Channel Variables

About

Channel variables are used to manipulate dialplan execution, to control call progress, and to provide options to applications. They play a pervasive role, as FreeSWITCH™ frequently consults channel variables as a way to customize processing prior to a channel's creation, during call progress, and after the channel hangs up.

Variable Expansion

We rely on variable expansion to create flexible, reusable dialplans:

- `$${variable}` is expanded once when FreeSWITCH™ first parses the configuration on startup or after invoking `reloadxml`. It is suitable for variables that do not change, such as the domain of a single-tenant FreeSWITCH™ server. That is why `$${domain}` is referenced so frequently in the vanilla dialplan examples.
- `${variable}` is expanded during each pass through the dialplan, so it is used for variables that are expected to change, such as the `${destination_number}` or `${sip_to_user}` fields.

Channel Variables in the XML Dialplan

Channel variables are set, appropriately enough, with the set application:

```
<action application="set" data="var_name=var value"/>
```

Reading channel variables requires the `${}` syntax:

```
<action application="log" data="INFO The value in the var_name chan var is ${var_name}"/>
<condition field="${var_name}" expression="some text">
```

Scoped Variables

Channel variables used to be global to the session. As of b2c3199f, it is possible to set variables that only exist within a single application execution and any subsequent applications under it. For example, applications can use scoped variables for named input params:

```
<action application="log" data="INFO myvar is '${myvar}'"/>
<action application="log" data="%[myvar=Hello]INFO myvar is '${myvar}'"/>
<action application="log" data="INFO myvar is '${myvar}'"/>
<action application="myapp" data="%[var1=val1,var2=val2]mydata"/>
```

Channel Variables in Dial Strings

The variable assignment syntax for dial strings differs depending on which scope they should apply to:

- `{foo=bar}` is **only** valid at the beginning of the dial string. It will set the same variables on **every** channel, but does not do so for enterprise bridging/originate.
- `<foo=bar>` is **only** valid at the beginning of a dial string. It will set the same variables on **every** channel, including all thos in an enterprise bridging/originate.
- `[foo=bar]` goes before each individual dial string and will set the variable values specified for only this channel.

Examples

Set `foo` variable for all channels implemented and `chan=1` will only be set for `blah`, while `chan=2` will only be set for `blah2`:

```
{foo=bar}[chan=1]sofia/default/blah@baz.com,[chan=2]sofia/default/blah2@baz.com
```

Set multiple variables by delimiting with commas:

```
[var1=abc,var2=def,var3=ghi]sofia/default/blah@baz.com
```

To have variables in [] override variables in {}, set local_var_clobber=true inside {}. You must also set local_var_clobber=true when you want to override channel variables that have been exported to your b-legs in your dialplan. In this example, the legs for blah1@baz.com and johndoe@example.com would be set to offer SRTP (RTP/SAVP) while janedoe@acme.com would not receive an SRTP offer (she would see RTP/AVP instead):

```
{local_var_clobber=true,rtp_secure_media=true}sofia/default/blah1@baz.com|sofia/default/johndoe@example.com|rtp_secure_media=false]sofia/default/janedoe@acme.com
```

Escaping/Redefining Delimiters

Commas are the default delimiter inside variable assignment tags. In some cases (like in absolute_codec_string), we may need to define variables whose values contain literal commas that should not be interpreted as delimiters.

We can redefine the delimiter for a variable using ^^ followed by the desired delimiter:

```
^^;one,two,three;four,five,six;seven,eight,nine
```

To set absolute_codec_string=PCMA@8000h@20i@64000b,PCMU@8000h@20i@64000b,G729@8000h@20i@8000b in a dial string:

```
{absolute_codec_string=^^:PCMA@8000h@20i@64000b:PCMU@8000h@20i@64000b:G729@8000h@20i@8000b,leg_time_out=10,process_cdr=b_only}
```

This approach does not work when setting sip_h_*, sip_rh_*, and sip_ph headers. To pass a comma into the contents of a private header, escape the comma with a backslash:

```
{sip_h_X-My-Header=one\,two\,three,leg_time_out=10,process_cdr=b_only}
```

Exporting Channel Variables in Bridge Operations

Variables from one call leg (A) can be exported to the other call leg (B) by using the export_vars variable. Its value is a comma separated list of variables that should propagate across calls.

```
<action application="set" data="export_vars=myvar,myvar2,foo,bar"/>
```

To set a variable on the A-leg and add it to the export list, use the export application:

```
<action application="export" data="myvar=true"/>
```

Using Channel Variables in Dialplan Condition Statements

Channel variables can be used in conditions, refer to XML Dialplan Conditions for more information.

⚠️ Some channel variables may not be set during the dialplan parsing phrase. See Inline Actions.

Custom Channel Variables

We are not constrained to the channel variables that FreeSWITCH™, its modules, and applications define. It is possible to set any number of unique channel variables for any purpose. They can also be logged in CDR.

The set application can be used to set any channel variable:

```
<action application="set" data="lead_id=2e4b5966-0aaf-11e8-ba89-0ed5f89f718b"/>
<action application="set" data="campaign_id=333814"/>
<action application="set" data="crm_tags=referral new loyal" />
```

In a command issued via mod_xml_rpc or mod_event_socket:

```
originate {lead_id=2e4b5966-0aaf-11e8-ba89-0ed5f89f718,campaign_id=333814}sofia/mydomain.com/18005551212@1.
2.3.4 15555551212
```

Values with spaces must be enclosed by quotes:

```
originate {crm_tags='referral new loyal'}sofia/mydomain.com/18005551212@1.2.3.4 15555551212
```

Channel Variable Manipulation

Channel variables can be manipulated for varied results. For example, a channel variable could be trimmed to get the first three digits of a phone number. Manipulating Channel Variables discusses this in detail.

Channel Variable Scope Example

Consider this example:

```
<extension name="test" continue="false">
  <condition field="destination_number" expression="^test([0-9]+)$">
    <action application="set" data="fruit=tomato" /> <!-- Set variable in local channel -->
    <action application="export" data="veggie=tomato" /> <!-- Set variable in local channel and export it to
new channels we bridge to -->
    <action application="bridge" data="{meat=tomato}sofia/gateway/testaccount/1234" /><!-- bridge new
channel and set variable only in new channel -->
  </condition>
</extension>
```

Leg A (the channel that called the dial plan) will have these variables set:

```
fruit: tomato
veggie: tomato
```

Leg B (the channel created with `sofia/gateway/testaccount/1234`) will have these variables set:

```
fruit: tomato
meat: tomato
```

Accessing Channel Variables in Other Environments

In addition to the dialplan, channel variables can be set in other environments as well.

In a FreeSWITCH™ module, written in C:

```
switch_channel_set_variable(channel,"name","value");
char* result = switch_channel_get_variable(channel,"name");
char* result = switch_channel_get_variable_partner(channel,"name");
```

In the console (or `fs_cli`, implemented in mod_commands):

```
uuid_getvar <uuid> <name>
uuid_setvar <uuid> <name> [<value>]
uuid_setvar_multi <uuid> <name>=<value>[;<name>=<value>[;...]]
```

Alternatively, call `uuid_dump` to get *all* the variables, or use the `eval` command, adding the prefix `variable_` to the key:

```
uuid_dump <uuid>
eval uuid:<uuid> ${variable_<name>}
```

In an event socket, just extend the above with the `api` prefix:

```
api uuid_getvar <uuid> <name>
```

In Lua, there are several ways to interact with variables. In the `freeswitch.Session()` invocation that creates a new `Session` object, variables go in square brackets:

```
s = freeswitch.Session("[myname=myvars]sofia/localhost/1003")
```

With the new `Session` object `s`:

```
local result1 = s:getVariable("myname") -- "myvars"
s:setVariable("name", "value")
local result2 = s:getVariable("name") -- "value"
```

Info Application Variable Names (`variable_xxxx`)

Some variables, as shown from the info app, may have `variable_` in front of their names. For example, if you pass a header variable called `type` from the proxy server, it will get displayed as `variable_sip_h_type` in FreeSWITCH™. To access that variable, you should strip off the `variable_`, and just do `${sip_h_type}`. Other variables shown in the info app are prepended with channel, which should be stripped as well. The example below show a list of info app variables and the corresponding channel variable names:

Info variable name	**channel variable name**	**Description**
Channel-State	state	Current state of the call
Channel-State-Number	state_number	Integer
Channel-Name	channel_name	Channel name
Unique-ID	uuid	uuid of this channel's call leg
Call-Direction	direction	Inbound or Outbound
Answer-State	state	-
Channel-Read-Codec-Name	read_codec	the read codec variable mean the source codec
Channel-Read-Codec-Rate	read_rate	the source rate
Channel-Write-Codec-Name	write_codec	the destination codec same to write_codec if not transcoded
Channel-Write-Codec-Rate	write_rate	destination rate same to read rate if not transcoded
Caller-Username	username	.
Caller-Dialplan	dialplan	user dialplan like xml, lua, enum, lcr
Caller-Caller-ID-Name	caller_id_name	.
Caller-Caller-ID-Number	caller_id_number	.
Caller-ANI	ani	ANI of caller, frequently the same as caller ID number
Caller-ANI-II	aniii	ANI II Digits (OLI - Originating Line Information), if available. Refer to: http://www.nanpa.com/number_resource_info/ani_ii_digits.html
Caller-Network-Addr	network_addr	IP address of calling party
Caller-Destination-Number	destination_number	Destination (dialed) number

Caller-Unique-ID	uuid	This channel's uuid
Caller-Source	source	Source module, i.e. mod_sofia, mod_openzap, etc.
Caller-Context	context	Dialplan context
Caller-RDNIS	rdnis	Redirected DNIS info. See mod_dptools: transfer application
Caller-Channel-Name	channel_name	.
Caller-Profile-Index	profile_index	.
Caller-Channel-Created-Time	created_time	.
Caller-Channel-Answered-Time	answered_time	.
Caller-Channel-Hangup-Time	hangup_time	.
Caller-Channel-Transfer-Time	transfer_time	.
Caller-Screen-Bit	screen_bit	.
Caller-Privacy-Hide-Name	privacy_hide_name	.
Caller-Privacy-Hide-Number	privacy_hide_number	This variable tells you if the inbound call is asking for CLIR[Calling Line ID presentation Restriction] (either with anonymous method or Privacy:id method)
	initial_callee_id_name	Sets the callee id name during the 183. This allows the phone to see a name of who they are calling prior to the phone being answered. An example of setting this to the caller id name of the number being dialled: `<action application="set" data="initial_callee_id_name='${user_data (${dialed_extension}@${domain_name} var effective_caller_id_name)}'"/>`
variable_sip_received_ip	sip_received_ip	.
variable_sip_received_port	sip_received_port	.
variable_sip_authorized	sip_authorized	.
variable_sip_mailbox	sip_mailbox	.
variable_sip_auth_username	sip_auth_username	.
variable_sip_auth_realm	sip_auth_realm	.
variable_mailbox	mailbox	.
variable_user_name	user_name	.
variable_domain_name	domain_name	.
variable_record_stereo	record_stereo	.
variable_accountcode	accountcode	Accountcode for the call. This is an arbitrary value. It can be defined in the user variables in the directory, or it can be set/modified from dialplan. The accountcode may be used to force a specific CDR CSV template for the call.
variable_user_context	user_context	.
variable_effective_caller_id_name	effective_caller_id_name	.

variable_effective_caller_id_number	effective_caller_id_number	.
variable_caller_domain	caller_domain	.
variable_sip_from_user	sip_from_user	.
variable_sip_from_uri	sip_from_uri	.
variable_sip_from_host	sip_from_host	.
variable_sip_from_user_stripped	sip_from_user_stripped	.
variable_sip_from_tag	sip_from_tag	.
variable_sofia_profile_name	sofia_profile_name	.
variable_sofia_profile_domain_name	sofia_profile_domain_name	.
variable_sip_full_route	sip_full_route	The complete contents of the Route: header.
variable_sip_full_via	sip_full_via	The complete contents of the Via: header.
variable_sip_full_from	sip_full_from	The complete contents of the From: header.
variable_sip_full_to	sip_full_to	The complete contents of the To: header.
variable_sip_req_params	sip_req_params	.
variable_sip_req_user	sip_req_user	.
variable_sip_req_uri	sip_req_uri	.
variable_sip_req_host	sip_req_host	.
variable_sip_to_params	sip_to_params	.
variable_sip_to_tag	sip_to_tag	.
variable_sip_to_user	sip_to_user	.
variable_sip_to_uri	sip_to_uri	.
variable_sip_to_host	sip_to_host	.
variable_sip_contact_params	sip_contact_params	.
variable_sip_contact_user	sip_contact_user	.
variable_sip_contact_port	sip_contact_port	.
variable_sip_contact_uri	sip_contact_uri	.

variable_sip_contact_host	sip_contact_host	.
variable_sip_invite_domain	sip_invite_domain	.
variable_channel_name	channel_name	.
variable_sip_call_id	sip_call_id	SIP header Call-ID
variable_sip_user_agent	sip_user_agent	.
variable_sip_via_host	sip_via_host	.
variable_sip_via_port	sip_via_port	.
variable_sip_via_rport	sip_via_rport	.
variable_presence_id	presence_id	.
variable_sip_h_P-Key-Flags	sip_h_p-key-flags	This will contain the optional P-Key-Flags header(s) that may be received from calling endpoint.
variable_switch_r_sdp	switch_r_sdp	The whole SDP received from calling endpoint.
variable_remote_media_ip	remote_media_ip	.
variable_remote_media_port	remote_media_port	.
variable_write_codec	write_codec	.
variable_write_rate	write_rate	.
variable_endpoint_disposition	endpoint_disposition	.
variable_dialed_ext	dialed_ext	.
variable_transfer_ringback	transfer_ringback	.
variable_call_timeout	call_timeout	.
variable_hangup_after_bridge	hangup_after_bridge	.
variable_continue_on_fail	continue_on_fail	.
variable_dialed_user	dialed_user	.
variable_dialed_domain	dialed_domain	.
variable_sip_redirect_contact_user_0	sip_redirect_contact_user_0	.
variable_sip_redirect_contact_host_0	sip_redirect_contact_host_0	.
variable_sip_h_Referred-By	sip_h_referred-by	.

variable_sip_refer_to	sip_refer_to	The full SIP URI received from a SIP Refer-To: response
variable_max_forwards	max_forwards	.
variable_originate_disposition	originate_disposition	.
variable_read_codec	read_codec	.
variable_read_rate	read_rate	.
variable_open	open	.
variable_use_profile	use_profile	.
variable_current_application	current_application	.
variable_ep_codec_string	ep_codec_string	This variable is only available if late negotiation is enabled on the profile. It's a readable string containing all the codecs proposed by the calling endpoint. This can be easily parsed in the dialplan.
variable_rtp_disable_hold	rtp_disable_hold	This variable when set will disable the hold feature of the phone.
variable_sip_acl_authed_by	sip_acl_authed_by	This variable holds what ACL rule allowed the call.
variable_curl_response_data	curl_response_data	This variable stores the output from the last curl made.
variable_drop_dtmf	drop_dtmf	Set on a channel to drop DTMF events on the way out.
variable_drop_dtmf_masking_file	drop_dtmf_masking_file	If drop_dtmf is true play specified file for every tone received.
variable_drop_dtmf_masking_digits	drop_dtmf_masking_digits	If drop_dtmf is true play specified tone for every tone received.
sip_codec_negotiation	sip_codec_negotiation	sip_codec_negotiation is basically a channel variable equivalent of inbound-codec-negotiation. sip_codec_negotiation accepts "scrooge" & "greedy" as values. This means you can change codec negotiation on a per call basis.
Caller-Callee-ID-Name	-	-
Caller-Callee-ID-Number	-	-
Caller-Channel-Progress-Media-Time	-	-
Caller-Channel-Progress-Time	-	-
Caller-Direction	-	-
Caller-Profile-Created-Time	profile_created	-
Caller-Transfer-Source	-	-
Channel-Call-State	-	-
Channel-Call-UUID	-	-
Channel-HIT-Dialplan	-	-
Channel-Read-Codec-Bit-Rate	-	-

Channel-Write-Codec-Bit-Rate	-	-
Core-UUID	-	-
Event-Calling-File	-	-
Event-Calling-Function	-	-
Event-Calling-Line-Number	-	-
Event-Date-GMT	-	-
Event-Date-Local	-	-
Event-Date-Timestamp	-	-
Event-Name	-	-
Event-Sequence	-	-
FreeSWITCH-Hostname	-	-
FreeSWITCH-IPv4	-	-
FreeSWITCH-IPv6	-	-
FreeSWITCH-Switchname	-	-
Hunt-ANI	-	-
Hunt-Callee-ID-Name	-	-
Hunt-Callee-ID-Number	-	-
Hunt-Caller-ID-Name	-	-
Hunt-Caller-ID-Number	-	-
Hunt-Channel-Answered-Time	-	-
Hunt-Channel-Created-Time	-	-
Hunt-Channel-Hangup-Time	-	-
Hunt-Channel-Name	-	-
Hunt-Channel-Progress-Media-Time	-	-
Hunt-Channel-Progress-Time	-	-
Hunt-Channel-Transfer-Time	-	-
Hunt-Context	-	-
Hunt-Destination-Number	-	-
Hunt-Dialplan	-	-
Hunt-Direction	-	-

Hunt-Network-Addr	-	-
Hunt-Privacy-Hide-Name	-	-
Hunt-Privacy-Hide-Number	-	-
Hunt-Profile-Created-Time	profile_created	-
Hunt-Profile-Index	-	-
Hunt-RDNIS	-	-
Hunt-Screen-Bit	-	-
Hunt-Source	-	-
Hunt-Transfer-Source	-	-
Hunt-Unique-ID	-	-
Hunt-Username	-	-
Presence-Call-Direction	-	-
variable_DIALSTATUS	-	-
variable_absolute_codec_string	-	-
variable_advertised_media_ip	-	-
variable_answersec		
variable_answermsec		
variable_answerusec		
variable_billsec		
variable_billmsec		
variable_billusec		
variable_bridge_channel	-	-
variable_bridge_hangup_cause	-	-
variable_bridge_uuid	-	-
variable_call_uuid	-	-
variable_current_application_response	-	-
variable_direction	-	-
variable_duration		
variable_mduration		
variable_uduration		
variable_inherit_codec	-	-

variable_is_outbound	-	-
variable_last_bridge_to	-	-
variable_last_sent_callee_id_name	-	-
variable_last_sent_callee_id_number	-	-
variable_local_media_ip	-	-
variable_local_media_port	-	-
variable_number_alias	-	-
variable_originate_early_media	-	-
variable_originating_leg_uuid	-	-
variable_originator	-	-
variable_originator_codec	-	-
variable_outbound_caller_id_number	-	-
variable_progresssec		
variable_progressmsec		
variable_progressusec		
variable_progress_mediasec		
variable_progress_mediamsec		
variable_progress_mediausec		
variable_recovery_profile_name	-	-
variable_rtp_use_ssrc	-	-
variable_session_id	-	-
variable_sip_2833_recv_payload	-	-
variable_sip_2833_send_payload	-	-
variable_sip_P-Asserted-Identity	-	-
variable_sip_Privacy	-	-
variable_sip_audio_recv_pt	-	-

variable_sip_cid_type	-	-
variable_sip_cseq	-	-
variable_sip_destination_url	-	-
variable_sip_from_display	sip_from_display	'User' element of SIP From: line
variable_sip_from_port	-	-
variable_sip_gateway	-	-
variable_sip_gateway_name	-	-
variable_sip_h_P-Charging-Vector	-	-
variable_sip_local_network_addr	-	-
variable_sip_local_sdp_str	-	-
variable_sip_network_ip	-	-
variable_sip_network_port	-	-
variable_sip_number_alias	-	-
variable_sip_outgoing_contact_uri	-	-
variable_sip_ph_P-Charging-Vector	-	-
variable_sip_profile_name	-	-
variable_sip_recover_contact	-	-
variable_sip_recover_via	-	-
variable_sip_reply_host	-	-
variable_sip_reply_port	-	-
variable_sip_req_port	-	-
variable_sip_to_port	-	-
variable_sip_use_codec_name	-	-
variable_sip_use_codec_ptime	-	-
variable_sip_use_codec_rate	-	-
variable_sip_use_pt	-	-
variable_sip_via_protocol	-	-

variable_switch_m_sdp	-	-
variable_transfer_history	-	-
variable_transfer_source	-	-
variable_uuid	-	-
variable_waitsec		
variable_waitmsec		
variable_waitusec		