

Mod_perl_and_Generating_XML

There are two primary ways to use mod_perl in freeswitch. This page will cover using it to generate dialplan/directory/configuration XML. It should be noted that if you do not need direct access to freeswitch/other variables you can use perl to generate XML without using mod_perl by just using the curl support (as you can then run any application and have the XML injected back into freeswitch). If your solution requires more advanced access however mod_perl is your solution. You must configure mod_perl for generating XML plans, for details on that please see the main [mod_perl](#) entry.

How to access request parameters and how to return data

You have two hashes that are populated for you by freeswitch. Those hashes are:

- %XML_REQUEST
- %XML_DATA

You also have \$params that is a wrapped switch event.

You should return a valid XML dialplan in the \$XML_STRING variable before exiting your code.

See the perl code example below to start experimenting. the code will dump all the data you need for your login on the console.

Perl Code Examples

[Mod_perl Rosetta](#) for Asterisk::AGI

- Called from dialplan

```
#!/usr/bin/perl

freeswitch::console_log("info", "Perl rocks\n");

while( ($name, $value) = each(%XML_REQUEST)) {
    freeswitch::console_log("info", '[XML_REQUEST] '.$name => $value\n");
}

# %XML_DATA is only present when perl is used as a binding
while( ($name, $value) = each(%XML_DATA)) {
    freeswitch::console_log("info", '[XML_DATA] '.$name => $value\n");
}

# even $params is full of data
my $xml_dump = $params->serialize();
freeswitch::console_log("info", "[PARAMS] $xml_dump\n");

$xml_string = '
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<document type="freeswitch/xml">
  <section name="dialplan" description="Perl RE Dial Plan For FreeSwitch">
    <context name="default">
```

Mod_perl_and_Generating_XML

```
<extension name="test">
  <condition field="destination_number" expression="^991$">
    <action application="playback" data="tone_stream://path=${base_dir}/conf/tetris.ttml;lo
  </condition>
</extension>
</context>
</section>
</document>
';
```

- Called from CLI or webapi

```
#!/usr/bin/perl
#
use strict;
use warnings;

our $stream;
$stream->write("Content-Type: text/plain\n\n");

## Get OS Type and make a decision
if ( $^O eq "linux" ) {
    my $res = `uname -a`;
    $stream->write("OS Info: $res\n");
} else {
    $stream->write("Loser! Your OS is: $^O\n");
}

# End with a "true value" as it were
1;
```

Note: I put this script into my `/usr/local/freeswitch/scripts` directory as `sysinfo.pl`. It can then be run thusly:

```
perl sysinfo.pl -OR- perlrun sysinfo.pl (from the CLI)
http://your-fs-host-name-or-ip:8080/api/perl?sysinfo.pl (URL for a browser)
```