installaing Radiator Policy and Charging Module

Radiator runs on a wide range of platforms. The installation procedure depends on the platform and the type of package selected. Typical installation procedures are described below.

Start by installing Radiator. See the Radiator reference manual [https://files.radiatorsoftware.com/radiator/ref.pdf] for the installation instructions. When Radiator has been installed, proceed with Radiator policy and charging support installation.

2.1. Prerequisites

The prerequisites for Radiator and Radiator policy and charging support are defined in the Radiator reference manual. If you require Radiator SIM Pack for EAP-SIM (Extensible Authentication Protocol - Subscriber Identity Module), EAP-AKA (Extensible Authentication Protocol - Authentication and Key Agreement) or EAP-AKA' (Extensible Authentication Protocol - Authentication and Key Agreement Prime), see the Radiator SIM module reference manual for the installation instructions.

2.2. Installing and upgrading Radiator Policy and Charging Module

The recommended method is to install Radiator and other Radiator products, such as Radiator Policy and Charging 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 documentation [https://radiatorsoftware.com/products/radiator/] for additional installation topics, such as container installation, management with Ansible, package and repository download location, and post-installation and troubleshooting instructions.

Start by installing Radiator as described the Radiator reference manual [https://files.radiatorsoftware.com/radiator/ref.pdf] If you plan to use Radiator as an OCS or a PCRF, you need to also install the Radiator Service Provider module [https://radiatorsoftware.com/products/radiator-service-provider-pack/]

To install stand-alone Radiator Service Policy and Charging Module operating system specific packages:

1. Download Radiator Policy and Charging Module distribution package for your operating system.
2. Install it with the package manager. For example with Red Hat Enterprise Linux 9 and compatible systems:

   ```
   rpm -Uvh radiator-policy-charging-2.3-1.el9.noarch.rpm
   ```

To install Radiator Policy and Charging Module using source code package:

1. Unpack the distribution package into a separate working directory.
2. Change to the distribution directory:

   ```
   cd Radius-Policy-Charging-n.xx
   ```

3. Prepare the distribution for installation:

   ```
   perl Makefile.PL
   ```

4. Do the installation as root:

   ```
   make install
   ```

5. Build a Radiator configuration file based on the examples in `goodies/*.cfg`.
6. Prepare your desired Diameter application, such as PCRF (Policy and Charging Rules Function) or PCEF (Policy and Charging Enforcement Function) to use Radiator.
7. Run Radiator with the configuration file developed above.
8. Test and refine the configuration file.
9. Set Radiator to start automatically when booting. For more information, see Radiator reference manual [https://radiatorsoftware.com/products/radiator/].

3. General configuration

3.1. General information

The Radiator reference manual describes the details of Radiator configuration file and the statements that you can use in the configuration file to control the behaviour of the Radiator server, *`radiusd`*.

Radiator policy and charging support distribution comes with its own goodies directory which has configuration samples for PCRF, PCEF, and other applications. There are also examples for setting up SQL DB schemas and utilities for testing.

In general terms, the configuration file allows you control the following things:

- Behavior of the server in general, including Logging
- Which RADIUS clients the server will respond to, if RADIUS clients are required
- Which Diameter peers the server will work with
- Which Diameter applications are enabled

3.2. Diameter configuration

Radiator Diameter configuration requires defining one or more Diameter peers, the required Diameter applications and possible the Diameter server for accepting incoming Diameter connections from the peers. If no Diameter peers are configured, the Diameter applications can not create outgoing connections. Also, at least one Diameter peer definition is required to accept incoming Diameter connections from the peers.

Diameter peers are defined with `<DiaPeerDef ...>` clauses. Diameter applications that create outgoing connections use the parameters in `DiaPeerDef` to locate the destination host. The attributes advertised with the CER (Capabilities Exchange Request) messages sent to the peers during the initial capability negotiation phase are also defined in `DiaPeerDef` clauses.

Radiator can support different Diameter applications, such as Relay, PCRF or OCS (Online Charging System) simultaneously. Some applications, such as PCRF, may require defining additional clauses as described in this manual.

Some Diameter applications , such as the relay application `<AuthBy DiaRelay>` or online charging system `<AuthBy DiaOCS>` process messages from peers that have initiated a Diameter connection towards Radiator. To receive incoming connections, you need to define a `<Server DIAMETERTelco>` clause. This clause uses `DiaPeerDef` clauses to create a correct answer to the incoming CER from the connecting peers.

4. Radiator PCRF configuration

The PCRF support in Radiator uses multiple clauses. The main processing is done by *`<AuthBy DiaPCRF>`* which uses separate modules for Diameter Gx request processing, Rx request processing, accessing SPR (Subscriber Profile Repository) and Rx and Gx session information.

4.1. <AuthBy DiaPCRF>

A *`DiaPCRF`* clause defines PCRF which handles Diameter Gx requests originated by a PCEF running on a GGSN (Gateway GPRS Support Node), PDN GW (Packet Data Network Gateway), or other gateway.

The PCRF in Radiator has been designed to be modular and uses a number of modules to process the Diameter Gx and Rx messages, fetch and update information in the SPR, store Gx and Rx session information. These modules are configured with separate clauses and referenced by their identifier from *`<AuthBy DiaPCRF>`* clause.

A number of hooks are defined to customise the action when, for example, usage monitoring thresholds are reached, quota runs out or quota is used for the first time. The operators are encouraged to develop their own functions and Perl modules to handle the different events.

4.1.1. PCRFGx

The Identifier of *`<PCRFGx>`* clause to use for processing Diameter Gx messages. *`PCRFGx`* module is required to read information from Gx requests and add information to Gx answers. This allows separating the PCRF policy decisions from Diameter messages details.

4.1.2. PCRFRx

The Identifier of *`<PCRFRx>`* clause to use for processing Diameter Rx messages. *`PCRFRx`* module is required to read information from Rx requests and add information to Rx answers. This allows separating the PCRF policy decisions from Diameter messages details.

4.1.3. PCRFSPR

The Identifier of clause that defines a SPR. For example, the SQL based SPR shipped with Radiator policy and charging support modules is configured with *`<PCRFSPRSQL>`* clause.

The SPR clause must implement the interface defined by the `PCRFSPRGeneric.pm` module. The backend is not limited to SQL but can be LDAP (Lightweight Directory Access Protocol) or any repository that can hold subscriber information

4.1.4. PCRFGxSession

The Identifier of clause that defines a Gx session repository. For example, the SQL based GxSession module shipped with Radiator policy and charging support modules is configured with *<PCRFGxSessionSQL>* clause.

The GxSession clause must implement the interface defined by the `PCRFGxSessionGeneric.pm` module. The backend is not limited to SQL but should use a a backend that supports frequent creation and updating of session data.

4.1.5. PCRFRxSession

The Identifier of clause that defines a Rx session repository. For example, the SQL based RxSession module shipped with Radiator policy and charging support modules is configured with *<PCRFRxSessionSQL>* clause.

The RxSession clause must implement the interface defined by the `PCRFRxSessionGeneric.pm` module. The backend is not limited to SQL but should use a a backend that supports frequent creation and updating of session data

4.1.6. GxSUBIdTypes

Subscription-Id-Type values to consider when looking up subscribers from Gx *INITIAL_REQUEST* messages. Defaults to IMSI (International mobile subscriber identity) which will work for EAP-SIM and EAP-AKA based authentication. If, for example, PEAP (Protected Extensible Authentication Protocol) is used for authentication, PCEF may use NAI (Network Access Identifier) as the subscription id where NAI is the PEAP user name.

Example

```
# Support EAP-SIM, EAP-AKA, EAP-AKA' and PEAP based auth
GxSubIdTypes IMSI, NAI
```

4.1.7. SupportedGxFeatures

The default value for Gx Supported-Features is **`Rel11`**. You can set the value of Supported-Features as required with the *SupportedGxFeatures* option. The value is a comma-separated list of feature names.

Example

```
# We support Rel10 + Application Detection and Control
SupportedGxFeatures Rel10,ADC
```

4.1.8. SupportedRxFeatures

The default value for Rx Supported-Features is **`Rel10`**. You can set the value of Supported-Features as required with the *SupportedRxFeatures* option. The value is a comma-separated list of feature names.

Example

```
# We support Rel10 + UE address and mask in filters
SupportedRxFeatures Rel10,ExtendedFilter
```

4.1.9. DefaultOctets

Number of octets to allocate when requested by PCEF. Defaults to **`50 000 000`**.

Example

```
# Use a really small value for testing
DefaultOctets 200
```

4.1.10. MinOctets

Minimum number of octets to allocate when there is less than *`DefaultOctets`* left. Defaults to **`500 000`**.

Example

```
# Allow very small allocations
MinOctets 100
```

4.1.11. DefaultTime

Default time allocation when usage monitoring is enabled. Defaults to **`600`** seconds.

Example

```
# Allocate 20 minutes by default
DefaultTime 1200
```

4.1.12. MinTime

The minimum number of seconds to return when the usage monitoring quota is starting the reach the limit. Defaults to **`10`** seconds.

Example

```
# Use 15 seconds as the smallest available time
MinTime 15
```

4.1.13. EventTriggers

Comma-separated list of *`Event-Trigger`* attributes to return with CCA (Credit-Control Application). There is no default and the returned triggers depend on, for example, if usage monitoring is enabled. When usage monitoring is enabled, it will automatically return the appropriate trigger.

Example

```
# We want to know about OCS credit situation
EventTriggers OUT_OF_CREDIT,REALLOCATION_OF_CREDIT
```

4.1.14. EventTriggerHook

This hook will be called when CCR (Credit-Control-Request) contains *`Event-Trigger`* attributes. This allows, for example, activating or deactivating predefined PCC (Policy and Charging Control) rules in the PCEF with the CCA.

This hook is not defined by default.

The following arguments are passed to the hook in the following order:

- Reference to this AuthBy
- Reference to the current *PCRFmsg*
- Array of event trigger names

4.1.15. UsageMonitorInitHook

This hook will be called when the subscriber's quota is initialised to the value defined in the SPR data plan. The quota is initialised typically when during the first use or subsequently when the remaining quota is initialised by provisioning system or for some other external reason.

The default hook will log the event, but you may change the hook to call send SMS, call an external process or do any other action.

The following arguments are passed to the hook in the following order:

- Reference to this AuthBy
- Reference to the current *PCRFmsg*
- Number of octets in the initial quota

4.1.16. ThresholdHook

This hook will be called when the subscriber's quota reaches one or more threshold percentages defined in the subscriber's data plan.

The default hook will log the event, but you may change the hook to call send SMS, call an external process or do any other action.

Note

If multiple thresholds are reached during the quota allocation, the *ThresholdHook* will be called once for each threshold. The order of calls depends on the order the thresholds are defined in the SPR.

The following arguments are passed to the hook in the following order:

- Reference to this AuthBy
- Reference to the current *PCRFmsg*
- Threshold percentage that was reached
- Quota percentage before allocation but after unused bytes from previous allocation have been credited
- Quota percentage after allocation

4.1.17. MinUnitsHook

This hook will be called when the quota available for allocation for a subscriber reaches the minimum allocation as defined by the *MinOctets* or *MinOctets* configuration parameter. Quota available for allocation is defined as the remaining quota minus the sum of all outstanding allocations. The quota type (octets or time), remaining quota and sum of outstanding allocations are passed to the hook as parameters.

The default hook will log the event and set the *Result-Code* attribute to *DIAMETER_-CREDIT_LIMIT_REACHED* effectively terminating the session. You may change the hook to call send SMS, call an external process or do any other action.

The following arguments are passed to the hook in the following order:

- Reference to this AuthBy
- Reference to the current *PCRFmsg*
- Quota type as string: 'time' or 'octets'
- Remaining quota in bytes
- Outstanding allocated quota in bytes

4.1.18. RxSIPServiceIdentifier

RxSIPServiceIdentifier defines the Service-Identifier for Gx PCC rules installed in PCEF for AF (Application Function) SIP (Session Initiation Protocol) flows. There is no default.

Example

```
RxSIPServiceIdentifier 200
```

4.1.19. RxSIPRatingGroup

RxSIPRatingGroup defines the Rating-Group for Gx PCC rules installed in PCEF for AF SIP flows. There is no default.

Example

```
RxSIPRatingGroup 14
```

4.1.20. RxSIPReportingLevel

RxSIPReportingLevel defines on what level PCEF reports the usage for Gx PCC rule installed for AF SIP flow. There is no default.

Example

```
RxSIPReportingLevel SERVICE_IDENTIFIER_LEVEL
```

4.1.21. RxSIPOnline and RxSIPOffline

RxSIPOnline and *RxSIPOffline* enable or disable PCEF online and offline charging interfaces for Gx PCC rules installed for AF SIP flows. There is no default.

Example

```
RxSIPOnline DISABLE_ONLINE
RxSIPOffline ENABLE_OFFLINE
```

4.1.22. RxSIPMeteringMethod

RxSIPMeteringMethod controls PCEF online and offline charging. Possible value is one of: **DURATION**, **VOLUME** and **DURATION_VOLUME**. There is no default.

Example

```
RxSIPMeteringMethod DURATION
```

4.1.23. RxSIPPriorityLevel

RxSIPPriorityLevel sets the *Priority-Level* attribute value in *Allocation-Retention-Priority* attribute for PCC rules installed for AF SIP flows. If **0** or unset, no *AllocationRetention-Priority* is added in PCC rules. Otherwise *Allocation-Retention-Priority* is added with *Pre-emption-Capability* and *-Vulnerability* set to **PRE-EMPTION_CAPABILITY_ENABLED** and **PRE-EMPTION_VULNERABILITY_DISABLED**, respectively. There is no default.

Example

```
RxSIPPriorityLevel 8
```

4.2. <PCRFGx>

PCRFGx module processes incoming Diameter Gx messages and adds any attributes required by SPR PCC rules, usage monitoring and other processing done by PCRF.

The other modules used by *<Auth DiaPCRF>* do not directly use Diameter Gx messages.

4.2.1. Identifier

This optional parameter allows you to assign a symbolic name which is used by *<AuthBy DiaPCRF>* clause *PCRFGx* parameter.

4.3. <PCRFRx>

PCRFRx module processes incoming Diameter Rx messages and adds any attributes required by processing done by PCRF.

The other modules used by *<Auth DiaPCRF>* do not directly use Diameter Rx messages.

4.3.1. Identifier

This optional parameter allows you to assign a symbolic name which is used by *<AuthBy DiaPCRF>* clause *PCRFRx* parameter.

4.4. <PCRFSPRSQL>

The SPR that is shipped with Radiator policy and charging support uses SQL for storing the subscriber information.

<PCRFSPRSQL> implements the interface defined by `PCRFSPRGeneric.pm` which allows creating customised SPR that uses, for example, LDAP as the user repository

4.4.1. Identifier

This optional parameter allows you to assign a symbolic name which is used by *<AuthBy DiaPCRF>* SPR parameter.

4.4.2. SubscriberSelectE164

Defines the SQL statement that is called to find and fetch subscriber information based on the end user's E.164.

4.4.3. SubscriberSelectIMSI

Defines the SQL statement that is called to find and fetch subscriber information based on the end user's IMSI.

4.4.4. SubscriberSelectSIP_URI

Defines the SQL statement that is called to find and fetch subscriber information based on the end user's SIP URI.

4.4.5. SubscriberSelectNAI

Defines the SQL statement that is called to find and fetch subscriber information based on the end user's NAI.

4.4.6. SubscriberSelectPRIVATE

Defines the SQL statement that is called to find and fetch subscriber information based on the end user's private identifier.

4.4.7. SubscriberSelectById

Defines the SQL statement that is called to find and fetch subscriber information based on the end user's ID in the subscriber table. Used for example, when the ID is stored in the GxSession repository.

4.4.8. SubscriberSelectParam

This parameter specifies the bind variables to be used with all SubscriberSelect* statements. **`%0`** is replaced by end user's E.164, IMSI, SIP URI (Uniform Resource Identifier), NAI, or ID based on which type of identity is used to lookup the subscriber.

4.4.9. SubscriberColumnDef

This optional parameter allows you to define the way Radiator interprets the result of the SubscriberSelect* statements.

You can specify any number of *`SusbscriberColumnDef`* parameters, one for each interesting field returned by *`SubscriberColumnSelect`*. The general format is:

```
SubscriberColumnDef n, dataitem
```

- *`n`* is the index of the field in the result of *`SubscriberSelect`*. 0 is the first field.
- *`dataitem`* is the name of a entry in the subscriber information.

The subscriber information entries required by the modules shipped with Radiator policy and charging support use the following information:

Table 1. Subscriber information fetched from the SPR

Name	Description
`id`	Index or other identifier that uniquely identifies the subscription in the SPR
`e164`	E.164 formatted telephone number for the user
`imsi`	IMSI for the user
`nai`	NAI for the user
`sip_uri`	SIP_URI for the user
`private`	Private identifier for the user
`enabled`	Value the evaluates as true in Perl means the subscription is enabled

Name	Description
pcc_rules	Comma separated list of names from *pcc_rules* table
qos_information	Name of one entry in *qos_information* table
wlan_max_sess	Non-NULL if session count should be limited for RAT-Type WLAN
wlan_curr_sess	Current count of WLAN sessions
usage_tracking	Name of one entry in usage tracking table
quota_octets	Remaining default octet quota. Used when PCRF usage monitoring is enabled
all_reservations_octets	Total default octet quota reserved for sessions. Used when PCRF usage monitoring is enabled
lte_quota_octets	Non-NULL if usage monitoring it done for LTE. Currently RAT-types UTRAN, GERAN, GAN HSPA-Evolution and EUTRAN and the unknown RAT-Type are considered as LTE for octet usage monitoring purposes
lte_all_reservations_octets	Total octet quota reserved for LTE sessions. Used when PCRF usage monitoring is enabled for LTE octets
wlan_quota_octets	Non-NULL if usage monitoring it done for RAT-Type WLAN octets
wlan_all_reservations_octets	Total WLAN octet quota reserved for sessions. Used when PCRF usage monitoring is enabled for WLAN octets
wlan_quota_time	Non-NULL if usage monitoring is done for RAT-Type WLAN time
wlan_all_reservations_time	Total WLAN time quota reserved for sessions. Used when PCRF usage monitoring is enabled for WLAN time

4.4.10. UpdateSubscriberQuery

Defines the SQL statement to update usage monitoring and related information for a subscriber. This is typically done after a request is received over Gx.

4.4.11. UpdateSubscriberQueryParam

This parameter specifies the bind variables to be used with *UpdateSubscriberQuery*. **%0** is replaced by the id retrieved with *SubscriberSelect*. The other replacements are:

- **%1** adjustment in RAT-Type WLAN session count
- **%2** change in default octet quota
- **%3** change in default octet quota reserved across all sessions
- **%4** change in octet quota for RAT-Type LTE
- **%5** change in all octet reservations for RAT-Type LTE
- **%6** change in octet quota for RAT-Type WLAN
- **%7** change in all octet reservations for RAT-Type WLAN
- **%8** change in time quota for RAT-Type WLAN
- **%9** change in all time reservations RAT-Type WLAN

4.4.12. PCCRuleSelect

Defines the SQL statement to find and fetch the preconfigured PCC rule sets from the SPR.

4.4.13. PCCRuleSelectParam

This parameter specifies the bind variables to be used with *PCCRuleSelect*.

4.4.14. PCCRuleColumnDef

This optional parameter allows you to define the way Radiator interprets the result of the *PCCRuleSelect* statement.

You can specify any number of *PCCRuleColumnDef* parameters, one for each interesting field returned by *PCCRuleSelect*. The general format is:

```
PCCRuleColumnDef n, dataitem
```

- *n* is the index of the field in the result of *PCCRuleSelect*. 0 is the first field
- *dataitem* is the name of a entry in the PCC rule information

The subscriber information entries required by the modules shipped with Radiator policy and charging support use the following information:

Table 2. PCC rule information fetched from the SPR

Name	Description
id	Index or other identifier that uniquely identifies this PCC rule set in the SPR
name	Free form description of this rule set for humans. Not used with Gx protocol
rule_names	Comma separated list of names of rules predefined in the PCEF to activate
rule_basenames	Comma separated list of names of rule groups predefined in the PCEF to activate
dyn_rule_names	Comma separated list of names in *dynamic_rules* table
activation_time	*Activation-Time* for this *Charging-Rule-Install* in Diameter Time format
deactivation_time	*Deactivation-Time* for this *Charging-Rule-Install* in Diameter Time format

4.4.15. DynRuleSelect

Defines the SQL statement to find and fetch the dynamic PCC rule information from the SPR. The information is used to fill in *Charging-Rule-Definition* AVP (Attribute-Value Pair)s where each table row describes one AVP.

4.4.16. DynRuleSelectParam

This parameter specifies the bind variables to be used with *DynRuleSelect*.

4.4.17. DynRuleColumnDef

This optional parameter allows you to define the way Radiator interprets the result of the *DynRuleSelect* statement.

You can specify any number of *DynRuleColumnDef* parameters, one for each interesting field returned by *DynRuleSelect*. The general format is:

```
DynRuleColumnDef n, dataitem
```

- *n* is the index of the field in the result of DynRuleSelect. 0 is the first field.
- *dataitem* is the name of a entry in the dynamic rule information

The subscriber information entries required by the modules shipped with Radiator policy and charging support use the following information:

Table 3. Dynamic PCC rule information fetched from the SPR

Name	Description
id	Index or other identifier that uniquely identifies this dynamic PCC rule in the SPR
name	Free form description of this dynamic rule for humans. Not used with Gx protocol
rule_name	*Charging-Rule-Name* for this rule
tdf_application_id	*TDF-Application-Identifier* AVP value
flow_information	Comma separated list of names in *flow_information* table
qos_information	Name of entry in *qos_information* table
online	Value for *Online*, for example *DISABLE_ONLINE*
offline	Value for *Offline*, for example *ENABLE_OFFLINE*
precedence	Precedence AVP value

4.4.18. FlowInfoSelect

Defines the SQL statement to find and fetch flow information from the SPR for dynamic PCC rules. The information is used to fill in *Flow-Information* AVPs where each table row describes one AVP.

4.4.19. FlowInfoSelectParam

This parameter specifies the bind variables to be used with *FlowInfoSelect*.

4.4.20. FlowInfoColumnDef

This optional parameter allows you to define the way Radiator interprets the result of the *FlowInfoSelect* statement.

You can specify any number of *FlowInfoColumnDef* parameters, one for each interesting field returned by *FlowInfoSelect*. The general format is:

```
FlowInfoColumnDef n, dataitem
```

- *n* is the index of the field in the result of FlowInfoSelect. 0 is the first field.
- *dataitem* is the name of a entry in the dynamic rule information

The subscriber information entries required by the modules shipped with Radiator policy and charging support use the following information:

Table 4. Flow information fetched from the SPR

Name	Description
id	Index or other identifier that uniquely identifies this dynamic PCC rule in the SPR

Name	Description
name	Free form description of this flow for humans. Not used with Gx protocol
description	*IPFilter* rule, see Rx spec, 29.214 5.4.2
packet_filter_usage P	*Packet-Filter-Usage*: **SEND_TO_UE** if defined
tos_traffic_class	*ToS-Traffic-Class*
spi_index	*Security-Parameter-Index*
flow_label	IPv6 header flow label
flow_direction	Values for *Flow-Direction*, for example, **BIDIRECTIONAL**

4.4.21. QoSInfoSelect

Defines the SQL statement to find and fetch the subscribers QoS information based on the subscriber information.

4.4.22. QoSInfoSelectParam

This parameter specifies the bind variables to be used with *QoSInfoSelect*.

4.4.23. QoSInfoColumnDef

This optional parameter allows you to define the way Radiator interprets the result of the *QoSInfoSelect* statement.

You can specify any number of *QoSInfoColumnDef* parameters, one for each interesting field returned by *QoSInfoSelect*. The general format is:

```
QoSInfoColumnDef n, dataitem
```

- *n* is the index of the field in the result of QosInfoSelect. 0 is the first field.
- *dataitem* is the name of a entry in the QoS information

The subscriber information entries required by the modules shipped with Radiator policy and charging support use the following information:

Table 5. QoS information fetched from the SPR

Name	Description
id	Index or other identifier that uniquely identifies this PCC rule set in the SPR
name	Free form description of this rule set for humans. Not used with Gx protocol
qos_class_id	Value for *QoS-Class-Identifier* attribute in the Gx answer
max_requested_bw_ul	Value for *Max_Requested-BW-UL* attribute in the Gx answer
max_requested_bw_dl	Value for *Max_Requested-BW-DL* attribute in the Gx answer
guaranteed_bitrate_ul	Value for *Guaranteed-Bitrate-UL* attribute in the Gx answer
guaranteed_bitrate_dl	Value for *Guaranteed-Bitrate-DL* attribute in the Gx answer

Name	Description
alloc_retention_priority	Value for *Priority-Level* in *Allocation-Retention-Priority* in the Gx answer
apn_agg_max_bitrate_ul	Value for *APN-Aggregate-Max-Bitrate-UL* in the Gx answer
apn_agg_max_bitrate_dl	Value for *APN-Aggregate-Max-Bitrate-DL* in the Gx answer

4.4.24. DataPlanSelect

Defines the SQL statement to find and fetch subscriber's data plan information based on the subscriber information.

4.4.25. DataPlanSelectParam

This parameter specifies the bind variables to be used with *DataPlanSelect*. **%0** is replaced by the data plan identifier from the subscriber's usage tracking information.

4.4.26. DataPlanColumnDef

This optional parameter allows you to define the way Radiator interprets the result of the *DataPlanSelect* statement.

You can specify any number of *DataPlanColumnDef* parameters, one for each interesting field returned by *DataPlanSelect*. The general format is:

```
DataPlanColumnDef n, dataitem
```

- *n* is the index of the field in the result of *DataPlanSelect*. 0 is the first field.
- *dataitem* is the name of a entry in the data plan information

The subscriber information entries required by the modules shipped with Radiator policy and charging support use the following information:

Table 6. Data plan information fetched from the SPR

Name	Description
id	Index or other identifier that uniquely identifies this data plan in the SPR
name	Free form description of this data plan for humans. Not used with Gx protocol.
octets	Size of data plan in octets. Value is in octets, for example **1000000000** for **1*10^9** octets
default_octets	Default octets to allocate. Set to NULL to use *DefaultOctets* configuration parameter
min_octets	Minimum octets to allocate. Set to NULL to use *MinOctets* configuration parameter
time	Size of data plan in seconds
default_time	Default octets to allocate. Set to NULL to use *DefaultTime* configuration parameter
min_time	Minimum time to allocate. Set to NULL to use *MinTime* configuration parameter

Name	Description
thresholds	List of comma separated threshold percentages. For example, **75,50,25,0**. When one or more threshold values are reached, *ThresholdHook* is called once for each value.

4.4.27. UsageSelect

Defines the SQL statement to find and fetch information about the user related to usage monitoring, online and off line charging.

4.4.28. UsageSelectParam

This parameter specifies the bind variables to be used with *UsageSelect*. **%0** is replaced by the subscriber ID.

4.4.29. UsageColumnDef

This optional parameter allows you to define the way Radiator interprets the result of the *UsageSelect* statement.

You can specify any number of *UsageColumnDef* parameters, one for each interesting field returned by *UsageSelect*. The general format is:

```
UsageColumnDef n, dataitem
```

- *n* is the index of the field in the result of *UsageSelect*. 0 is the first field.
- *dataitem* is the name of a entry in the subscriber information

The usage tracking information entries required by the modules shipped with Radiator policy and charging support use the following information:

Table 7. Usage tracking information fetched from the SPR

Name	Description
id	Index or other identifier that uniquely identifies a single row in the SPR
name	Index of subscriber information table for the current subscriber
monitoring_enabled	Value the evaluates as true in Perl means usage monitoring is enabled
data_plan	Index or other identifier to Data plan, see *DataPlanSelect* for the details
online	Value for Online, for example **DISABLE_ONLINE**
offline	Value for Offline, for example **ENABLE_OFFLINE**
online_primary	Diameter URI for the primary online charging server
online_secondary	Diameter URI for the secondary online charging server
offline_primary	Diameter URI for the primary offline charging server
offline_secondary	Diameter URI for the secondary offline charging server

4.5. <PCRFGxSessionSQL>

<PCRFGxSessionSQL> stores information about subscribers' Gx sessions in SQL. The Gx session table has one row for each active session. By default session information is kept after the session is finished providing history information about all Gx sessions.

<PCRFGxSessionSQL> implements the interface defined by `PCRFGxSessionGeneric.pm` which allows creating customised Gx session repositories.

4.5.1. Identifier

This optional parameter allows you to assign a symbolic name which is used by *<AuthBy DiaPCRF PCRFGxSession>* parameter.

4.5.2. AddSessionQuery

Defines the SQL statement to add a new session after Gx *INITIAL_REQUEST* is received.

4.5.3. AddSessionQueryParam

This parameter specifies the bind variables to be used with *AddSessionQuery*. The values for **%0** - **%12** come from the incoming Gx message attributes and quota reserved for the new session.

4.5.4. UpdateSessionQuery

Defines the SQL statement to update a session after Gx *UPDATE_REQUEST* reporting used octets is received.

4.5.5. UpdateSessionQueryParam

This parameter specifies the bind variables to be used with UpdateSessionQuery. **%0** is replaced by the session ID fetched with *GetSessionSelect*. The other replacements are:

- **%1** change in reserved octet quota for this session
- **%2** change in reserved time quota for this session

4.5.6. CloseSessionQuery

Defines the SQL statement to call after Gx *TERMINATION_REQUEST* is received.

4.5.7. CloseSessionQueryParam

This parameter specifies the bind variables to be used with CloseSessionQuery. *%0* is replaced by the session id fetched with GetSessionSelect.

4.5.8. GetSessionSelect

Defines the SQL statement to find and fetch information about a Gx session.

4.5.9. GetSessionSelectParam

This parameter specifies the bind variables to be used with *GetSessionSelect*. *%0* is replaced by the *Diameter Session-Id* attribute value.

4.5.10. GetSessionColumnDef

This optional parameter allows you to define the way Radiator interprets the result of the *GetSessionSelect* statement.

You can specify any number of *GetSessionColumnDef* parameters, one for each interesting field returned by *GetSessionSelect*. The general format is:

```
GetSessionColumnDef n, dataitem
```

- *n* is the index of the field in the result of *GetSessionSelect*. **0** is the first field.
- *dataitem* is the name of a entry in the subscriber information

The Gx session information entries required by the modules shipped with Radiator policy and charging support use the following information:

Table 8. Session information fetched from the GxSession

Name	Description
id	Index or other identifier that uniquely identifies this session in this table
start_time	Unix timestamp of the session start
alive_time	Unix timestamp of the last update for this session
stop_time	Unix timestamp of the session end
subscription_id	Index of subscriber information table for this subscriber
session_id	*Session-Id* for this Gx session
imsi	IMSI for the subscriber
nai	NAI for the subscriber
e164	E.164 formatted telephone number for the subscriber
sip_uri	SIP URI for the subscriber
origin_host	*Origin-Host* value for this Rx session
origin_realm	*Origin-Realm* value for this Rx session
framed_ip_address	*Framed-IP-Address* for this Rx session
framed_ip_address	*Framed-IPv6-Prefix* for this Rx session
rat_type	*RAT-Type* value for this Gx session
reserved_octets	Number of octets currently allocated for this session
reserved_time	Number of seconds currently allocated for this session

4.5.11. GetSubscriberSessionsSelect

Defines the SQL statement to find and fetch information about multiple Gx sessions. Uses *GetSessionColumnDef* parameters. For this reason the query must return same values as *GetSessionSelect*. *%0* is the subscriber ID fetched from the SPR.

4.5.12. GetSubscriberSessionsSelectParam

This parameter specifies the bind variables to be used with *GetSubscriberSessionsSelect*. *%0* is replaced by the subscriber ID fetched from the SPR.

4.6. <PCRFRxSessionSQL>

<PCRFRxSessionSQL> stores information about subscribers' Rx sessions in SQL. The Rx session table has one row for each active session. By default session information is kept after the session is finished providing history information about all Rx sessions.

<PCRFRxSessionSQL> implements the interface defined by `PCRFRxSessionGeneric.pm` which allows creating customised Gx session repositories.

4.6.1. Identifier

This optional parameter allows you to assign a symbolic name which is used by *<AuthBy DiaPCRF PCRFRxSession>* parameter.

4.6.2. AddSessionQuery

Defines the SQL statement to add a new session after the first AAR (AA Request) for the session is received.

4.6.3. AddSessionQueryParam

This parameter specifies the bind variables to be used with *AddSessionQuery*. The values for *%0* - *%7* come from the incoming Rx message attributes.

4.6.4. UpdateSessionQuery

Defines the SQL statement to update a session when a Rx AAR update for an ongoing session is received.

4.6.5. UpdateSessionQueryParam

This parameter specifies the bind variables to be used with *UpdateSessionQuery*. *%0* is replaced by the session ID fetched with *GetSessionSelect*.

4.6.6. CloseSessionQuery

Defines the SQL statement to call after Rx STR (Session-Termination-Request) is received

4.6.7. CloseSessionQueryParam

This parameter specifies the bind variables to be used with *CloseSessionQuery*. *%0* is replaced by the session id fetched with *GetSessionSelect*.

4.6.8. GetSessionSelect

Defines the SQL statement to find and fetch information about a Rx session.

4.6.9. GetSessionSelectParam

This parameter specifies the bind variables to be used with *GetSessionSelect*. *%0* is replaced by the *Diameter Session-Id* attribute value.

4.6.10. GetSessionColumnDef

This optional parameter allows you to define the way Radiator interprets the result of the *GetSessionSelect* statement.

You can specify any number of *GetSessionColumnDef* parameters, one for each interesting field returned by *GetSessionSelect*. The general format is:

```
GetSessionColumnDef n, dataitem
```

- *n* is the index of the field in the result of *GetSessionSelect*. **0** is the first field

- *dataitem* is the name of a entry in the subscriber information

The Rx session information entries required by the modules shipped with Radiator policy and charging support use the following information:

Table 9. Session information fetched from the RxSession

Name	Description
id	Index or other identifier that uniquely identifies this session in this table
start_time	Unix timestamp of the session start
alive_time	Unix timestamp of the last update for this session
stop_time	Unix timestamp of the session end
subscription_id	Index of subscriber information table for this subscriber
session_id	*Session-Id* for this Rx session
imsi	IMSI for the subscriber
nai	NAI for the subscriber
e*164*	E.164 formatted telephone number for the subscriber
sip_uri	SIP URI for the subscriber
origin_host	*Origin-Host* value for this Rx session
origin_realm	*Origin-Realm* value for this Rx session
framed_ip_address	*Framed-IP-Address* for this Rx session
framed_ipv6_prefix	*Framed-IPv6-Prefix* for this Rx session

4.6.11. InstallGxRuleQuery

InstallGxRuleQuery defines the SQL statement to add a new entry in the table that tracks Gx PCC rules installed in PCEF for AF flows reported over Rx interface.

4.6.12. InstallGxRuleQueryParam

This parameter specifies the bind variables to be used with *InstallGxRuleQuery*. *%0* is replaced with the Rx Session-Id, *%1* with Gx *Session-Id*, *%2* with the newly installed PCC rule name, and *%3* with the subscriber's *SIP_URI*.

4.6.13. RemoveGxRuleQuery

RemoveGxRuleQuery defines the SQL statement to call when the Gx PCC rule for an AF flow is removed from the PCEF.

4.6.14. RemoveGxRuleQueryParam

RemoveGxRuleQueryParam specifies the bind variables to be used with *RemoveGxRuleQuery*. *%0* is replaced with the rule id fetched with *GxRuleSelect*.

4.6.15. GxRuleSelect

GxRuleSelect defines the SQL statement to call when the Gx PCC rules installed in PCEF for a certain Rx session need to be fetched.

4.6.16. GxRuleSelectParam

GxRuleSelectParam specifies the bind variables to be used *GxRuleSelect*. *%0* is replaced with the Rx *Session-Id*.

4.6.17. GxRuleColumnDef

GxRuleColumnDef allows you to define the way Radiator interprets the result of the *GxRuleSelect* statement.

You can specify any number of *GxRuleColumnDef* parameters, one for each interesting field returned by *GxRuleSelect*. The general format is:

```
GxRuleColumnDef n, dataitem
```

- *n* is the index of the field in the result of *GxRuleSelect*. **0** is the first field.
- *dataitem* is the name of a entry in the Gx PCC rule information

The Gx PCC rule information entries required by the modules shipped with Radiator policy and charging support use the following information:

Table 10. Gx PCC rule information for AF flows fetched from the database

Name	Description
id	Index or other identifier that uniquely identifies this rule in the table
start_time	Unix timestamp of the rule creation
rx_session_id	*Session-Id* of Rx session for which the rule was created for
gx_session_id	*Session-Id* of Gx session for which the rule belongs to
rule_name	Name of the Gx PCC rule installed in the PCEF
sip_uri	*SIP_URI* of the subscriber the rule was created for as reported over Rx

5. Radiator RADIUS - Diameter PCEF configuration

The PCEF included in Radiator policy and charging support allows you to use the existing PCRF, OCS and subscriber information for RADIUS AAA network access.

For example, you can authenticate users with SIM (Subscriber Identity Module)/USIM (Universal Subscriber Identity Module) or plain password, authorise the network use from your existing PCRF and use RADIUS accounting to enforce usage with usage monitoring done by PCRF or online charging done by OCS. This enables extending the same policies and charging services that are used for 3G/4G/LTE networks with Wi-Fi and other RADIUS AAA based networks.

See `goodies/eap_sim_wx_gx.cfg` for an example of EAP-SIM based authentication followed by PCRF based policy control and usage monitoring. Usage monitoring is done based on RADIUS Accounting-Request messages or WiMAX-PPAQ. Dynamic QoS changes can be applied based on CCA and RAR (Reauthentication Request) messages from the PCRF. Prepaid charging with OCS is similar to usage monitoring with PCRF.

WiMAX-PPAQ is described in the WiMAX forum document WMF-T33-002-R010v05 and internet draft draft-lior-radius-prepaid-extensions. The PCEF implementation uses WiMAX prepaid accounting automatically when the attributes are present in RADIUS authentication requests. Radiator PCEF can translate and send WiMAX prepaid accounting to both PCRF and OCS.

For another example that shows how to use both PCRF and OCS with Radiator's PCEF implementation, see `goodies/diameter-pcef.cfg`.

5.1. Common PCEF parameters

Here is listed the parameters that are common to the both PCEF configuration clauses *<AuthBy DiaGx>* and *<AuthBy DiaGy>*.

5.1.1. DiaPeerDef

Each *<AuthBy DiaGx>* and *<AuthBy DiaGy>* clause must have one *DiaPeerDef* parameter which identifies the *DiaPeerDef* clause that defines the PCRF or OCS the clause uses.

Example 1. DiaPeerDef for AuthBy DiaGx

```
# Use peer defined by DiaPeerDef with Identifier osc-pcrf
# as our PCRF
DiaPeerDef osc-pcrf
```

5.1.2. DefaultDestinationRealm

This optional string defines the used Diameter default realm, if the user name does not define the realm already.

5.1.3. DestinationRealm

This optional string defines the used Diameter default realm, regardless if the user name defines it already or not. This also overrides the *DefaultRealm*.

5.1.4. BindingAttribute

Name of the attribute that can be used to bind authentication requests to the subsequent accounting requests for a session. Defaults to **Acct-Session-Id**.

5.1.5. IMSIAttribute

Name of the attribute in the RADIUS request object that has the subscriber's IMSI as its value. Typically set by a previous AuthBy, such as *AuthBy SIMWX*. Defaults to **OSC-SIM-IMSI**.

5.1.6. TimeThreshold

Threshold value in seconds for requesting new time quota from the PCRF or OCS. When the quota left for the RADIUS session reaches the threshold, new quota must be requested for the session. Defaults to **5** seconds.

Example 2. TimeThreshold

```
# Make sure we have time to switch over to secondary PCRF
TimeThreshold 10
```

5.1.7. QuotaThreshold

Threshold value in octets for requesting new octet quota from the PCRF or OCS. When the quota left for the RADIUS session reaches the threshold, new quota must be requested for the session. Defaults to **100 000** octets.

Example 3. QuotaThreshold

```
# Raise the default value for high speed connections
```

```
QuotaThreshold 10000000
```

5.1.8. CCAInitialHook

This hook is called when Diameter CCA for *INITIAL_REQUEST* is received during the RADIUS authentication phase.

The following arguments are passed to the hook in the following order:

- Reference to this AuthBy
- Reference to a PCEFmsg that provides access to RADIUS and Diameter messages and other information related to session handling

5.1.9. CCAUpdateHook

This hook is called when Diameter CCA for *UPDATE_REQUEST* is received during RADIUS accounting or WiMAX usage update phase.

The following arguments are passed to the hook in the following order:

- Reference to this *AuthBy*
- Reference to a PCEFmsg that provides access to RADIUS and Diameter messages and other information related to session handling
- Total volume quota returned with Diameter CCA message
- Total time quota returned with Diameter CCA message

5.1.10. RARHook

This hook is called when Diameter RAR is received from the PCRF or OCS.

The following arguments are passed to the hook in the following order:

- Reference to this AuthBy
- Reference to a PCEFmsg that provides access to RADIUS and Diameter messages and other information related to session handling
- Reference to the RADIUS dynamic authorization request for sending from the hook

5.1.11. ReAuthenticationHook

This hook is called when a RADIUS *Access-Request* is received and lookup from the session database with the *BindingAttribute* indicates there is already an active session. The authentication is consider a reauthentication. The reauthentication is likely caused by a CoA-Request.

The following arguments are passed to the hook in the following order:

- Reference to this AuthBy
- Reference to a PCEFmsg that provides access to RADIUS and Diameter messages and other information related to session handling

5.1.12. MonitoringResponseHook

This hook defines the Perl function that is called for the following cases:

- Accounting-Responses with RADIUS accounting-based monitoring

- Access-Requests for WiMAX-based prepaid monitoring

The following arguments are passed to the hook in the following order:

- Reference to this AuthBy
- Reference to a PCEFmsg that provides access to RADIUS and Diameter messages and other information related to session handling
- Total time quota returned to the client
- Total volume quota returned to the client

5.1.13. ValidityTime

`ValidityTime` defines the quota validity time in seconds. If this option is not defined, *`Validity-Time`* attribute is not included in the CCA and the client will use its default value. There is no default value.

Example

```
# Instruct the client report no later than after 10 minutes
ValidityTime 600
```

5.1.14. DynauthSender

Identifier of module used for RFC 5176 dynamic authentication requests. *`%0`* is replaced with NAS source address. There is no default value.

Example 4. DynauthSender

```
# Send Disconnect and Change-of-Authorization messages
# using AuthBy RADIUS with Identifier that has format
# dynauth-sender-nnn.nnn.nnn.nnn
DynauthSender dynauth-sender-%0
```

5.1.15. DynauthIdentificationAttr

Attributes to copy from RADIUS request to the dynamic authentication request to identify the user session. Defaults to: **`NAS-IP-Address NAS-IPv6-Address Acct-Session-Id Calling-Station-Id`**. If the attribute is not present in the request that triggers the *`dynauth`* request, the attribute is not used.

Example 5. DynauthIdentificationAttr

```
# Attributes required by this vendor
DynauthIdentificationAttr Framed-IP-Address Acct-Session-Id
User-Name
```

5.1.16. DynauthNoMessageAuthenticator

Some implementations do not support *`Message-Authenticator`* in dynamic authentication requests. Defaults to not set and *`Message-Authenticator`* is sent.

Example 6. DynauthNoMessageAuthenticator

```
# Dynauth request fails with Message-Authenticator
DynauthNoMessageAuthenticator
```

5.2. <AuthBy DiaGx>

A *DiaGx* clause defines a PCEF. *<AuthBy DiaGx>* implements the PCEF functionality. It receives RADIUS messages from RADIUS clients, processes them, and sends the processed messages to PCRF. *<AuthBy DiaGx>* uses 3GPP Gx interface for communicating with PCRF.

5.2.1. NoMoreQuotaAction

This string defines the action that is taken when PCRF has previously returned quota and now stops returning it. The only possible value is **disconnect**. This is not set by default.

5.3. <AuthBy DiaGy>

The *<AuthBy DiaGy>* clause defines a PCEF. It receives RADIUS messages from RADIUS clients, processes them, and sends the processed messages to OCS. *<AuthBy DiaGy>* uses 3GPP Gy interface for communicating with OCS.

5.3.1. ServiceIdentifier

This integer defines the service identifier, which is used when requesting quota. This is not set by default.

5.3.2. RatingGroup

This integer defines the rating group, which is used when requesting quota. The default value is **9048**.

6. Radiator OCS configuration

The OCS support in Radiator uses multiple clauses. The main processing is done by *<AuthBy DiaOCS>*. This module uses separate modules for Diameter Gy request processing, accessing OCS database and Gy session information.

See `goodies/diameter-ocs.cfg` for a configuration example.

6.1. <AuthBy DiaOCS>

A *DiaOCS* clause defines OCS which handles Diameter Gy requests originated by a PCEF running on a GGSN, PDN GW or other gateway.

The OCS in Radiator has been designed to be modular and uses a number modules to process the Diameter Gy messages, fetch and update information in the OCS database and store Gy session information. These modules are configured with separate clauses and referenced by their identifier from *<AuthBy DiaOCS>* clause.

A number of hooks are defined to customise the action when, for example, quota thresholds are reached, quota runs out or quota is used for the first time. The operators are encouraged to develop their own functions and Perl modules to handle the different events.

6.1.1. OCSGy

The Identifier of *<OCSGy>* clause to use for processing Diameter Gy messages. *OCSGy* module is required to read information from Gy requests and add information to Gy answers. This allows separating the OCS logic from Diameter messages details.

6.1.2. OCSDB

The Identifier of clause that defines a database for OCS. For example, the SQL based OCS DB shipped with Radiator policy and charging support modules is configured with *<OCSDBSQL>* clause.

The SPR clause must implement the interface defined by the `OCSDBGeneric.pm` module. The backend is not limited to SQL but can be LDAP or any repository that can hold subscriber information.

6.1.3. OCSGySession

The Identifier of clause that defines a Gy session repository. For example, the SQL based *`OCSGySession`* module shipped with Radiator policy and charging support modules is configured with *`<OCSGySessionSQL>`* clause.

The O*`CSGySession`* clause must implement the interface defined by the `GySessionGeneric.pm` module. The backend is not limited to SQL but should use a a backend that supports frequent creation and updating of session data.

6.1.4. FirstUseHook

This hook will be called when the subscriber's quota still at the initial value and DCC (Diameter Credit-Control) message is received. The quota is initialised typically when during the first use or subsequently when the remaining quota is initialised by provisioning system or for some other external reason.

The default hook will log the event, but you may change the hook to call send SMS, call an external process or do any other action.

The following arguments are passed to the hook in the following order:

- Reference to this AuthBy
- Reference to the current *`OCSmsg`*
- Amount of quota left after this reservation

6.1.5. QuotaUseHook

This hook will be called for each *`Multiple-Services-Credit-Control`* attribute that reports unit usage. There is no default.

The following arguments are passed to the hook in the following order:

- Reference to this AuthBy
- Reference to the current *`OCSmsg`*
- Hash ref with the allocation type, rating group, service identifier, and other values related to the *`M-S-C-C`* and its allocation

6.1.6. FinalUnitHook

This hook will be called for each *`Multiple-Services-Credit-Control`* attribute that returns final units to the PCEF. The default is to add *`Final-Unit-Indication`* with *`Final-UnitAction = TERMINATE`*.

The following arguments are passed to the hook in the following order:

- Reference to this AuthBy
- Reference to the current *`OCSmsg`*
- Reference to the current *`Multiple-Services-Credit-Control`* attribute that has *`Service-Identifier`* and *`Rating-Group`* filled in, where applicable

6.1.7. Thresholds

This optional parameter allows you to define threshold percentages that trigger *`ThresholdHook`*.

Example

```
# Trigger ThresholdHook when half and little remaining
Thresholds 50, 10
```

6.1.8. ThresholdHook

This hook will be called when the subscriber's quota reaches one or more threshold percentages defined in the configuration.

The default hook will log the event, but you may change the hook to call send SMS, call an external process or do any other action.

Note

If multiple thresholds are reached during the quota allocation, the *`ThresholdHook`* will be called once for each threshold. The order of calls depends on the order the thresholds are configured with the *`Thresholds`* parameter.

The following arguments are passed to the hook in the following order:

- Reference to this AuthBy
- Reference to the current *`OCSmsg`*
- Threshold percentage that was reached
- Quota percentage before allocation but after unused bytes from previous allocation have been credited
- Quota percentage after allocation

6.1.9. ValidityTime

`ValidityTime` defines the quota validity time in seconds. If this option is not defined, *`Validity-Time`* attribute is not included in the CCA and the client will use its default value. There is no default value.

Example

```
# Instruct the client report no later than after 10 minutes
ValidityTime 600
```

6.1.10. TccMultiplier

This optional parameter defines the multiplier used with *`ValidityTime`* to derive Tcc timer value for Gy session supervision. Defaults to **`2`**.

Example

```
# Tcc timer expires a bit later than RFC suggests
TccMultiplier 2.1
```

6.1.11. TccTimer

When *`TccTimer`* is defined, it will set the value of Tcc timer in seconds directly ignoring the possible *`TccMultiplier`*. When Tcc timer is set to **`0`** with *`TccTimer`* or combination of *`ValidityTime`* and

TccMultiplier, or *TccCheckInterval* is **0**, Radiator will not do Gy session supervision. *TccTimer* is not defined by default.

Example

```
# Tcc timer is 1200. No dependency on the other variables
TccTimer 1200
```

6.1.12. TccCheckInterval

Interval in seconds to run Tcc timer checks. This affects only how often the checks are run. It does not affect how the Tcc timer value is calculated for Gy session supervision. When the *TccCheckInterval* is set to **0**, Radiator will not do Gy session supervision allowing an external process to supervise the sessions. The default is **30** seconds.

Example

```
# Check the Tcc timer expiries one a minute
TccCheckInterval 60
```

6.1.13. QuotaHoldingTime

Interval in seconds to run Tcc timer checks. This affects only how often the checks are run. It does not affect how the Tcc timer value is calculated for Gy session supervision. When the *TccCheckInterval* is set to **0**, Radiator will not do Gy session supervision allowing an external process to supervise the sessions. The default is **30** seconds.

Example

```
# Check the Tcc timer expiries one a minute
TccCheckInterval 60
```

6.1.14. Triggers

Triggers defines the Trigger AVP contents returned with *INITIAL_REQUEST* CCA. Empty value is permitted causing empty Trigger AVP to be sent. If the option is not defined, which is the default, then Trigger AVP is not included in the CCA and the client defaults are used.

Example

```
# Disable all triggers
Triggers
```

6.1.15. SessionFailover

When SessionFailover is set, Radiator will add *CC-Session-Failover* in CCA messages. The possible values are: **FAILOVER_NOT_SUPPORTED** and **FAILOVER_SUPPORTED**. *SessionFailover* is not set by default, and hence the clients default to **FAILOVER_NOT_SUPPORTED**.

Example

```
# Tell clients we support failover
SessionFailover FAILOVER_SUPPORTED
```

6.1.16. FailureHandling

When F*ailureHandling* is set, Radiator will add *Credit-Control-Failure-Handling* in CCA messages. The possible values are: **TERMINATE**, **CONTINUE**, and **RETRY_AND_TERMINATE**. *FailureHandling* is not set by default, and hence the clients default to **TERMINATE**.

Example

```
# Tell clients they can try our other instances
FailureHandling RETRY_AND_TERMINATE
```

6.2. <OCSGy>

OCSGy module processes incoming Diameter Gy messages and adds any attributes required by allocations and other processing done by OCS.

The other modules used by *<Auth DiaOCS>* do not directly use Diameter Gy messages.

6.2.1. Identifier

This optional parameter allows you to assign a symbolic name which is used by *<AuthBy DiaOCS>* clause *OCSGy* parameter.

6.3. <OCSDBSQL>

The OCSDB (Online Charging System Database) that is shipped with Radiator policy and charging support uses SQL for storing the information.

OCSDBSQL implements the interface defined by `OCSDBGeneric.pm` which allows creating customised OCSDB that uses, for example, LDAP as the repository.

6.3.1. Identifier

This optional parameter allows you to assign a symbolic name which is used by *<AuthBy DiaOCS>* SPR parameter.

6.3.2. SubscriberSelectE164

Defines the SQL statement that is called to find and fetch subscriber information based on the end user's E.164.

6.3.3. SubscriberSelectIMSI

Defines the SQL statement that is called to find and fetch subscriber information based on the end user's IMSI.

6.3.4. SubscriberSelectPRIVATE

Defines the SQL statement that is called to find and fetch subscriber information based on the end user's private identifier.

6.3.5. SubscriberSelectById

Defines the SQL statement that is called to find and fetch subscriber information based on the end user's ID in the subscriber table. Used for example, when the ID is stored in the GxSession repository.

6.3.6. SubscriberSelectParam

This parameter specifies the bind variables to be used with all SubscriberSelect* statements. **%0** is replaced by end user's E.164, IMSI, SIP URI, NAI, or ID based on which type of identity is used to lookup the subscriber.

6.3.7. SubscriberColumnDef

This parameter allows you to define the way Radiator interprets the result of the *SubscriberSelect** statements.

You can specify any number of *SubscriberColumnDef* parameters, one for each interesting field returned by *SubscriberColumnSelect*. The general format is:

```
SubscriberColumnDef n, dataitem
```

- *n* is the index of the field in the result of *SubscriberSelect**. **0** is the first field.
- *dataitem* is the name of a entry in the subscriber information

The subscriber information entries required by the modules shipped with Radiator policy and charging support use the following information:

Table 11. Subscriber information fetched from the OCSDB

Name	Description
id	Index or other identifier that uniquely identifies the subscription in the OCSDB
e164	E.164 formatted telephone number for the user
imsi	IMSI for the user
nai	NAI for the user
private	Private identifier for the user
enabled	Value the evaluates as true in Perl means the subscription is enabled
initial_quota	Initial quota for the subscriber
quota	Currently remaining quota
all_reservations	Quota that is currently reserved across all active Gy sessions

6.3.8. ServiceSelect

Defines the SQL statement to fetch all supported services.

6.3.9. ServiceColumnDef

This parameter allows you to define the way Radiator interprets the result of the *ServiceSelect* statement.

You can specify any number of *ServiceColumnDef* parameters, one for each interesting field returned by *ServiceSelect*. The general format is:

```
ServiceColumnDef n, dataitem
```

- *n* is the index of the field in the result of *ServiceSelect*. **0** is the first field.
- *dataitem* is the name of a entry in the PCC rule information

The service information entries required by the modules shipped with Radiator policy and charging support use the following information:

Table 12. Service information fetched from the OCSDB

Name	Description
id	Index or other identifier that uniquely identifies this service in the OCSDB
name	Free form description of this service for humans. Not used with Gy protocol.
service_identifier	Value of *Service-Identifier* attribute in Gy messages
default_allocation	How many service units are allocated for the service by default
unit_type	Currently supported value is *TOTAL_OCTETS*
unit_price	Price for one *unit_type*
rating_group	Defined as value of *Rating-Group* attribute if service belongs to a rating group. **NULL** or empty otherwise
pool_identifier	Defined as value of *G-S-U-Pool-Identifier* attribute if service belongs to a pool. **NULL** or empty otherwise

6.3.10. RatingGroupSelect

SQL statement to fetch all supported rating groups.

6.3.11. RatingGroupColumnDef

This optional parameter allows you to define the way Radiator interprets the result of the *RatingGroupSelect* statement.

You can specify any number of *RatingGroupColumnDef* parameters, one for each interesting field returned by *RatingGroupSelect*. The general format is:

```
RatingGroupColumnDef n, dataitem
```

- *n* is the index of the field in the result of *RatingGroupSelect*. **0** is the first field.
- *dataitem* is the name of a entry in the data plan information

The service information entries required by the modules shipped with Radiator policy and charging support use the following information:

Table 13. Rating group information fetched from the OCSDB

Name	Description
id	Index or other identifier that uniquely identifies this rating group in the OCSDB
name	Free form description of this rating group for humans. Not used with Gy protocol.
rating_group	Value of *Rating-Group* attribute in Gy messages
default_allocation	How many service units are allocated for the rating group by default
unit_type	Currently supported value is *TOTAL_OCTETS*
unit_price	Price for one *unit_type*
pool_identifier	Defined as value of *G-S-U-Pool-Identifier* attribute if the rating group belongs to a pool. **NULL** or empty otherwise

6.3.12. PoolSelect

SQL statement to fetch all supported pool definitions.

6.3.13. PoolColumnDef

This optional parameter allows you to define the way Radiator interprets the result of the *PoolSelect* statement.

You can specify any number of *PoolColumnDef* parameters, one for each interesting field returned by *PoolSelect*. The general format is:

```
PoolColumnDef n, dataitem
```

- *n* is the index of the field in the result of *PoolSelect*. **0** is the first field
- *dataitem* is the name of a entry in the subscriber information

The usage tracking information entries required by the modules shipped with Radiator policy and charging support use the following information:

Table 14. Pool information fetched from the OCSDB

Name	Description
id	Index or other identifier that uniquely identifies this pool in the OCSDB
name	Free form description of this pool for humans. Not used with Gy protocol.
pool_identifier	Value of *G-S-U-Pool-Identifier* attribute for this pool
default_allocation	How many service units are allocated for the pool by default
exponent	Value of Exponent in the *Unit-Value* attribute for this pool

6.3.14. RefreshPeriod

RefreshPeriod specifies the time period in seconds that *OCSDBSQL* will refresh the service, rating group and pool definitions by rereading the database. If set to **0**, then *OCSDBSQL* will only read the definitions from the database at startup and on *SIGHUP*. Defaults to **0**.

Example

```
# Refresh the informations two times a day
RefreshPeriod 43200.
```

6.4. <OCSGySessionSQL>

<OCSGySessionSQL> stores information about subscribers' Gy sessions and reservations within sessions in SQL. The Gy session table has one row for each active session. For each service, rating group and pool reservation within a session, one row is created to hold the allocation information. By default session and allocation information is kept after the session is finished providing history information about all Gy sessions.

<OCSGySessionSQL> implements the interface defined by `OCSGySessionGeneric.pm` which allows creating customised Gy session repositories.

6.4.1. Identifier

This optional parameter allows you to assign a symbolic name which is used by *AuthBy DiaOCS GySession* parameter.

6.4.2. AddSessionQuery

SQL query to add a new session after Gy *INITIAL_REQUEST* is received.

6.4.3. AddSessionQueryParam

This parameter specifies the bind variables to be used with *AddSessionQuery*. The values for **%0 - %7** come from *OCSmsg*.

6.4.4. UpdateSessionQuery

SQL query to run when Gy *UPDATE_REQUEST* is received.

6.4.5. UpdateSessionQueryParam

This parameter specifies the bind variables to be used with *UpdateSessionQuery*. **%0** is the ID fetched with *GetSessionSelect*. **%1** is the quota reservation change for this session.

6.4.6. CloseSessionQuery

SQL query to run when Gy *TERMINATION_REQUEST* is received.

6.4.7. CloseSessionQueryParam

This parameter specifies the bind variables to be used with *CloseSessionQuery*. **%0** is the id fetched with *GetSessionSelect*.

6.4.8. GetSessionSelect

SQL query to fetch information about a Gy session.

6.4.9. GetSessionSelectParam

This parameter specifies the bind variables to be used with *GetSessionSelect*. **%0** is replaced by the *Diameter Session-Id* attribute value.

6.4.10. GetSessionColumnDef

This optional parameter allows you to define the way Radiator interprets the result of the *GetSessionSelect* statement.

You can specify any number of *GetSessionColumnDef* parameters, one for each interesting field returned by *GetSessionSelect*. The general format is:

```
GetSessionColumnDef n, dataitem
```

- *n* is the index of the field in the result of *GetSessionSelect*. **0** is the first field.
- *dataitem* is the name of a entry in the subscriber information

The subscriber information entries required by the modules shipped with Radiator policy and charging support use the following information:

Table 15. Session information fetched from the GySession

Name	Description
id	Index or other identifier that uniquely identifies this session in the Gy Session
start_time	Unix timestamp for the session start

Name	Description
reserved_quota	Quota currently reserved for this session
subscription_id	Value of id identifying the subscriber as fetched by OCSDB
session_id	*Session-Id* for this Gy session
imsi	IMSI for the subscriber
nai	NAI for the subscriber
e164	E.164 formatted telephone number for the subscriber

6.4.11. TccQuery

SQL query to fetch and optionally close sessions for which the Tcc timer has expired.

6.4.12. TccQueryParam

This parameter specifies the bind variables to be used with *TccQuery*. **%0** is the Unix timestamp adjusted by Tcc timer value from *Auth DiaOCS*.

6.4.13. TccCleanupQuery

SQL query to process sessions for which Tcc timer has expired. If the query is not defined, it will not be run allowing *TccQuery* to do the clean up.

6.4.14. TccCleanupQueryParam

This parameter specifies the bind variables to be used with *TccCleanupQuery*. **%0** is the Unix timestamp adjusted by Tcc timer value from *Auth DiaOCS*.

6.4.15. AddAllocationQuery

SQL query to add one allocation for a Gy session. An allocation is done when quota is reserved for a service, rating group or a pool. For pooled allocations, one allocation is added to describe the whole pool and one allocation is added for each pooled service and rating group.

6.4.16. AddAllocationQueryParam

This parameter specifies the bind variables to be used with *AddAllocationQuery*.

- **%0** is the id fetched with *GetAllocationsSelect*
- **%1** tells if this reservation is for a session, rating group or pool
- **%2** is the *Service-Identifier*
- **%3** is the *Rating-Group*
- **%4** is the number of the reserved service units
- **%5** is the amount of quota corresponding to the reserved service units
- **%6** is the pool identifier which identifies the information about a pool in OCSDB

6.4.17. UpdateServiceAllocationQuery

SQL query to update a service based allocation for a Gy session.

6.4.18. UpdateServiceAllocationQueryParam

This parameter specifies the bind variables to be used with *`UpdateServiceAllocationQuery`*.

- **`%0`** is the id fetched with *`GetAllocationsSelect`*
- **`%1`** is the *`Service-Identifier`*
- **`%2`** is the *`Rating-Group`*
- **`%3`** is the number of the reserved service units
- **`%4`** is the amount of quota corresponding to the reserved service units
- **`%5`** is the pool identifier which identifies the information about a pool in OCSDB

6.4.19. UpdateRGAllocationQuery

SQL query to update a rating group based allocation for a Gy session. Behaves the same as *`UpdateServiceAllocationQuery`* and uses *`UpdateRGAllocationQueryParam`* with the same special values as defined for *`UpdateServiceAllocationQueryParam`*.

6.4.20. UpdatePoolAllocationQuery

SQL query to update a rating group based allocation for a Gy session. Behaves the same as *`UpdateServiceAllocationQuery`* and uses *`UpdateRGAllocationQueryParam`* with the same special values as defined for *`UpdateServiceAllocationQueryParam`*.

6.4.21. GetAllocationsSelect

SQL query to run when information about all active allocations for a Gy session is needed.

6.4.22. GetAllocationsSelectParam

This parameter specifies the bind variables to be used with *`GetAllocationsSelect`*. **`%0`** is replaced by the *`Diameter Session-Id`* attribute value.

6.4.23. GetAllocationsColumnDef

This optional parameter allows you to define the way Radiator interprets the result of the *`GetAllocationsSelect`* statement.

You can specify any number of *`GetSessionColumnDef parameters`*, one for each interesting field returned by *`GetSessionSelect`*. The general format is:

```
GetAllocationsColumnDef n, dataitem
```

- *`n`* is the index of the field in the result of *`GetAllocationsSelect`*. **`0`** is the first field.
- *`dataitem`* is the name of a entry in the subscriber information

The subscriber information entries required by the modules shipped with Radiator policy and charging support use the following information:

Table 16. Allocation information fetched from the GySession

Name	Description
`id`	Index or other identifier that uniquely identifies this allocation in the Gy session
`allocation_type`	Is this allocation for a service, rating group or pool

Name	Description
service_identifier	Has a non-empty and non-NULL value for service allocation
rating_group	Has a non-empty and non-NULL value for rating group allocation
pool_identifier	Has a non-empty and non-NULL when the service or rating group is a pool member or the allocation describes the pool
service_units	The number of units allocated for this service or rating group. **Note** Not meaningful for allocation the describes a pool
allocated_quota	The amount of quota this the *service_units* reserve
p*ool_identifier*	Has a non-empty and non-NULL when the service or rating group is a pool member or the allocation describes the pool

6.4.24. CloseAllocationQuery

SQL query to run when this allocation is not in use anymore.

6.4.25. CloseAllocationQueryParam

This parameter specifies the bind variables to be used with *CloseAllocationQuery*. **%0** is the allocation ID.

7. Abbreviations

Authentication, Authorisation, Accounting

AAA (Authentication, Authorisation, Accounting)

Acronym: **AAA**

AA Request

AAR (AA Request)

Acronym: **AAR**

Application Function

AF (Application Function)

Acronym: **AF**

Attribute-Value Pair

AVP (Attribute-Value Pair)

Acronym: **AVP**

Credit-Control Application

CCA (Credit-Control Application)

Acronym: **CCA**

Credit-Control-Request

CCR (Credit-Control-Request)

Acronym: **CCR**

Capabilities Exchange Answer

CEA (Capabilities Exchange Answer)

Acronym: **CEA**

Capabilities Exchange Request

CER (Capabilities Exchange Request)

Acronym: **CER**

Diameter Credit-COntrol

DCC (Diameter Credit-Control)

Acronym: **DCC**

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**

Evolved Packet Data Gateway

ePDG (Evolved Packet Data Gateway)

Acronym: **ePDG**

Gateway GPRS Support Node

GGSN (Gateway GPRS Support Node)

Acronym: **GGSN**

International mobile subscriber identity

IMSI (International mobile subscriber identity)

Acronym: **IMSI**

Lightweight Directory Access Protocol

LDAP (Lightweight Directory Access Protocol)

Acronym: **LDAP**

Network Access Identifier

NAI (Network Access Identifier)

Acronym: **NAI**

Network Access Server

NAS (Network Access Server)

Acronym: **NAS**

Online Charging System

OCS (Online Charging System)

Acronym: **OCS**

Online Charging System Database

OCSDB (Online Charging System Database)

Acronym: **OCSDB**

Offline Charging System

OFCS (Offline Charging System)

Acronym: **OFCS**

Policy and Charging Control

PCC (Policy and Charging Control)

Acronym: **PCC**

Policy and Charging Enforcement Function

PCEF (Policy and Charging Enforcement Function)

Acronym: **PCEF**

Policy and Charging Rules Function

PCRF (Policy and Charging Rules Function)

Acronym: **PCRF**

Packet Data Network Gateway

PDN GW (Packet Data Network Gateway)

Acronym: **PDN GW**

Note

Sometimes acronym PGW is used.

Protected Extensible Authentication Protocol

PEAP (Protected Extensible Authentication Protocol)

Acronym: **PEAP**

Reauthentication Answer

RAA (Reauthentication Answer)

Acronym: **RAA**

Reauthentication Request

RAR (Reauthentication Request)

Acronym: **RAR**

Subscriber Identity Module

SIM (Subscriber Identity Module)

Acronym: **SIM**

Session Initiation Protocol

SIP (Session Initiation Protocol)

Acronym: **SIP**

Subscriber Profile Repository

SPR (Subscriber Profile Repository)

Acronym: **SPR**

Session-Termination-Request

STR (Session-Termination-Request)

Acronym: **STR**

Transport Layer Security

TLS (Transport Layer Security)

Acronym: **TLS**

Uniform Resource Identifier

URI (Uniform Resource Identifier)

Acronym: **URI**

Universal Subscriber Identity Module

USIM (Universal Subscriber Identity Module)

Acronym: **USIM**

Virtual Routing and Forwarding

VRF (Virtual Routing and Forwarding)

Acronym: **VRF**