OpenSER Admin Course

Accounting with OpenSER

Voice System SRL

http://www.voice-system.ro

http://www.openser.org

- accounting in OpenSER is based on events and not on CDRs
  - OpenSER generates START and STOP – like events for each call
  - not able to generated CDRs as OpenSER is not call statefull
- how to get CDRs?
  - use an external entity to match the START and STOP event from the same call, by using the call attributes
    - Call-ID
    - From tag
    - To tag

IMPORTANT: as there is no dialog state, OpenSER cannot figure out if the STOP event is missing. Form SIP point of view, there is no guarantee that a call will terminate via BYE (like unplugged or disconnected phones)

How to do accounting

- auto accounting
  - this is transaction based accounting, so you need to use stateful processing of the request
  - you just have to mark the transaction you want to be accounted and OpenSER will automatically do it
  - based on the method name, OpenSER will decide if there will be a START or STOP event
- manual accounting
  - make use of script function that are exported by the “acc” module
  - it is the script writer decision what and when to account

Accounting can be done via multiple (even in parallel) backends:

- **log**
  - the accounting information is logged (syslog or stderr)
  - each event is a log line
  - attribute format: “name1=value1;name2=value2”
  - available by default
- **database**
  - accounting information is stored in database
  - each event is a table row
  - attributes go into different columns
  - available by default, but requires to load a database module

- **radius**
  - the accounting information is sent via RADIUS accounting requests
  - each event is a separate RADIUS request
  - attributes are mapped as RADIUS AVPs
  - not available by default; you need to enable it at compile time, by editing the module Makefile
  - requires external library for radius client implementation:
    - libradiusclient-ng
    - freeradius
- **diameter**
  - not updated

What is accounted?

- all accounting records contain the following minimum information:
  - method name (from request)
  - from tag (from request)
  - to tag (from reply)
  - callid (from request)
  - sip code (from reply)
  - sip status (from reply)

- to this, you can add whatever else other information via "**extra accounting**" functionality

- **log**

ACC: transaction answered: timestamp=1190362043;method=INVITE;from_tag=327EF7EB;to_tag=b137d780e2b72679i0;call_id=1629788544@192.168.2.2;code=200;reason=OK

ACC: transaction answered: timestamp=1190362055;method=BYE;from_tag=327EF7EB;to_tag=b137d780e2b72679i0;call_id=1629788544@192.168.2.2;code=200;reason=OK

- **DB**

```
+----+--------+-----------+-------------+---------------------------------+-----+------+--------------------------+
| id | method| from_tag| to_tag       | callid                          | sip_code | sip_reason | time       |
+----+--------+-----------+-------------+---------------------------------+-----+------+--------------------------+
|  5 | INVITE | 1610014 | 5DF5E36E | 65EC-AEA6@192.168.1.3 | 200  | OK  | 2007-04-30 20:31:36 |
|  6 | ACK    | 1610014 | 5DF5E36E | 65EC-AEA6@192.168.1.3 | 200  | OK  | 2007-04-30 20:31:36 |
|  7 | BYE    | 1610014 | 5DF5E36E | 65EC-AEA6@192.168.1.3 | 200  | Ok  | 2007-04-30 20:31:58 |
```

- **RADIUS**

Acct-Status-Type = Start
Service-Type = Sip-Session
Sip-Response-Code = 200
Sip-Method = INVITE
Event-Timestamp = 1142177361
Sip-To-Tag = "00D0E90101B8_T9513"
Sip-From-Tag = "111aa0fda452c726"
Acct-Session-Id = "1dbe198c8@192.168.0.11"
Sip-Src-IP = "192.168.0.11"
Sip-Src-Port = "5068"
NAS-IP-Address = 127.0.0.1
NAS-Port = 5060
Client-IP-Address = 10.10.10.10
Acct-Unique-Session-Id = "37fb00358437ff4d"

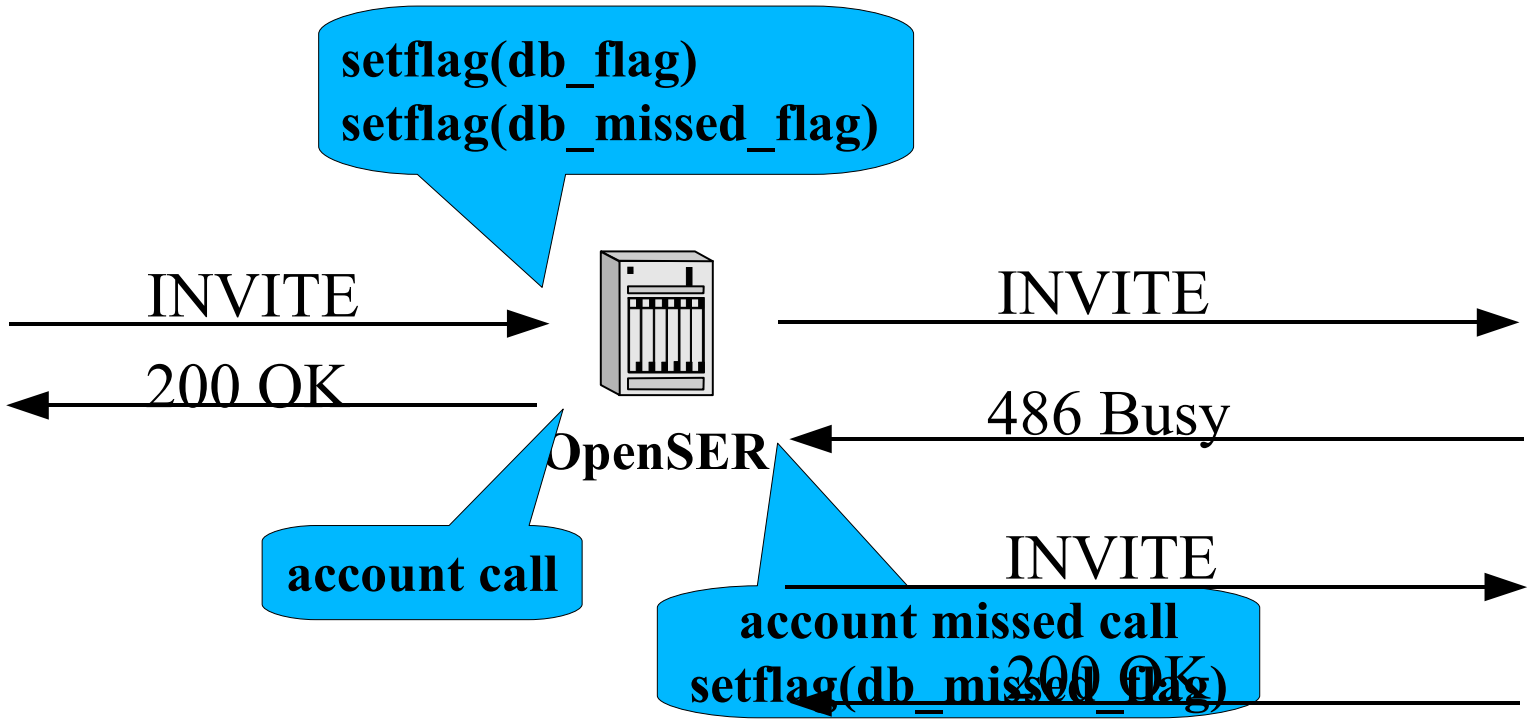
Acct-Status-Type = Stop
Service-Type = Sip-Session
Sip-Response-Code = 200
Sip-Method = BYE
Event-Timestamp = 1142177661
Sip-To-Tag = "00D0E90101B8_T9513"
Sip-From-Tag = "111aa0fda452c726"
Acct-Session-Id = "1dbe198c8@192.168.0.11"
Sip-Src-IP = "192.168.0.11"
Sip-Src-Port = "5068"
NAS-IP-Address = 127.0.0.1
NAS-Port = 5060
Client-IP-Address = 10.10.10.10
Acct-Unique-Session-Id = "37fb00358437ff4d"

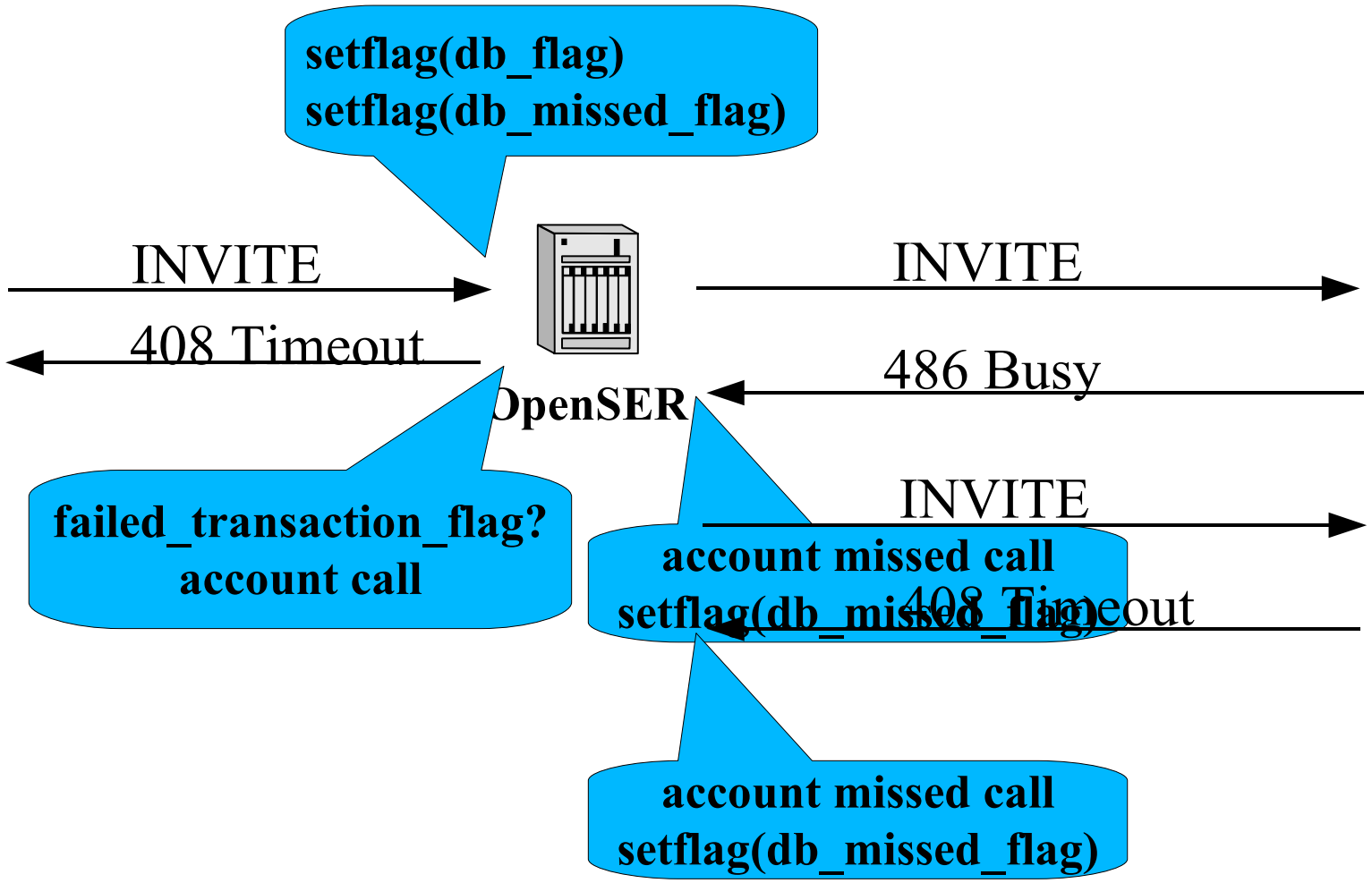
Accounting options...

- by setting the acc flag (**log|radius|db_flag**), only the successful transactions are accounted.
- to get also the failed transactions (completed with negative replies) you need to set the **failed_transaction_flag**
- if during serial forking you want to log some of the intermediary failed branches, you need to set the **log|radius|db_missed_flag**
  - this flag will log the next failed branch as a "FAILED" event
  - the flag will be automatically reset after the branch completion
  - IMPORTANT: this flag triggers accounting when a branch fails (on the client side), when failed_transaction_flag triggers accounting when the whole transaction fails (on the server side)
  - as the name says, it is useful for accounting the the "missed call" event during serial forking.

More accounting options:

- **early_media**
  - if enabled, an accounting event will be generated (START) when a 183 reply is received (and not only for 200 OK)
- **report_ack**
  - if enabled, an accounting event will be generated (START) when ACK is detected (for the transaction)
- **report_cancels**
  - if enabled, an accounting event will be generated (STOP) when CANCEL is detected (for the transaction)
- **detect_direction**
  - if the accounting records should be transparent for the SIP direction; useful for sequential requests where TO and FROM are swapped depending of who is sending the request.

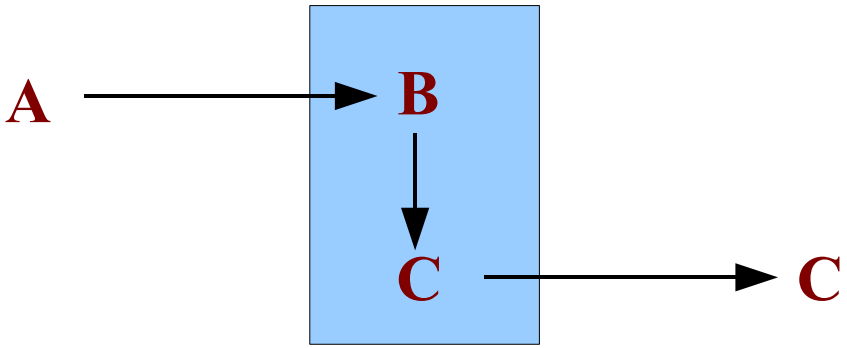

setflag(db_flag)
setflag(db_missed_flag)
INVITE
200 OK
OpenSER
INVITE
486 Busy
account call
INVITE
account missed call
setflag(db_missed_flag)
200 OK


setflag(db_flag)
setflag(db_missed_flag)
INVITE
408 Timeout
OpenSER
INVITE
486 Busy
failed_transaction_flag?
account call
INVITE
account missed call
setflag(db_missed_flag)
408 Timeout
account missed call
setflag(db_missed_flag)

Script example

Multi-leg accounting

- multi-leg accounting is about logging the “virtual” intermediary hops due call forward.
- normal accounting will contain only the first source and the last destination, but the middle hops.



A calls B who has set a redirect to C

- standard accounting ⇒ CDR (A,C)
- multi-leg accounting ⇒ CDR (A,B) and (B,C)

- additional to the default accounting information, set some additional information to be accounted per leg.
- use **multi_leg_info** module parameter to define the set of AVPs used for leg information

```
# for syslog-based accounting, use any text you want to be printed
modparam("acc", "multi_leg_info",
    "text1=$avp(src);text2=$avp(dst)")
# for mysql-based accounting, use the names of the columns
modparam("acc", "multi_leg_info",
    "leg_src=$avp(src);leg_dst=$avp(dst)")
# for RADIUS-based accounting, use the names of the RADIUS AVPs
modparam("acc", "multi_leg_info",
    "RAD_LEG_SRC=$avp(src);RAD_LEG_SRC=$avp(dst)")
```

- define the AVP set to contain the multi-leg specific info:
  (**A**, **B**, **C**) where logically **A**=source, **B**=destination, **C**=type

- for each leg (virtual in script), populate ALL the AVPs from the set. **IMPORTANT**: is it mandatory to set them all (even if there is not useful information for some of AVPs – otherwise the leg will be mixed.
  LEG1 (user1, user2, 0)
  LEG2 (user2, user3, 1)
  .....

- **HINT**: set the multi leg accounting only for INVITEs (or initial requests)
- **NOTE**: the legs will be accounted in reversed order!

The multi-leg information will be differently logged, depending on the accounting backend:

- **syslog**
  - as new attributes, repeating themselves
- **mysql**
  - each leg will be a separate CDR record – the CDRs will share the same non-leg information
  - there is the DB limitation as number of attributes a record can have
- **radius**
  - multiple RADIUS AVPs will be added to the RADIUS requests, corresponding to each leg

- **log**

ACC: transaction answered: timestamp=1190362043;method=INVITE;from_tag=327EF7EB;to_tag=b137d780e2b72679i0;call_id=1629788544@192.168.2.2;code=200;reason=OK;A=user2;B=user3;C=1;A=user1;B=user2;C=0

ACC: transaction answered: timestamp=1190362055;method=BYE;from_tag=327EF7EB;to_tag=b137d780e2b72679i0;call_id=1629788544@192.168.2.2;code=200;reason=OK

- **DB**

id	method	....	time	A	B	C
5	INVITE	.....	2007-04-30 20:31:36	user2	user3	1
6	INVITE	.....	2007-04-30 20:31:36	user1	user2	0
7	BYE	.....	2007-04-30 20:31:58	NULL	NULL	NULL

- **RADIUS**

Acct-Status-Type = Start
Service-Type = Sip-Session
Sip-Response-Code = 200
Sip-Method = INVITE
Event-Timestamp = 1142177361
Sip-To-Tag = "00D0E90101B8_T9513"
Sip-From-Tag = "111aa0fda452c726"
Acct-Session-Id = "1dbe198c8@192.168.0.11"
Sip-Src-IP = "192.168.0.11"
Sip-Src-Port = "5068"
NAS-IP-Address = 127.0.0.1
NAS-Port = 5060
Client-IP-Address = 10.10.10.10
Acct-Unique-Session-Id = "37fb00358437ff4d"
A = user2
B = user3
C = 1
A = user1
B = user2
C = 0

Acct-Status-Type = Stop
Service-Type = Sip-Session
Sip-Response-Code = 200
Sip-Method = BYE
Event-Timestamp = 1142177661
Sip-To-Tag = "00D0E90101B8_T9513"
Sip-From-Tag = "111aa0fda452c726"
Acct-Session-Id = "1dbe198c8@192.168.0.11"
Sip-Src-IP = "192.168.0.11"
Sip-Src-Port = "5068"
NAS-IP-Address = 127.0.0.1
NAS-Port = 5060
Client-IP-Address = 10.10.10.10
Acct-Unique-Session-Id = "37fb00358437ff4d"

Script example

Extra accounting

- along the static default information, ACC modules allows dynamical selection of extra information to be logged.
- this allows you to log any pseudo-variable (AVPs, parts of the request, etc)
- selection of extra information is done via xxx_extra parameters (specific for the accounting backedn) by specifying the names (labels) and the values for the additional data.

  label = value

```
modparam("acc", "log_extra",
        "ua=$hdr(User-Agent);uuid=$avp(i:123)")
```

- the label defines how/where the data will be logged; its meaning depends of the accounting support which is used

- label will be printed along with the data in ***label=data format;***

modparam("acc", "log_extra",
"ua=$hdr(User-Agent);uuid=$avp(i:123)")

⇒

ACC: transaction answered: timestamp=1190362043;method=
INVITE;from_tag=327EF7EB;to_tag=b137d780e2b72679i0;
call_id=1629788544@192.168.2.2;code=200;reason=OK;
acc=Grandstream;uuid=123423

- label is the name of the DB column where the data will be stored. **IMPORTANT**: add in db accounting tables the columns corresponding to each extra data

```
modparam("acc", "db_extra",
        "ua=$hdr(User-Agent);uuid=$avp(i:123)")
```

⇒

```
+----+-------+-----+-------------------------+----------------+----------+
| id  |method| .... | time                    |      ua        |  uuid    |
+----+-------+-----+-------------------------+----------------+----------+
|  5 | INVITE | ..... | 2007-04-30 20:31:36 | Grandstream | 123423 |
```

- label will be the RADIUS AVP name used for packing the data into RADIUS message. The label will be translated to AVP ID via the dictionary. **IMPORTANT**: add in RADIUS dictionary the label attribute.

```
modparam("acc", "radius_extra",
        "UA-ATTR=$hdr(User-Agent);UUID-ATTR=$avp(i:123)")
```

⇒

Acct-Status-Type = Start
Service-Type = Sip-Session
Sip-Response-Code = 200
Sip-Method = INVITE
Event-Timestamp = 1142177361
Sip-To-Tag = "00D0E90101B8_T9513"
Sip-From-Tag = "111aa0fda452c726"
Acct-Session-Id = "1dbe198c8@192.168.0.11"
Sip-Src-IP = "192.168.0.11"
Sip-Src-Port = "5068"
NAS-IP-Address = 127.0.0.1
NAS-Port = 5060
Client-IP-Address = 10.10.10.10
Acct-Unique-Session-Id = "37fb00358437ff4d"
UA-ATTR= Grandstream
UUID-ATTR = 123423

More on extra-accounting

- be sure when the information you want to account becomes available – it has to be before the accounting time (see accounting basics)
- note that the pseudo-variables related to message information take values from the request message (and not from replies); for this use AVPs in onreply_route
- some pseudo variables may return more than one value (like headers or AVPs); in this case, the returned values are embedded in a single string in a comma-separated format.
- extra accounting applies for all accounted events and types (normal or missed calls)
- extra accounting can be safely combined with multi-leg accounting