CDR

About

Call Detail Records are the data recorded during each call session. A CDR may contain attributes specific to each call session and eventually each leg of the call.

CDR contain the phone numbers originating the call and receiving the call, time of the call, call duration and many more attributes.

CDR Examples

An example of FreeSWITCH CDRs stored in /usr/local/freeswitch/log/cdr-csv/Master.csv

Sample CSV CDR

```
"9007","9007","0034688886392","public","2014-05-29 16:59:50","","2014-05-29 16:59:50","0","0","
NORMAL_CLEARING","a5c9f6c0-e752-11e3-8bfb-65b6c3cdac7d","","","","",""
"9008","9008","+34688886392","public","2014-05-29 17:01:32","","2014-05-29 17:01:32","0","0","NORMAL_CLEARING","
e29ff0ae-e752-11e3-8bff-65b6c3cdac7d","","","","",""
"9008","9008","34688886392","public","2014-05-29 17:01:33","","2014-05-29 17:01:33","0","0","NORMAL_CLEARING","
e2fe712e-e752-11e3-8c03-65b6c3cdac7d","","","","",""
"9008","9008","0034688886392","public","2014-05-29 17:01:33","","2014-05-29 17:01:33","0","0","
NORMAL_CLEARING","e34619ac-e752-11e3-8c07-65b6c3cdac7d","","","","",""
"9008","9008","00034688886392","public","2014-05-29 17:01:34","","2014-05-29 17:01:34","0","0","
NORMAL_CLEARING","e38cb5b0-e752-11e3-8c0b-65b6c3cdac7d","","","","",""
```

An example of detailed XML CDR stored in /usr/local/freeswitch/log/xml_cdr/a_80183ec8-d424-11e3-8fb2-65b6c3cdac7d.cdr.xml

Sample XML CDR

```
<?xml version="1.0"?>
<cdr core-uuid="b658d05e-c42c-11e3-bdcd-65b6c3cdac7d">
  <channel_data>
    <state>CS_REPORTING</state>
    <direction>inbound</direction>
    <state_number>11</state_number>
    <flags>0=1;37=1;39=1;73=1</flags>
    <caps>1=1;2=1;3=1;4=1;5=1;6=1</caps>
  </channel_data>
  <variables>
    <direction>inbound</direction>
    <uuid>80183ec8-d424-11e3-8fb2-65b6c3cdac7d</uuid>
    <session_id>1600</session_id>
    <sip_from_user>8009</sip_from_user>
    <sip_from_uri>8009%40XX.XX.XXX.XXX</sip_from_uri>
    <sip_from_host>XX.XX.XXX.XXX</sip_from_host>
    <channel_name>sofia/external/8009%40XX.XX.XXX.XXX</channel_name>
    <sip_call_id>8d8cccc68b962aa709a2c90d36e6165c</sip_call_id>
    <sip_local_network_addr>XX.XX.XXX.XXX</sip_local_network_addr>
    <sip_network_ip>XX.XX.XXX.XXX</sip_network_ip>
    <sip_network_port>5082</sip_network_port>
    <sip_received_ip>XX.XX.XXX.XXX</sip_received_ip>
    <sip_received_port>5082</sip_received_port>
    <sip_via_protocol>udp</sip_via_protocol>
    <sip_from_user_stripped>8009</sip_from_user_stripped>
    <sip_from_tag>1df994ea</sip_from_tag>
    <sofia_profile_name>external</sofia_profile_name>
    <recovery_profile_name>external</recovery_profile_name>
    <sip_full_via>SIP/2.0/UDP%20XX.XX.XXX.XXX%3A5082%3Bbranch%3Dz9hG4bK-8d8cccc68b962aa709a2c90d36e6165c%
3Brport%3D5082</sip_full_via>
    <sip_from_display>8009</sip_from_display>
    <sip_full_from>8009%20%3Csip%3A8009%40XX.XX.XXX.XXX%3E%3Btag%3D1df994ea</sip_full_from>
    <sip_to_display>003468888646444</sip_to_display>
    <sip_full_to>003468888646444%20%3Csip%3A003468888646444%40XX.XX.XXX.XXX%3E</sip_full_to>
```

```
    <sip_req_user>003468888646444</sip_req_user>
    <sip_req_port>5080</sip_req_port>
    <sip_req_uri>003468888646444%40XX.XX.XXX.XXX%3A5080</sip_req_uri>
    <sip_req_host>XX.XX.XXX.XXX</sip_req_host>
    <sip_to_user>003468888646444</sip_to_user>
    <sip_to_uri>003468888646444%40XX.XX.XXX.XXX</sip_to_uri>
    <sip_to_host>XX.XX.XXX.XXX</sip_to_host>
    <sip_contact_user>8009</sip_contact_user>
    <sip_contact_port>5082</sip_contact_port>
    <sip_contact_uri>8009%40XX.XX.XXX.XXX%3A5082</sip_contact_uri>
    <sip_contact_host>XX.XX.XXX.XXX</sip_contact_host>
    <rtp_use_codec_string>G722,PCMU,PCMA,GSM</rtp_use_codec_string>
    <sip_user_agent>sipcli/v1.8</sip_user_agent>
    <sip_via_host>XX.XX.XXX.XXX</sip_via_host>
    <sip_via_port>5082</sip_via_port>
    <sip_via_rport>5082</sip_via_rport>
    <max_forwards>70</max_forwards>
    <switch_r_sdp>v%3D0%0D%0Ao%3Dsipcli-Session%201224945619%201785273517%20IN%20IP4%20XX.XX.XXX.XXX%0D%0As%
3Dsipcli%0D%0Ac%3DIN%20IP4%20XX.XX.XXX.XXX%0D%0At%3D0%200%0D%0Am%3Daudio%205083%20RTP/AVP%2018%200%208%20101%0D%
0Aa%3Drtpmap%3A18%20G729/8000%0D%0Aa%3Drtpmap%3A0%20PCMU/8000%0D%0Aa%3Drtpmap%3A8%20PCMA/8000%0D%0Aa%3Drtpmap%
3A101%20telephone-event/8000%0D%0Aa%3Dfmtp%3A101%200-15%0D%0Aa%3Dptime%3A20%0D%0A</switch_r_sdp>
    <ep_codec_string>PCMU%408000h%4020i%4064000b,PCMA%408000h%4020i%4064000b</ep_codec_string>
    <endpoint_disposition>DELAYED%20NEGOTIATION</endpoint_disposition>
    <call_uuid>80183ec8-d424-11e3-8fb2-65b6c3cdac7d</call_uuid>
    <outside_call>true</outside_call>
    <current_application_data>RFC2822_DATE%3DMon,%2005%20May%202014%2007%3A11%3A38%20%2B0000<
/current_application_data>
    <current_application>export</current_application>
    <RFC2822_DATE>Mon,%2005%20May%202014%2007%3A11%3A38%20%2B0000</RFC2822_DATE>
    <export_vars>RFC2822_DATE</export_vars>
    <hangup_cause>NORMAL_CLEARING</hangup_cause>
    <hangup_cause_q850>16</hangup_cause_q850>
    <digits_dialed>none</digits_dialed>
    <start_stamp>2014-05-05%2007%3A11%3A38</start_stamp>
    <profile_start_stamp>2014-05-05%2007%3A11%3A38</profile_start_stamp>
    <end_stamp>2014-05-05%2007%3A11%3A38</end_stamp>
    <start_epoch>1399273898</start_epoch>
    <start_uepoch>1399273898421766</start_uepoch>
    <profile_start_epoch>1399273898</profile_start_epoch>
    <profile_start_uepoch>1399273898421766</profile_start_uepoch>
    <answer_epoch>0</answer_epoch>
    <answer_uepoch>0</answer_uepoch>
    <bridge_epoch>0</bridge_epoch>
    <bridge_uepoch>0</bridge_uepoch>
    <last_hold_epoch>0</last_hold_epoch>
    <last_hold_uepoch>0</last_hold_uepoch>
    <hold_accum_seconds>0</hold_accum_seconds>
    <hold_accum_usec>0</hold_accum_usec>
    <hold_accum_ms>0</hold_accum_ms>
    <resurrect_epoch>0</resurrect_epoch>
    <resurrect_uepoch>0</resurrect_uepoch>
    <progress_epoch>0</progress_epoch>
    <progress_uepoch>0</progress_uepoch>
    <progress_media_epoch>0</progress_media_epoch>
    <progress_media_uepoch>0</progress_media_uepoch>
    <end_epoch>1399273898</end_epoch>
    <end_uepoch>1399273898421766</end_uepoch>
    <last_app>export</last_app>
    <last_arg>RFC2822_DATE%3DMon,%2005%20May%202014%2007%3A11%3A38%20%2B0000</last_arg>
    <caller_id>%228009%22%20%3C8009%3E</caller_id>
    <duration>0</duration>
    <billsec>0</billsec>
    <progresssec>0</progresssec>
    <answersec>0</answersec>
    <waitsec>0</waitsec>
    <progress_mediasec>0</progress_mediasec>
    <flow_billsec>0</flow_billsec>
    <mduration>0</mduration>
    <billmsec>0</billmsec>
    <progressmsec>0</progressmsec>
    <answermsec>0</answermsec>
```

```
    <waitmsec>0</waitmsec>
    <progress_mediamsec>0</progress_mediamsec>
    <flow_billmsec>0</flow_billmsec>
    <uduration>0</uduration>
    <billusec>0</billusec>
    <progressusec>0</progressusec>
    <answerusec>0</answerusec>
    <waitusec>0</waitusec>
    <progress_mediausec>0</progress_mediausec>
    <flow_billusec>0</flow_billusec>
    <sip_hangup_disposition>send_refuse</sip_hangup_disposition>
  </variables>
  <app_log>
    <application app_name="set" app_data="outside_call=true" app_stamp="1399273898425694"></application>
    <application app_name="export" app_data="RFC2822_DATE=Mon, 05 May 2014 07:11:38 +0000" app_stamp="
1399273898426099"></application>
  </app_log>
  <callflow dialplan="XML" unique-id="801853d6-d424-11e3-8fb3-65b6c3cdac7d" profile_index="1">
    <extension name="outside_call" number="003468888646444">
      <application app_name="set" app_data="outside_call=true"></application>
      <application app_name="export" app_data="RFC2822_DATE=${strftime(%a, %d %b %Y %T %z)}"></application>
    </extension>
    <caller_profile>
      <username>8009</username>
      <dialplan>XML</dialplan>
      <caller_id_name>8009</caller_id_name>
      <caller_id_number>8009</caller_id_number>
      <callee_id_name></callee_id_name>
      <callee_id_number></callee_id_number>
      <ani>8009</ani>
      <aniii></aniii>
      <network_addr>XX.XX.XXX.XXX</network_addr>
      <rdnis></rdnis>
      <destination_number>003468888646444</destination_number>
      <uuid>80183ec8-d424-11e3-8fb2-65b6c3cdac7d</uuid>
      <source>mod_sofia</source>
      <context>public</context>
      <chan_name>sofia/external/8009@XX.XX.XXX.XXX</chan_name>
    </caller_profile>
    <times>
      <created_time>1399273898421766</created_time>
      <profile_created_time>1399273898421766</profile_created_time>
      <progress_time>0</progress_time>
      <progress_media_time>0</progress_media_time>
      <answered_time>0</answered_time>
      <bridged_time>0</bridged_time>
      <last_hold_time>0</last_hold_time>
      <hold_accum_time>0</hold_accum_time>
      <hangup_time>1399273898421766</hangup_time>
      <resurrect_time>0</resurrect_time>
      <transfer_time>0</transfer_time>
    </times>
  </callflow>
</cdr>
```

Store CDR

There are several ways which FreeSWITCH can save CDR (Call Detail Record).

The best way to store "live" call detail records is to write all the data fields to a temporary area on disk or RAM drive and write a script to scan that same area of the file system for the long-running processing of storing them in your database. This avoids hanging the voice call thread in FS if the http or db server is down at the end of the calls.

DO NOT write CDR scripts in the hangup hook of your dialplan or ESL script as this will delay the termination of the voice thread and will not scale to large systems. Allow the voice thread to handle only voice; handle your back-end business processes separately, off-line. This approach works for all installations from small to huge.

Popular ways include:

- **mod_cdr_csv** - saves a CSV file with the variables you specify in a template.
- **mod_cdr_mongodb** - saves detailed CDR data to a MongoDB database, in a format similar to mod_json_cdr.
- **mod_odbc_cdr** - saves any channel variable from call to your choice of ODBC database.
- **mod_cdr_pg_csv** - Asterisk Compatible CDR Module with PostgreSQL interface.
- **mod_cdr_sqlite** - saves directly to an sqlite DB with the variables you specify in a template.
- **mod_json_cdr** - Saves to file or POSTs a JSON representation of the channel variable and callflow. It can post directly to CouchDB.
- **mod_xml_cdr** - Saves to file or POSTs an XML representation of the channel variable and callflow.
    - Xmlcdrd or JavaCDRLogger may assist you in saving the results.
- **mod_radius_cdr** - RADIUS CDR Module.

- **CDR via ESL** - Retrieve the CDR in an ESL event

- **tiny_cdr** from bougyman at https://github.com/rubyists/tiny_cdr

CDR Handling

You can instruct FreeSWITCH to not log a call, or only log leg B or the like. See: Variable_process_cdr

You can also specify hangup causes that should not generate a CDR. See Variable_skip_cdr_causes (added V1.2.3 3cf238fc)

Calculating Various Time Values For A Call

FreeSWITCH CDRs contain lots of information. From those values one can extract other information:

What time a call started ringing (PDD)

When a call starts ringing it either has media or does not have media. If it has media then **progress_media_time** will be set; if not, **progress_time** will be set. The value not set will be zero. Therefore this formula will always yield the exact time the call started to ring:

Calculating Ring Start

```
ring_start_time = progress_time + progress_media_time
```

The PDD (post-dial delay) is the period of silence between the call starting and the call ringing, therefore for calls that ring:

Post-Dial Delay

```
pdd = ring_start_time - created_time
```

How long was the talk time

If the call was answered then there is a talk time. If call was unanswered then **answered_time** will be 0. If **answered_time** is greater than zero:

Calculating Call Duration

```
length_of_talk_time = hangup_time - answered_time
```

How long did the phone ring

How long the phone rang is determined either by the **hangup_time** or the **answered_time**:

Calculating Ring Duration

```
if ( answered_time == 0 ) then
    length_of_phone_ringing = hangup_time - created_time
else
    length_of_phone_ringing = answered_time - created_time
end
```

How long was the call on hold

As of September 13th, 2012, FreeSWITCH also provides the **hold_events** field. This field is structured as a flattened, multi-dimensional array of uepoch values representing the start and stop time of hold events for the channel. For example: {{1347907323618493,1347907328495937},{1347907309458486,1347907314655415},{1347907298602214,1347907304095908},{1347907285118780,1347907291355494}}

It is structured in native PostgreSQL array text format so it is very easy to work with in PostgreSQL. Here is an example showing how to work with such a field in PostgreSQL:

Calculating Hold Time

```
CREATE OR REPLACE FUNCTION test_hold() RETURNS VOID AS $$
DECLARE
        hold_recs BIGINT[][];
        tmp RECORD;
BEGIN
        hold_recs := '{{1347907323618493,1347907328495937},{1347907309458486,1347907314655415},
{1347907298602214,1347907304095908},{1347907285118780,1347907291355494}}';

        FOR tmp IN
                WITH uepochs AS (
                        SELECT unnest((SELECT hr1[array_lower(hold_recs, 1):array_upper(hold_recs, 1)][1:1]
FROM (SELECT hold_recs::BIGINT[] AS hr1) AS ss)) AS start_time,
                               unnest((SELECT hr1[array_lower(hold_recs, 1):array_upper(hold_recs, 1)][2:2]
FROM (SELECT hold_recs::BIGINT[] AS hr1) AS ss)) AS stop_time
                )
                SELECT (TIMESTAMP WITH TIME ZONE 'epoch' + start_time * INTERVAL '1 microsecond') AS
start_time, (TIMESTAMP WITH TIME ZONE 'epoch' + stop_time * INTERVAL '1 microsecond') AS stop_time FROM uepochs
        LOOP
                RAISE NOTICE 'Start time: %; Stop time: %', tmp.start_time, tmp.stop_time;
        END LOOP;
END;
$$
LANGUAGE plpgsql;

SELECT * FROM test_hold();
```

The "WITH" clause converts the multi-dimensional array into a set of row result records with two columns: start_time and stop_time. The "SELECT" clause below the "WITH" clause converts the resulting micro-second Epoch timestamps into TIMESTAMP WITH TIME ZONE types and returns the resulting set of rows. The FOR loop assigns each row in turn to the RECORD variable "tmp," which is used in the RAISE NOTICE line to print out the resulting values. This allows you to treat the **hold_events** field as a TEXT type in the database and convert it into a record-based result-set on demand anywhere you need to iterate through the hold records.

Putting Together CDRs

If you are saving both A and B leg for CDRs, you'll need to piece them together.

1) The self-uuid is stored in "uuid"

2) A-legs - it's an A-leg if:

- "originatee" or "origination" or "originate_disposition" is set. (NOTE: The original channel that triggered a loopback shows up as B leg in xml_cdr)

3) B-legs - it's a B-leg if:

- "ent_originate_aleg_uuid" is set. "ent_originate_aleg_uuid" contains the UUID of the A leg.
- "originator" or "originating_leg_uuid" is set. Both contain the UUID of the A leg.

4) What's the connecting leg?

- For enterprise originates: on B-leg, use ent_originate_aleg_uuid.
- "bridge_uuid" (not in bypass mode)
- "signal_bond"
- "last_bridge_to"
- In B-leg: originator or originating_leg_uuid
- In A-leg: Parse originated_legs or originate_causes (added 2012-10-18, commit 3099445a95933a52954c64d5f2fd314a55577c9d)