**mod_memcache**

**About**

`mod_memcache` implements an API interface to `memcached` which is a "high-performance, distributed memory object caching system, generic in nature, but intended for use in speeding up dynamic web applications by alleviating database load.”

This module can be used to distribute key/value pairs without having to resort to using ODBC and a database.

**Requirements**

`mod_memcache` requires one to have at least one `memcached` server running to be useful.

**Installing**

To use `mod_memcache`:

Tell FreeSWITCH to compile in this module by editing `modules.conf` in the root of your freeswitch source tree and uncomment:

```
#applications/mod_memcache
```

Now go recompile FreeSWITCH.

```
make
make install
```

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

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

Finally, edit the default `config` in the `autoload_configs` directory to hold your memcache connection information.

Now load up FreeSWITCH!

**CLI**

From the commandline issue something similar to:

```
memcache status verbose
```

which would respond with something like:

```
Lib version: 0.27
Servers: 1
  localhost (11211)
    pid: 9542
    uptime: 25971
[...]
```

This verifies that you are actually communicating with your memcache servers.

**Dialplan Application**
The memcache API can be called from the dialplan using the `${api(args)}` method. Some examples:

```xml
<action application="set" data="foo=${memcache(get key)}"/>
<action application="set" data="ignore=${memcache(set key value)}"/>
<action application="set" data="ignore=${memcache(set key $channel_var)}"/>
```

**API**

The API can be called via ESL or used directly from fs_cli. There is pretty much a 1:1 relationship between the api below and the api exposed via libmemcached.

```plaintext
memcache <set|replace|add> <key> <value> [expiration [flags]]
   expiration is in seconds
   flags is a 64bit (preferably hex) value
memcache <get|getflags> <key>
   gets values or flags
memcache <delete> <key>
memcache <increment|decrement> <key> [offset [expires [flags]]]
   offset = how much to increase/decrease by
   expires = expiration in seconds -- subsequent inc/dec operations DO NOT extend lifetime of object
   flags = 64bit (preferably hex) value
memcache <flush>
   remove all keys (careful if this is a shared memcache server)
memcache <status> [verbose]
   retrieve server information
```

**Sharing Data With Other Applications**

This is also very useful to share data across applications. Here an example how to share data between FreeSWITCH and a ruby memcache-client:

On Ruby/Rails set the namespace e.g. to "freeswitch" for the same memcached server in environment.rb

In FreeSWITCH add the following line to the dialplan:

```xml
<action application="set" data="ignore=${memcache(set freeswitch:test 'This is a test')}"/>
```

Take care to prefix your key (here "test") with the Ruby namespace "freeswitch:"

Now you can receive the data in Ruby in raw mode:

```ruby
>> CACHE.get("test",0)
=> 'This is a test'
```

The 0 as second parameter is important for the raw mode, otherwise ruby will try to marshall the result from memcached and fails.