# mod_lcr

## About

mod_lcr implements LCR (Least Cost Routing) for FreeSWITCH. You can use multiple profiles - that sort using different methods, e.g. quality rather than price.

It can be called in many ways: transferred to in the dial plan, bridged to, save the results and bridge to it, or invoked from the command line or ESL.

It's best to store the lcr table in E.164 format, without any 00 or 011 prefix. If you need that for a particular carrier, you can set the prefix field for that carrier.

Therefore, NANP numbers would be 11 digits (12145551212 not 2145551212) and India would be 91 not 01191.

NOTE: Put theory of operation here. Basically, order by longest digits and then by cost. Return a properly formatted dialstring.

For more information about mod_lcr check out this discussion by Rupa Schomaker:

- MPEG
- Ogg

## Usage Summary

CLI:

```
lcr <digits> [<lcr profile>] [caller_id] [intrastate] [as xml]
e.g. lcr 12145551111
```

Bridge

```
lcr/[$profile/]$number (optional $profile added as of b70ac96 2013-02-28, but uses default profile if empty.)
e.g. <action application="bridge" data="lcr/[$profile/]$number"/>
```

Application (profile optional)

```
<action application="lcr" data="$1 [$profile]"/>
<action application="bridge" data="${lcr_auto_route}"/>
```

Transfer

```
<action application="transfer" data="$1 lcr"/> (profile?)
```

## Requirements

mod_lcr uses FreeSWITCH ODBC core functions to read from your database in real time. You need to have that support before trying to install this module. For more information, see Using ODBC in the core.

## Installing

To use LCR:

Tell FreeSWITCH to compile in this module by editing modules.conf in /usr/src/freeswitch and uncomment:

```
#applications/mod_lcr
```

Now go recompile FreeSWITCH.

```
make
make install
```

Or compile individually:

```
make mod_lcr-install
```

Tell FreeSWITCH to actually use the LCR module when running by adding the module to modules.conf.xml in /usr/local/freeswitch/conf/autoload_configs:

```
<load module="mod_lcr"/>
```

If FreeSWITCH is already running, you can load it now:

```
load mod_lcr
```

Edit the default configs for your profiles and database connection information

```
 /usr/local/freeswitch/conf/autoload_configs/lcr.conf.xml
```

Note: When you load/reload mod_lcr, there might be an error which is below. Don't worry about this it is just trying to figure out if it should use rand() or random() with the database this is normal since there are differences between mysql and postgresql.

```
2010-09-23 13:23:50.314665 [ERR] switch_odbc.c:427 ERR: [SELECT rand();]
[STATE: 01000 CODE 4294967295 ERROR: Error while executing the query (non-fatal);
ERROR:  function rand() does not exist at character 8]
```

# Usage

## CLI / API

USAGE: lcr <digits> [<lcr profile>] [caller_id] [intrastate] [as xml]

From the command line or ESL issue something similar to:

```
lcr 12145551111
```

Which would respond with something like:

```
API CALL [lcr(12145551111)] output:
| Digit Match | Carrier  | Rate     | Dialstring                          |
| 1214        | carrier1 | 0.01000  | sofia/gateway/carrier1/12145551111 |
| 1           | carrier2 | 0.01440  | sofia/gateway/carrier2/12145551111 |
```

The "as xml" output is suitable for use from a event socket application.

## Dialplan, Endpoint Transfer

To use as a dialplan, just transfer to:

```
<action application="transfer" data="$1 lcr"/>
```

## Dialplan, Bridge

Use the format:

```
lcr/[$profile/]$number (optional $profile added as of b70ac96 2013-02-28 - uses default profile if empty.)
<action application="bridge" data="lcr/[$profile/]$number"/>
```

## Dialplan Application

The LCR application can be called from the dialplan by executing it from within your condition as follows:

```
<action application="lcr" data="$1 [profile]"/>
```

It will return a value suitable for use with the bridge application in the variable ${lcr_auto_route}.

It will also populate the variables lcr_route_N (where N is 1 to route count) and lcr_route_count for more flexible use of the lcr information in scripts.

```
<extension name="Outbound to PSTN 11 Digits">
  <condition field="destination_number" expression="^(1[2-9][0-9]{2}[2-9]{7})$">
    <action application="lcr" data="$1 [profile]"/>
    <action application="bridge" data="${lcr_auto_route}"/>
  </condition>
</extension>
```

## Channel Variables / CDR Logs

LCR will create some channel variables that document which route was actually used to complete the call. **lcr_carrier** contains the carrier name and **lcr_rate** contains the rate.

These variables show up automatically on the a and b legs for the XML CDRs. You can include them in your CSV CDRs by editing the template.

In addition the following channel variables are set on the a leg:

```
lcr_query_digits: digits passed to lcr
lcr_query_profile: profile id used by lcr
lcr_query_expanded_digits: expanded version of digits suitable for use in an IN list
```

## Configuration

- Ensure FreeSWITCH is compiled with ODBC support.
- Ensure your system is setup for ODBC support.
- Ensure your odbcinst.ini and odbc.ini file is in /etc it doesn't matter what your OS is mod_lcr only looks in /etc
- Import the appropriate database schema for your database. (PostgreSQL and MySQL schema included) -> contrib/intralanman/C/lcr/sql or /src /mod/applications/mod_lcr/sql
- Load sample data.
- Modify BASE/conf/autoload_configs/lcr.conf.xml to reflect your DSN.
- Modify BASE/conf/modules.conf.xml to load mod_lcr.
- Try it out from the CLI.

## Configuring LCR tables

- Someone come up with a layout that works here:

```
Table: carriers
Purpose: Defines your carriers
Field: carrier_name - name of the carrier
Field: enabled - whether the carrier (thus all it's gateways/lcr entries) are enabled

Table: carrier_gateway
Purpose: Defines gateway information for a given carrier
Field: carrier_id - maps to carrier
Field: prefix - the value to put before the phone number after any translation
Field: suffix - the value to put after the phone number after any translation
Field: codec - codec to use for absolute_codec_string.  Leave empty/null for default.
Field: enabled - whether the gateway (thus all it's lcr entries) are enabled

Table: lcr
Purpose: Defines rules for a given digit sequence
Field: digits - matching digits
Field: rate - rate
Field: intrastate_rate - rate for intrastate calls
Field: intralata_rate - rate for intralata calls
Field: carrier_id - which carrier for this entry
Field: lead_strip - how many digits to strip off front of passed in number
Field: trail_strip - how many digits to strip of end of passed in number
Field: prefix - value to add to front of passed in number
Field: suffix - vaulue to add to end of passed in number
Field: lcr_profile - profile_id
Field: date_start - when this LCR entry becomes valid
Field: date_end - when this LCR entry becomes invalid
Field: quality - alternate field to order by
Field: reliability - alternate field to order by
Field: cid - regular expression to modify the callers caller id number - channel variables are also valid when
called from the dial plan
Field: enabled - true/false - whether this LCR entry is enabled
```

## Sample Data

### Sample data for MySQL

```
-- insert two carriers

INSERT INTO carriers (id, carrier_name, enabled) VALUES (1, 'carrier1', 1);
INSERT INTO carriers (id, carrier_name, enabled) VALUES (2, 'carrier2', 1);

-- insert some gateway info

INSERT INTO carrier_gateway (id, carrier_id, prefix, suffix) VALUES (1, 1, 'sofia/gateway/carrier1/', '');
INSERT INTO carrier_gateway (id, carrier_id, prefix, suffix) VALUES (2, 2, 'sofia/external/', '@proxy.carrier2.
net:5060');

-- insert some lcr data

INSERT INTO lcr (id, digits, rate, carrier_id, lead_strip, trail_strip, prefix, suffix, date_start, date_end,
quality, reliability)
 VALUES (1, '1', 0.15, 1, 0, 0, '', '', current_timestamp - interval 1 year, current_timestamp + interval 1 year
, 0, 0);
INSERT INTO lcr (id, digits, rate, carrier_id, lead_strip, trail_strip, prefix, suffix, date_start, date_end,
quality, reliability)
 VALUES (2, '1', 0.12, 2, 1, 0, '0', '', current_timestamp - interval 1 year, current_timestamp + interval 1 year
, 0, 0);
INSERT INTO lcr (id, digits, rate, carrier_id, lead_strip, trail_strip, prefix, suffix, date_start, date_end,
quality, reliability)
 VALUES (3, '1234', 0.05, 1, 0, 0, '', '', current_timestamp - interval 1 year, current_timestamp + interval 1
year , 0, 0);
INSERT INTO lcr (id, digits, rate, carrier_id, lead_strip, trail_strip, prefix, suffix, date_start, date_end,
quality, reliability)
 VALUES (4, '1234', 0.02, 2, 1, 0, '0', '', current_timestamp - interval 1 year, current_timestamp + interval 1
year , 0, 0);
```

## Sample data for PostgreSQL

First import the schema and data (since we rely on sequences, drop your existing tables / sequences)

```
$ psql phone -f postgres-8.3.sql
$ psql phone -f sample_data.sql
```

Contents of sample_data.sql:

```
-- insert two carriers

INSERT INTO carriers (carrier_name) VALUES ('carrier1');
INSERT INTO carriers (carrier_name) VALUES ('carrier2');

-- insert some gateway info

INSERT INTO carrier_gateway (carrier_id, prefix, suffix) VALUES
  (1, 'sofia/gateway/carrier1/', '');
INSERT INTO carrier_gateway (carrier_id, prefix, suffix) VALUES
  (2, 'sofia/external/', '@proxy.carrier2.net:5060');

-- insert some lcr data

INSERT INTO lcr (digits, rate, carrier_id, lead_strip, trail_strip,
                 prefix, suffix,
                 date_start, date_end, quality, reliability) VALUES
       ('1', 0.15, 1, 0, 0, '', '',
        current_timestamp - interval 1 year,
        current_timestamp + interval 1 year
        , 0, 0);
INSERT INTO lcr (digits, rate, carrier_id, lead_strip, trail_strip,
                 prefix, suffix,
                 date_start, date_end, quality, reliability) VALUES
       ('1', 0.12, 2, 1, 0, '0', '',
        current_timestamp - interval 1 year,
        current_timestamp + interval 1 year
        , 0, 0);
INSERT INTO lcr (digits, rate, carrier_id, lead_strip, trail_strip,
                 prefix, suffix,
                 date_start, date_end, quality, reliability) VALUES
       ('1234', 0.05, 1, 0, 0, '', '',
        current_timestamp - interval 1 year,
        current_timestamp + interval 1 year
        , 0, 0);
INSERT INTO lcr (digits, rate, carrier_id, lead_strip, trail_strip,
                 prefix, suffix,
                 date_start, date_end, quality, reliability) VALUES
       ('1234', 0.02, 2, 1, 0, '0', '',
        current_timestamp - interval 1 year,
        current_timestamp + interval 1 year
        , 0, 0);
```

Now run some lookups:

Number that matches only on first digit. You can see that for carrier 2, the first number is stripped and prepended with 0 based on the LCR rules.

```
API CALL [lcr(12)] output:
 | Digit Match | Carrier  | Rate     | Dialstring                                |
 | 1           | carrier2 | 0.12000  | sofia/external/02@proxy.carrier2.net:5060 |
 | 1           | carrier1 | 0.15000  | sofia/gateway/carrier1/12                 |
```

Number that matches more specific. Again, same rule applies for carrier2.

```
API CALL [lcr(12345678)] output:
 | Digit Match | Carrier  | Rate     | Dialstring                                      |
 | 1234        | carrier2 | 0.02000  | sofia/external/02345678@proxy.carrier2.net:5060 |
 | 1234        | carrier1 | 0.05000  | sofia/gateway/carrier1/12345678                 |
```

Finally, a number that doesn't match:

```
API CALL [lcr(987)] output:
No Routes To Display
```

# Advanced Usage

## Profiles

**mod_lcr** also has the concept of a profile. This allows one to have alternate rule sets for different customers. Profiles can also be used to modify the field (s) that are used to order the results from the database lookup.

To reload changes to the config, do the following:

```
reloadxml
reload mod_lcr
```

CLI:

```
lcr number profile_name
lcr_admin show profiles
```

Or dialplan:

```
      <action application="lcr" data="$1 profile"/>
```

Parameters

- order_by - controls the ordering of results.
- id - additional parameter to limit query to record with id specified.
- reorder_by_rate - Forces the LCR module to re-order the query strictly on rate basis. By default this is turned off, but enabling this will always prefer rate over anything else.
- quote_in_list - will quote the prefixes in the IN() list passed to the database. May be necessary for MySQL.
- enable_sip_redir - Modifies the default behavior so that ${lcr_auto_route} is suitable for use with Misc._Dialplan_Tools_redirect.

Profiles are defined in the lcr.conf.xml file.

For discussion, consider the following:

```
  <profiles>
    <profile name="rate">
      <param name="order_by" value="rate"/>
    </profile>
    <profile name="quality">
      <param name="order_by" value="quality"/>
    </profile>
    <profile name="reliability">
      <param name="order_by" value="reliability"/>
    </profile>
    <profile name="foo">
      <param name="id" value="2"/>
      <param name="order_by" value="rate"/>
    </profile>
    <profile name=reorder_rate">
      <param name="reorder_by_rate" value="true"/> <!-- default false -->
    </profile>
  </profiles>
```

This defines five profiles. The first three are the default shipped. They return records ordered by rate, quality, and reliability respectively. The Fourth defines a profile that orders by rate but also selects lcr records that have a profile id of 2.

**NOTE:** The `lcr_profile` column corresponds to the value of the id parameter for the profile name defined in lcr.conf.xml

i.e.

```
<param name="id" value="200"/>
```

The fifth defines the parameter "reorder_by_rate" to true which will reorder the list ONLY by rate. **Warning:** I would generally recommend against this. All LCR modules I know of always use a primary sort order of the prefix.

mod_lcr is table driven. The rules for populating rate, quality, and reliability are up to you in your data load / maintenance scripts. The order_by parameter is really any valid sql clause in the order_by clause and is inserted after "ORDER BY digits" in the sql. "reliability" and "quality" are special in that they are recognized by the module and ordered in descending order automatically.

## Custom SQL

If the above does not give you enough flexibility, then you can also define a custom sql statement to do "whatever you want". But, you must keep in mind the following:

- The sql **must** return the following fields with the proper field name. Use SQL "AS" syntax to rename your existing field names to the appropriate ones.
- The sql **may** have %q where you want the dialed number to be inserted into the query.
- The sql **may** use channel variables.

The SQL is supplied on a per-profile basis using the custom-sql parameter:

```
   <profile name="rate">
     <param name="custom_sql" value="mysql goes %q here;"/>
   </profile>
```

or,

```
   <profile name="rate">
     <param name="custom_sql" value="mysql goes here where prefix IN (${lcr_query_expanded_digits});"/>
   </profile>
```

Here is another sample

```
    <profile name="test">
      <param name="order_by" value="rate"/>
      <param name="reorder_by_rate" value="true"/>
      <param name="export_fields" value="gw_extra_field"/>
      <param name="custom_sql" value="
SELECT l.digits AS lcr_digits,
       c.carrier_name AS lcr_carrier_name,
       l.${lcr_rate_field} as lcr_rate_field,
       cg.prefix AS lcr_gw_prefix, cg.suffix AS lcr_gw_suffix,
       l.lead_strip AS lcr_lead_strip, l.trail_strip AS lcr_trail_strip,
       l.prefix AS lcr_prefix, l.suffix AS lcr_suffix, 'PCMU' as
       gw_extra_field
  FROM lcr l
    JOIN carriers c ON l.carrier_id=c.id
    JOIN carrier_gateway cg ON c.id=cg.carrier_id
  WHERE c.enabled = '1' AND cg.enabled = '1' AND l.enabled = '1'
    AND digits IN (${lcr_query_expanded_digits})
    AND CURRENT_TIMESTAMP BETWEEN date_start AND date_end
  ORDER BY digits DESC, ${lcr_rate_field}, random();
      "/>

lcr_digits - the prefix that matched
lcr_carrier_name - name of the carrier
lcr_rate_field - rate
lcr_gw_prefix - data to prepend to the dial string
lcr_gw_suffix - data to append to the dial string
lcr_lead_strip - number of chars to remove from the left of the dialed #
lcr_trail_strip - number of chars to remove from the right of the dialed #
lcr_prefix - data to prepend to the dialed number
lcr_suffix - data to append to the dialed number
lcr_codec - codec to use for aboslute_codec_string - optional
lcr_cid - the regular expression used to modify the user's caller_id_number - optional
lcr_user_rate - the end-user rate for the call (refer to user rate section below) - optional
lcr_limit_realm - realm to use if using limit - optional
lcr_limit_id - id to use if using limit - optional
lcr_limit_max - max limit if using limit - optional
```

The dialstring is composed of:

lcr_gw_prefix lcr_prefix dialed_number (after strip processing) lcr_suffix lcr_gw_suffix

## Custom SQL with sub-query - for real-life ratesheet complexities

Rates rates can often be given both on per-NPA or per-NPANXX level depending on the carrier and on the NPA. Moreover, some carriers may have NPANXX rate lower than the corresponding NPA rate, while others will have it inverse. Proof of this is beyond WiKi article, but the fact is that neither simple ORDER BY rate, prefix; nor ORDER BY prefix, rate; give the truly cheapest route. The LCR logic should be two-step process to accomodate this:

- For each carrier, select only one prefix (longest prefix = most specific match)
- Select routes and rates for each of the pre-selected prefixes and sort results in the order you like.

SQL query below works for me on PostgreSQL. It works better that the sorting logic behind mod_lcr or behind simple custom SQL queries.

```
    <!-- This SQL is tricky. The lcr table we select from repeats twice. Don't miss if ever changing. -->
    <profile name="test">
      <param name="custom_sql" value="SELECT
       l.digits_prefix AS lcr_digits,
       c.carrier_name AS lcr_carrier_name,
       l.rate AS lcr_rate_field,
       cg.prefix AS lcr_gw_prefix,
       cg.suffix AS lcr_gw_suffix,
       l.lead_strip AS lcr_lead_strip,
       l.trail_strip AS lcr_trail_strip,
       l.prefix AS lcr_prefix,
       l.suffix AS lcr_suffix
       FROM lcr l JOIN
                                (SELECT carrier_id, max (digits_integer) digits_integer
                                FROM lcr
                                WHERE digits_prefix @> '%q'
                                GROUP BY carrier_id) tmp on l.digits_integer = tmp.digits_integer and l.
carrier_id = tmp.carrier_id
                             JOIN carriers c ON l.carrier_id=c.id JOIN carrier_gateway cg ON c.id=cg.carrier_id
       WHERE c.enabled = 'TRUE' AND cg.enabled = 'TRUE' AND l.enabled = 'TRUE'
       ORDER BY l.rate, random();"/>
    </profile>
```

Notes:

- Table lcr has digits_prefix (prefix_range) and digits_integer (bigint). Originally, I had everythign as prefix_range.....however aggregate funtion max() doesn't work with prefix_range type, that's why this colum has to be duplictaed as bigint
- SQL sub-query and GROUP_BY are needed to solve "greatest-n-per-group": http://stackoverflow.com/questions/tagged/greatest-n-per-group
- This query is just an example. It has some parts of WHERE clause removed as I don't need them. I don't select a bunch of stuff I don't need.

## Adding extra channel variables

It is possible to add extra variables to the dial string, such as absolute_codec_string.

- Add the extra field to the appropriate table. Each field should have only one key/value pair. Your job is to determine where the field goes based on the specificity of the data. eg: is it something that varies per prefix? Per carrier?
- Add the data in the field.
- Change the custom_sql param to include the field in your sql. You can alias the field to whatever you want the channel variable to be named.
- Change the export_fields param to include the extra field(s) you want to be included.

If you are having problems with this make sure you are using the svn or git version of freeswitch.

Refer to the nibblebill section of this wiki for a concrete example of how to do this.

## Integrating with Limit

mod_lcr can set a Limit on a per-route basis.

You can use limit in the dialplan before bridging to LCR to limit concurrent or CPS per user.

However, to limit per carrier, that can't be done before the bridge or even in the dialplan, but must be done special by mod_lcr.

- To use limit per carrier, this **requires** that one transfers, bridges, or execute_extension to the lcr/ end-point.

```
<action application="transfer" data="$1 lcr any_custom_profile"/> OR
<action application="bridge" data="lcr/$1"/> (You couldn't pass in a profile to bridge)
<action application="bridge" data="lcr/$profile/$1"/> -- works as of b70ac96 2013-02-28, but uses default if
empty.
```

It also **requires** that one uses custom_sql. Limit policy is up to the user and it is not clear whether the limit applies at the gateway or lcr level so instead I leave it to the user to configure based on custom_sql.

- Add to custom_sql:

```
'carriers' AS lcr_limit_realm, c.carrier_name AS lcr_limit_id, 5 AS lcr_limit_max
```

This hard-codes the realm to the string carriers, the id to the actual carrier name and the limit to 5. Both of those could just as easily be pulled from the database as well.

The three variable that must exist:

- - lcr_limit_realm for the broad hash category
  - lcr_limit_id for each group that gets limited
  - lcr_limit_max for the max of that group.
- Define which limit backend to use, in lcr.conf.xml per LCR profile. (As of 2012-10-22 there seems to be a bug making it impossible to use the DB backend, see FS-4761)

```
<param name="limit_type" value="hash"/>
```

## Integrating with mod_nibblebill

mod_lcr supports b-leg nibblebill usage. This allows one to lookup the user-rate and use it in your routes without additional dialplan scripting. To configure nibblebill one **must** use custom_sql. Again, this is because there is no clear policy for defining how a user_rate should be determined. If it can be expressed in SQL (including perhaps the use of a stored procedure, view, etc) then it can be retrieved by mod_lcr and passed to mod_nibblebill.

To setup:

- Modify your custom_sql. In my case, I added the following:

```
      a.account_code AS nibble_account,
      l.${lcr_rate_field} * 2 AS nibble_rate,
      a.warn_limit AS lowbal_amt,
      a.nobal_limit AS nobal_amt
 ...

   JOIN accounts a ON a.account_code = '${account_code}'
```

2) Add the parameter "export_fields" to add them to the b-leg channel vars:

```
      <param name="export_fields" value="nibble_account,nibble_rate,lowbal_amt,nobal_amt"/>
```

## PostgreSQL and contrib prefix

There is a third-party module available for PostgreSQL 8.1+ called prefix which provides a custom, GiST indexable column type for prefix matching. This module makes it possible to query a rate table with hundreds of thousands of entries in less than a millisecond.

Edit (10-05-2010): I haven't tested this personally but I read a blog that said PostgreSQL 9.0 was compatible with prefix and mod_lcr.

Example usage:

```
create table rates (id serial not null, digits_prefix prefix_range, [...]);
create index rates_prefix_idx on rates using gist (digits_prefix gist_prefix_range_ops);
insert [...]
select * from rates where digits_prefix @> '16666666666';
```

A sample query that "works for me" is the following:

```
    <profile name="use_prefix">
      <param name="custom_sql" value="
SELECT l.digits, c.carrier_name, l.${lcr_rate_field}, cg.prefix AS gw_prefix, cg.suffix AS gw_suffix,
       l.lead_strip, l.trail_strip, l.prefix, l.suffix, cg.codec, l.cid
FROM lcr l
     JOIN carriers c ON l.carrier_id=c.id
     JOIN carrier_gateway cg ON c.id=cg.carrier_id
WHERE c.enabled = '1' AND cg.enabled = '1' AND l.enabled = '1'
      AND digits_prefix @> '%q'
      AND CURRENT_TIMESTAMP BETWEEN date_start AND date_end
ORDER BY digits DESC, ${lcr_rate_field} asc, random();
      "/>
    </profile>
```

And can be called by typing at the CLI:

```
lcr 12145551212 use_prefix
```

- **Note (2009-03-25)** You may need to install prefix from CVS for PostgreSQL 8.1 and 8.2 support.

## intralata / intrastate rating

**Note:** this is NANP (North American Numbering Plan) specific.

Some carriers will rate intralata and intrastate calls differently than normal long distance. mod_lcr supports querying a table to determine if the call is intralata or intrastate and provide alternate rates for those calls. This is done by using the caller id number and the called number.

If you plan to use intra type rating you need to have the intrastate_rate or intralata_rate fields defined in the lcr table. Do NOT leave the field null or 0 for carriers that don't distinguish between regular LD and intra style routing. Instead set them to the same value as rate. However, if you have some digit matches that only have intrastate or interstate but not both, then leave the blank one as NULL and mod_lcr will properly handle it.

intra lata/state selection is done manually by setting the channel variables **intrastate** or **intralata** to the value **true**.

Automatic selection requires the npanxx table loaded into the table **npa_nxx_company_ocn**. This is the same table as used by Mod_cidlookup#Falling_back_to_.22City_State.22_in_the_absense_of_a_name and refer to that for data loading.

## overriding the Caller ID

The field **cid** allows one to override the CID value on a per-route basis. Sometimes your different carriers have different rules as to what to pass for CID.

The field takes a regular expression of the form:

```
/search/replace/
```

In search you put parenthesis '()' around the data you wish to capture. Each () is assigned a $n value which you can use in the replace string.

If you wanted to strip a leading 1 from a #:

```
/^1(\d+)/$1/
```

Using channel variables in the regular expression is also valid, and they will be expanded to the appropriate session values when called from the dial plan. For example, it's possible to use a custom channel variable to specify the outbound CLI (or just a prefix/suffix) for a specific route, usually when the effective outbound CLI set in the user directory can not be directly related to the outgoing numbers for a specific route. This example will add the area code defined in the channel variable "my_outbound_caller_area_code" to the effective caller ID number:

```
/(\d+)/${my_outbound_caller_area_code}$1/
```

Channel variables can also be used in the matching part of the regular expression.

## User Rates

In general, calls should be rated "after the fact" using an actual billing engine. There are times, however, where one may want an idea of what the end-user rate is. This can be used in concert with mod_nibblebill for instance.

User Rates require one to use custom sql. The reason is that user rates really should not be part of the lcr tables. Instead, one should be retrieving the user rates from some other table based on information on the caller (eg: account code). The user rate may even be a calculated value based on the real rate (eg: 20% markup). The logic is up to you and can be anything expressed in sql (or even a stored procedure call if it is very complex).

The user rate is in the 12th position in the returned data from the custom sql query (right after CID).

If one wants different user_rates based on intrastate/intralata/interstate then one can use the ${lcr_user_rate} variable in the query. This variable will be set to one of "user_rate", "user_intralata_rate" or "user_intrastate_rate".

## LRN

mod_lcr supports feeding an LRN in for the callee but not caller.

mod_lcr will use the LRN to look up inter/intra properly, and to pull the correct rate from the carrier.

The schema has been expanded to let you set if the LCR rate should be looked up as a direct match to the callee, or based on the LRN. It is done per-row so you can turn on LRN only for certain locations, rather than for the entire carrier.

## LRN caching table placeholder

I have this schema for my lrn cache info -Avi Marcus

```
id        int(10)        UNSIGNED        Primary key
AUTO_INCREMENT
digits        bigint(11) UNSIGNED
lrn        bigint(11) UNSIGNED
date_lrn        timestamp
sip        varchar(60) (why not do an ENUM for carriers while we are at it?)
date_sip        timestamp
```

So a (fake) entry might like this this:

```
Digits: 12019359999 lrn: 19084411234 date_lrn 2010-11-19 02:55:52 sip: 12019359999@in.callcentric.com date_sip:
2010-11-19 02:55:52
```

We could even have it set /sofia/external/number@provider for the enum, as the first string.

# Using mysql with mod_lcr

mod_lcr passes the number to check to mysql. If you don't enclose them in quotes, mysql may take upto 22 times longer than is necessary. Add

```
<param name="quote_in_list" value="yes"/>
```

to each profile in your configuration file and see if the performance improves.

**Update** This is debatable - the opposite is more likely to be the case

# Using sqlite with mod_lcr

This installation assumes a debian based system, for other distributions please adapt as necessary.

Install the pre-requisite packages:

```
apt-get install libsqliteodbc unixodbc-bin sqlite3 sqlite
```

Add the following to /opt/freeswitch/.odbc.ini:

```
[fslcrdb]
Description=Freeswitch LCR SQLite database
Driver=SQLite3
Database=/opt/freeswitch/conf/databases/fslcr.db
# optional lock timeout in milliseconds
Timeout=2000
```

Create sqlite3.sql:

```
-- SQL Lite 3 tables

-- Table: carriers

-- DROP TABLE carriers;

CREATE TABLE carriers
(
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  carrier_name VARCHAR(255) NOT NULL,
  enabled INTEGER NOT NULL DEFAULT '1'
);

-- Table: carrier_gateway

-- DROP TABLE carrier_gateway;

CREATE TABLE carrier_gateway
(
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  carrier_id integer REFERENCES carriers(id),
  prefix VARCHAR(128) NOT NULL DEFAULT '',
  suffix VARCHAR(128) NOT NULL DEFAULT '',
  codec VARCHAR(128) NOT NULL DEFAULT '',
  enabled INTEGER NOT NULL DEFAULT '1'
);

-- Index: gateway

-- DROP INDEX gateway;

CREATE UNIQUE INDEX gateway
  ON carrier_gateway
  (prefix, suffix);


-- Table: lcr

-- DROP TABLE lcr;

CREATE TABLE lcr
(
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  digits NUMERIC(20, 0),
  rate numeric(11,5) NOT NULL,
  carrier_id integer NOT NULL REFERENCES carriers(id),
  lead_strip integer NOT NULL DEFAULT 0,
  trail_strip integer NOT NULL DEFAULT 0,
  prefix VARCHAR(16) NOT NULL DEFAULT '',
  suffix VARCHAR(16) NOT NULL DEFAULT '',
  lcr_profile INTEGER NOT NULL DEFAULT 0,
  date_start timestamp with time zone NOT NULL DEFAULT '1970-01-01',
  date_end timestamp with time zone NOT NULL DEFAULT '2030-12-31',
  quality numeric(10,6) NOT NULL DEFAULT 0,
  reliability numeric(10,6) NOT NULL DEFAULT 0,
  cid VARCHAR(32) NOT NULL DEFAULT '',
  enabled INTEGER NOT NULL DEFAULT '1'
);

-- Index: digits_rate

-- DROP INDEX digits_rate;

CREATE INDEX digits_rate
  ON lcr
  -- not supported -- USING btree
  (digits, rate);

-- Index: profile_digits_15

-- DROP INDEX profile_digits_15;

CREATE INDEX profile_digits_15
  ON lcr
  (digits, lcr_profile);

-- Index: unique_route

-- DROP INDEX unique_route;

CREATE INDEX unique_route
  ON lcr
  (digits, carrier_id);
```

Create sample_data.sql:

```
-- insert two carriers

INSERT INTO carriers (carrier_name) VALUES ('carrier1');
INSERT INTO carriers (carrier_name) VALUES ('carrier2');

-- insert some gateway info

INSERT INTO carrier_gateway (carrier_id, prefix, suffix) VALUES
  (1, 'sofia/gateway/carrier1/', '');
INSERT INTO carrier_gateway (carrier_id, prefix, suffix) VALUES
  (2, 'sofia/external/', '@proxy.carrier2.net:5060');

-- insert some lcr data

INSERT INTO lcr (digits, rate, carrier_id, lead_strip, trail_strip,
                 prefix, suffix,
                 date_start, date_end, quality, reliability) VALUES
       ('1', 0.15, 1, 0, 0, '', '',
        datetime(current_timestamp,'-1 year'),
        datetime(current_timestamp,'+1 year')
        , 0, 0);
INSERT INTO lcr (digits, rate, carrier_id, lead_strip, trail_strip,
                 prefix, suffix,
                 date_start, date_end, quality, reliability) VALUES
       ('1', 0.12, 2, 1, 0, '0', '',
        datetime(current_timestamp,'-1 year'),
        datetime(current_timestamp,'+1 year')
        , 0, 0);
INSERT INTO lcr (digits, rate, carrier_id, lead_strip, trail_strip,
                 prefix, suffix,
                 date_start, date_end, quality, reliability) VALUES
       ('1234', 0.05, 1, 0, 0, '', '',
        datetime(current_timestamp,'-1 year'),
        datetime(current_timestamp,'+1 year')
        , 0, 0);
INSERT INTO lcr (digits, rate, carrier_id, lead_strip, trail_strip,
                 prefix, suffix,
                 date_start, date_end, quality, reliability) VALUES
       ('1234', 0.02, 2, 1, 0, '0', '',
        datetime(current_timestamp,'-1 year'),
        datetime(current_timestamp,'+1 year')
        , 0, 0);
```

Load in these two SQL files into the DB:

```
sqlite3 /opt/freeswitch/conf/databases/fslcr.db
.read sqlite3.sql
.read sample_data.sql
```

# Problems with ODBC

If you have a problem with odbc not working when it works with other mods double check that your odbc.ini file is in /etc, that is where mod_lcr looks for the files. It does not matter what OS you are using.