Serving Configuration with Lua

About

mod_lua allows you to replace static XML file lookups with configuration data served by Lua scripts.

Serving Configuration with mod_lua

With mod_lua you can serve your configurations. You can bind a Lua script to the XML req, like you do with URL in mod_xml_curl. When FreeSWITCH looks up sections in the XML registry, it calls your script. Your script will do any db lookups or whatever it needs, and it returns the XML string.

The file conf/autoload_configs/lua.conf.xml is used in the default FreeSWITCH setup.

> When you edit lua.conf.xml, giving the reloadxml command from the FreeSWITCH console is not sufficient to recognize the "xml-handler-script" parameters, you have to restart FreeSWITCH.

Here is a minimum configuration file, it will fetch a Dialplan from *Lua script.*

```
<configuration name="lua.conf" description="LUA Configuration">
  <settings>
    <param name="xml-handler-script" value="dp.lua"/>
    <param name="xml-handler-bindings" value="dialplan"/>
  </settings>
</configuration>
```

Some examples for the "xml-handler-bindings" parameter: "dialplan" "directory" "configuration" "dialplan|directory|configuration"

- The script is passed an XML_REQUEST object which has section, tag_name, key_name, and key_value (see mod_xml_curl). For some configuration requests, for directory requests, and for diaplan requests, an event object is also passed in the variable 'params'.
- Whatever you put in XML_STRING will be returned to FreeSWITCH once the script is finished.

You can access the XML_REQUEST object in lua with the variables XML_REQUEST["section"], XML_REQUEST["tag_name"], XML_REQUEST["key_name"], and XML_REQUEST["key_value"]

Example:

```
freeswitch.consoleLog("notice", "SECTION " .. XML_REQUEST["section"] .. "\n")
```

Module configuration

If mod_lua is loaded during initial startup, then modules loaded after initial starup may have their configuration served by the bound script whenever they request configuration, which for many modules is on loading.

```
<configuration name="lua.conf" description="LUA Configuration">
  <settings>
    <param name="xml-handler-script" value="configuration.lua"/>
    <param name="xml-handler-bindings" value="configuration"/>
  </settings>
</configuration>
```

Then when the module is started, XML_REQUEST object in the Lua script will have:

- **key_value =** 'iax.conf'|'event_socket.conf'|sofia.conf'|...
- **key_name =** 'name'
- **section =** 'configuration'
- **tag_name =** 'configuration'

'params' event object		
acl.conf	no	N/A
event_socket.conf	no	N/A
post_load_switch.conf	no	N/A

sofia.conf	yes	`Event-Name: REQUEST_PARAMS`
		`Core-UUID: 0f8afb73-2183-a1e2-2316-71053c746130`
		`FreeSWITCH-Hostname: hostname`
		`FreeSWITCH-IPv4: 192.168.1.12`
		`FreeSWITCH-IPv6: %3A%3A1`
		`Event-Date-Local: 2010-08-06%2014%3A04%3A38`
		`Event-Date-GMT: Fri,%2006%20Aug%202010%2018%3A04%3A38%20GMT`
		`Event-Date-Timestamp: 1281117878629975`
		`Event-Calling-File: sofia.c`
		`Event-Calling-Function: config_sofia`
		`Event-Calling-Line-Number: 2637`

switch.conf	no	N/A
syslog.conf	no	N/A

Directory

Initial startup

During initial startup/reading profiles, the directory is read for gateways and aliasing domains.

Then when the module is started, XML_REQUEST object in the Lua script will have:

- **key_value =** ''
- **key_name =** ''
- **section =** 'directory'
- **tag_name =** ''

And **"params"** will have an **event object**.

The directory is read once for each profile in the sofia configuration

> The variables are the empty string, not nil.

Params event headers

External profile

Event External profile Expand source

```
Event-Name: REQUEST_PARAMS
Core-UUID: <uuid>
FreeSWITCH-Hostname: hostname
FreeSWITCH-IPv4: 192.168.1.11
FreeSWITCH-IPv6: %3A%3A1
Event-Date-Local: 2010-08-06%2014%3A04%3A40
Event-Date-GMT: Fri,%2006%20Aug%202010%2018%3A04%3A40%20GMT
Event-Date-Timestamp: 1281117880813532
Event-Calling-File: sofia.c
Event-Calling-Function: config_sofia
Event-Calling-Line-Number: 3481
purpose: gateways
profile: external
```

The important bits:

```
purpose: gateways
profile: external
...
FreeSWITCH-Hostname: hostname
FreeSWITCH-IPv4: 192.168.1.11
```

Internal profile

Event Internal profile Expand source

```
Event-Name: REQUEST_PARAMS
Core-UUID: <uuid>
FreeSWITCH-Hostname: hostname
FreeSWITCH-IPv4: 192.168.1.11
FreeSWITCH-IPv6: %3A%3A1
Event-Date-Local: 2010-08-06%2014%3A04%3A41
Event-Date-GMT: Fri,%2006%20Aug%202010%2018%3A04%3A41%20GMT
Event-Date-Timestamp: 1281117881174514
Event-Calling-File: sofia.c
Event-Calling-Function: config_sofia
Event-Calling-Line-Number: 3481
purpose: gateways
profile: internal
```

The important bits:

```
purpose: gateways
profile: internal
...
FreeSWITCH-Hostname: hostname
FreeSWITCH-IPv4: 192.168.1.11
```

Network lists

Also during initial startup the network lists are read from the directory, however at this point the XML_REQUEST has:

- **key_value =** <name-of-domain> (e.g. 192.168.1.11)
- **key_name =** 'name'
- **section =** 'directory'
- **tag_name =** 'domain'

Event Network lists Expand source

```
Event-Name: GENERAL
Core-UUID: <uuid>
FreeSWITCH-Hostname: hostname
FreeSWITCH-IPv4: 192.168.1.11
FreeSWITCH-IPv6: %3A%3A1
Event-Date-Local: 2010-08-06%2014%3A04%3A46
Event-Date-GMT: Fri,%2006%20Aug%202010%2018%3A04%3A46%20GMT
Event-Date-Timestamp: 1281117886025842
Event-Calling-File: switch_core.c
Event-Calling-Function: switch_load_network_lists
Event-Calling-Line-Number: 1040
domain: 192.168.1.11
purpose: network-list
```

The important bits:

```
domain: 192.168.1.11
purpose: network-list
...
FreeSWITCH-Hostname: hostname
FreeSWITCH-IPv4: 192.168.1.11
```

When looking for a user

When looking for a user, FreeSWITCH looks for the user in a particular domain.

XML_REQUEST will have:

- **key_value =** '<name-of-domain>'
- **key_name =** 'name'
- **section =** 'directory'
- **tag_name =** 'domain'

and '**params**' will have an **event object**

When registering (REGISTER)

Event REGISTER Expand source

```
Event-Name: REQUEST_PARAMS
Core-UUID: <uuid>
FreeSWITCH-Hostname: hostname
FreeSWITCH-IPv4: 192.168.1.11
FreeSWITCH-IPv6: %3A%3A1
Event-Date-Local: 2010-08-06%2014%3A04%3A43
Event-Date-GMT: Fri,%2006%20Aug%202010%2018%3A04%3A43%20GMT
Event-Date-Timestamp: 1281117883173795
Event-Calling-File: sofia_reg.c
Event-Calling-Function: sofia_reg_parse_auth
Event-Calling-Line-Number: 1797
action: sip_auth
sip_profile: internal
sip_user_agent: IP-Phone-V3.2.49T5.13%20-%20G729
sip_auth_username: 1000
sip_auth_realm: 192.168.1.11
sip_auth_nonce: <auth_nonce_uuid>
sip_auth_uri: sip%3A192.168.1.11
sip_contact_user: 1000
sip_contact_host: 192.168.88.202
sip_to_user: 1000
sip_to_host: 192.168.1.11
sip_to_port: 5060
sip_from_user: 1000
sip_from_host: 192.168.1.11
sip_from_port: 5060
sip_request_host: 192.168.1.11
sip_auth_qop: auth
sip_auth_cnonce: 829326
sip_auth_nc: 00000001
sip_auth_response: <auth_response - md5sum?>
sip_auth_method: REGISTER
key: id
user: 1000
domain: 192.168.1.11
ip: 192.168.88.202
```

The important bits:

```
key: id
user: 1000
domain: 192.168.1.11
...
action: sip_auth
sip_profile: internal
FreeSWITCH-Hostname: hostname
FreeSWITCH-IPv4: 192.168.1.11
ip: 192.168.88.202
```

When calling (INVITE)

Event INVITE Expand source

```
Event-Name: REQUEST_PARAMS
Core-UUID: <uuid>
FreeSWITCH-Hostname: hostname
FreeSWITCH-IPv4: 192.168.1.11
FreeSWITCH-IPv6: %3A%3A1
Event-Date-Local: 2010-08-06%2016%3A28%3A08
Event-Date-GMT: Fri,%2006%20Aug%202010%2020%3A28%3A08%20GMT
Event-Date-Timestamp: 1281126488274011
Event-Calling-File: sofia_reg.c
Event-Calling-Function: sofia_reg_parse_auth
Event-Calling-Line-Number: 1797
action: sip_auth
sip_profile: internal
sip_user_agent: IP-Phone-V3.2.49T5.13%20-%20G729
sip_auth_username: 1001
sip_auth_realm: 192.168.1.11
sip_auth_nonce: 1fe3d1fa-a199-11df-b392-b105e374638e
sip_auth_uri: sip%3A1000%40192.168.1.11
sip_contact_user: 1001
sip_contact_host: 192.168.88.99
sip_to_user: 1000
sip_to_host: 192.168.1.11
sip_to_port: 5060
sip_from_user: 1001
sip_from_host: 192.168.1.11
sip_from_port: 5060
sip_request_user: 1000
sip_request_host: 192.168.1.11
sip_auth_qop: auth
sip_auth_cnonce: 10560d0
sip_auth_nc: 00000001
sip_auth_response: b99d1213022480a2b6c4e14432661821
sip_auth_method: INVITE
key: id
user: 1001
domain: 192.168.1.11
ip: 192.168.88.99
```

The important bits:

```
key: id
user: 1001
domain: 192.168.1.11
...
ip: 192.168.88.99
FreeSWITCH-Hostname: hostname
FreeSWITCH-IPv4: 192.168.1.11
action: sip_auth
sip_profile: internal
```

When being called (by another extension)

Event when being called Expand source

```
Event-Name: REQUEST_PARAMS
Core-UUID: <uuid>
FreeSWITCH-Hostname: hostname
FreeSWITCH-IPv4: 192.168.1.11
FreeSWITCH-IPv6: %3A%3A1
Event-Date-Local: 2010-08-06%2016%3A28%3A09
Event-Date-GMT: Fri,%2006%20Aug%202010%2020%3A28%3A09%20GMT
Event-Date-Timestamp: 1281126489462522
Event-Calling-File: mod_dptools.c
Event-Calling-Function: user_outgoing_channel
Event-Calling-Line-Number: 2662
as_channel: true
action: user_call
key: id
user: 1000
domain: 192.168.1.11
```

the important bits:

```
key: id
user: 1000
domain: 192.168.1.11
...
FreeSWITCH-Hostname: hostname
FreeSWITCH-IPv4: 192.168.1.11
as_channel: true
Event-Calling-Function: user_outgoing_channel
```

Dialplan

XML

This is a Lua script sample dp.lua to provision a Dialplan, whatever is in XML_STRING will be returned to Freeswitch once the script is finished.

Lua script dp.lua - Dialplan Expand source

```
-- params is the event passed into us we can use params:getHeader to grab things we
want.
io.write("TEST\n" .. params:serialize("xml") .. "\n");

mydialplan = [[
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<document type="freeswitch/xml">
  <section name="dialplan" description="RE Dial Plan For FreeSwitch">
    <context name="default">
      <extension name="freeswitch_public_conf_via_sip">
        <condition field="destination_number" expression="^9(888|1616)$">
          <action application="bridge"
data="sofia/${use_profile}/$1@conference.freeswitch.org"/>
        </condition>
      </extension>
    </context>
  </section>
</document>
]]

XML_STRING = mydialplan
-- comment the following line for production:
freeswitch.consoleLog("notice", "Debug XML:\n" .. XML_STRING .. "\n")
```

Lua

In for example a sip profile you can - instead of XML dialplan - use Lua directly to generate a dialplan:

```
<profile name="phones">
  <!-- ... -->
  <settings>
    <param name="dialplan" value="LUA"/>
    <param name="context" value="dialplan-from-phones.lua"/>
    <!-- ... -->
  </settings>
</profile>
```

Then create a dialplan in scripts/dialplan-from-phones.lua like this:

```
-- dialplan-from-phones.lua

ACTIONS = {}
freeswitch.consoleLog("notice", "from your script")

table.insert(ACTIONS, "answer")
table.insert(ACTIONS, {"log", "NOTICE after your script"})
```

The entire table (array) of actions will be executed **after** the Lua script has stopped.

This works the same as when the XML dialplan generates a list of actions that is run after it has reached the end.

Note that you must not use media directly from your script (eg *session:answer()*) like you could if it were called as an IVR script.

Advantage of this method over answering from an IVR script is that during the call itself there will be no Lua script (per session) running - which could add up if you have a few thousand sessions going.

Alternative is to call a Lua script from the XML dialplan that only sets some channel variables that will be picked up by consecutive applications - or you can generate an XML stub as is described in the previous section.

Of course it is also possible to transfer from one dialplan to another:

From Lua to XML:

```
table.insert(ACTIONS, {"transfer", "123 XML some-context"})
```

From XML to Lua:

```
<action application="transfer" data="123 LUA some-dialplan.lua"/>
```

Not found

If your LUA application receives a request and you don't wish to serve dialplan or like to fallback to plain XML dialplan, then you should return the following "not found" result.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<document type="freeswitch/xml">
  <section name="result">
    <result status="not found" />
  </section>
</document>
```

If you return an empty response instead of the not found, you may see the following error;

```
[ERR] switch_xml.c:1534 switch_xml_locate() Error[[error near line 1]: root tag
missing]
```