

Contents

- [1 Introduction](#)
- [2 Codec Negotiation in FreeSWITCH](#)
 - ◆ [2.1 Early Negotiation \(default behavior\)](#)
 - ◇ [2.1.1 General principle](#)
 - ◇ [2.1.2 Early Negotiation parameters](#)
 - ◆ [2.2 Late Negotiation \(requires param\)](#)
 - ◆ [2.3 Proxy Media](#)
 - ◆ [2.4 Warnings](#)
- [3 Examples](#)
 - ◆ [3.1 Modifying the codec when using proxy media mode](#)
 - ◆ [3.2 Rewriting SDP](#)
 - ◆ [3.3 Disable G.729b on outbound](#)
 - ◆ [3.4 Codec Negotiation when proxy media enabled](#)
 - ◇ [3.4.1 Correct](#)
 - ◇ [3.4.2 Not correct](#)
- [4 See Also](#)

Introduction

Codec negotiation can be a confusing subject. If you are not familiar with [SDP](#) (Session Description Protocol) then this adds an extra layer of mystery. If you are new to the subject (or are just confused by what you've read and experienced) then consider this brief introduction and hopefully things will be a bit clearer.

First of all, what do we mean when we say "codec negotiation"? We are talking about the *process* of choosing which codec will be used on each leg of a call. FreeSWITCH supports a lot of codecs. Most SIP endpoints will also support multiple codecs. Each leg can have only one codec, so the codec negotiation process sifts through the choices and settles on a single codec to use. It's this "sifting process" that can cause confusion. How does this process work? How can it be modified? This page is written to help you answer those questions.

Codec Negotiation in FreeSWITCH

FreeSWITCH supports two basic modes of codec negotiation: early and late. *Early* negotiation means that the codec is negotiated between FreeSWITCH and the endpoint as soon as possible, even before FreeSWITCH needs to send media (such as ringing) or answer the call. This occurs before an incoming call even hits the dialplan. *Late* negotiation means to delay the choosing of the codec until after the call hits the dialplan and more information can be gathered. This additional information can be used to influence the negotiation process. Let's illustrate the differences between early- and late-negotiation.

Consider this setup:

- Alice has a phone with codecs G722 and PCMU

Codec_Negotiation

- Bob has a phone with PCMU and PCMA
- FreeSWITCH has an outbound "codec prefs" list of G722,G722.1,PCMU,PCMA,G729

(This is the list of codecs that FS will offer to all outbound calls)

Look at these two call flows. See if you can spot the difference in exactly when the codec on the A leg is negotiated:

- Early Negotiation

Alice (A) dials Bob (B)
A calls FS and offers two codecs: G722 and PCMU
FS sees these two codecs and chooses G722 immediately for the A leg
FS calls B and offers G722,G722.1,PCMU,PCMA,G729
B sees this codecs list and chooses PCMU
FS bridges A to B
FS is transcoding G722 to PCMU

- Late Negotiation with "inherit codec" set

Alice (A) dials Bob (B)
A calls FS and offers two codecs: G722 and PCMU
FS sees these two codecs but does *NOT* choose one just yet
FS calls B and offers G722 and PCMU
B sees these two codecs and chooses PCMU
FS now chooses PCMU on the A leg
FS bridges A to B
A and B are connected with PCMU to PCMU (no extra transcoding)

Note the difference in the timing of the negotiation of the A leg codec. With early negotiation, FreeSWITCH chooses a codec (G722) immediately. It does not wait to see what happens on the outbound leg (that is, the B leg) of the call. So the A leg is using G722. After deciding on the codec for the A leg, FreeSWITCH passes this call through the dialplan, which eventually comes to a bridge application to call Bob's phone. To initiate the bridge, FreeSWITCH calls the B leg and offers a pretty big list of codecs. These are the "outbound codec prefs" for FreeSWITCH. (This list is customizable, but more on that later.) Bob's phone sees all those codecs and chooses the first one that matches, in this case PCMU. At this point we have both call legs negotiated:

- A leg has G722
- B leg has PCMU

As soon as Bob's phone sends back a ringing signal, the two legs are bridged together. FreeSWITCH has to transcode from Alice's codec (G722) to Bob's codec (PCMU) because two different codecs were negotiated. Usually this is okay, but let's say you'd prefer that both phones have the same codec so that there's less CPU usage. This is where late-negotiation comes into play.

As you can see, when FreeSWITCH picks a codec for the A leg without knowing what codec is set on the B leg then there can easily be a "codec mismatch." A codec "mismatch" isn't always a bad thing, but many times it is nice to have both call legs share the same codec. Using late-negotiation and a technique called "inherit codec" we can force the A leg to use which ever codec is negotiated on the B leg. (This process is described in specific detail later on - for now we are only considering the basic concepts.)

The basic call flow in our second example has FreeSWITCH getting the A leg but *not* negotiating a codec right away. Instead, FreeSWITCH holds off on negotiating a codec for the A leg until after the A leg has

Codec_Negotiation

passed through the dialplan. In this case, the dialplan ended up with a bridge to Bob's phone. During the bridge process, FreeSWITCH negotiated a codec with Bob's phone - in this case PCMU. Once that codec was chosen on the B leg, FreeSWITCH went back to Alice's phone and told it that we would be using PCMU on the A leg. Now both call legs are using the same codec, which is what we wanted.

That's really all there is to the difference between early- and late-negotiation. Just keep this in mind: simply enabling late-negotiation does not automatically force the A leg to inherit the B leg's codec. It simply allows for "inherit codec" as one way of negotiating codecs. Late-negotiation allows for other codec negotiation tricks. Read on for a more specific description of codec negotiation in FreeSWITCH.

Early Negotiation (default behavior)

General principle

- When leg A calls FreeSWITCH, the offered codecs will be compared against the content of inbound-codec-prefs in the relevant SIP profile. As soon as a codec offered by A matches a codec allowed by FS, it is selected as the codec for leg A. If none of the offered codecs matches the allowed codecs, the call fails.

Remark: Because of this algorithm, the codecs order in the inbound SDP has priority on the codecs order in inbound-codec-prefs.

- When FS calls leg B, the list of codecs in outbound-codec-prefs for the SIP profile is reorganized by pushing the codec negotiated above for leg A at the top. If B does not accept any of the codecs, the call fails, obviously.

So for instance:

```
A ----- GSM/PCMA/G729 -----> FS (allowing G729/PCMA/PCMU) ----- PCMA/G729/PCMU ----->
```

What is happening:

- A proposes GSM/PCMA/G729 to FS.
- FS checks the proposed codecs in order of priority (as listed in the SDP) against its list of allowed codecs (as configured in inbound-codec-prefs), and selects PCMA as the first authorized codec, so the codecs list becomes: PCMA/G729/PCMU.
- FS proposes this codec list to B.

Early Negotiation parameters

disable-transcoding

This is a parameter you may set in the outbound SIP profile.

This will force the codec proposed to leg B (outbound leg) to be the same as the codec negotiated on leg A (inbound leg).

To set this, add the following line to the required SIP profile:

Codec_Negotiation

```
<param name="disable-transcoding" value="true"/>
```

Note: it is commonly misunderstood that this parameter disables the transcoding capability in FS. That is wrong.

This parameter just changes the outbound codec to match the one negotiated on the inbound leg so that no transcoding will be required.

The same result could be achieved by setting `absolute_codec_string` to the value of the inbound codec.

This parameter (only applicable when late-negotiation is off) will take the negotiated codec from leg A and offer it as the only option in leg B (ignoring any other codec settings) if the B leg cannot use that same codec the call will fail

absolute_codec_string

This is a channel variable you may set in the dialplan, typically just before bridging. This will force the codecs list proposed to leg B, without taking into account anything else. Here is an example:

```
<action application="export" data="nolocal:absolute_codec_string=PCMA,PCMU"/>
<action application="bridge" data="sofia/gateway/mygateway/mynumber"/>
```

or

```
<action application="bridge" data="{absolute_codec_string='PCMA,PCMU'}sofia/gateway/mygateway/mynumber"/>
```

Make sure you have single quotes ('PCMA,PCMU') around comma (',') separated list of codecs to protect it from parsing list of variables inside of
{var1=val1,var2=val2,absolute_codec_string='GSM,PCMU'}

Note: This parameter prevails on `disable-transcoding`. So if you disable transcoding in the outbound SIP profile and you use `absolute_codec_string` in the dialplan to set the outbound codec to be different from the inbound codec, transcoding will occur anyway.

codec_string

This is a channel variable you may set in the dialplan, typically just before bridging. The defined codec list will override the one set in the `outbound-codec-prefs` parameter of the outbound profile.

Here is an example:

```
<action application="export" data="nolocal:codec_string=PCMA,PCMU"/>
<action application="bridge" data="sofia/gateway/mygateway/mynumber"/>
```

or

```
<action application="bridge" data="{codec_string='PCMA,PCMU'}sofia/gateway/mygateway/mynumber"/>
```

Early Negotiation + Disable Transcoding:

Codec_Negotiation

```
<param name="disable-transcoding" value="true"/>
```

If this sofia profile param is set, all B legs that you invite will contain *only* the codec negotiated by the A leg.

Also in either case the variable "codec_string" on the A leg controls what codecs will be offered to the B leg.

The variable "absolute_codec_string" is similar except it implies the implicit list of codecs and will disable the addition of the A leg codec to the list as described in the default behavior.

Late Negotiation (requires param)

```
<param name="inbound-late-negotiation" value="true"/>
```

- The call will hit the dialplan without looking at the codecs at all.
- The negotiation will take place when leg A is answered.
- This allows you to route the call to a script and examine the sdp and rewrite the acceptable codecs with the "codec_string" channel variable.

inherit_codec

```
<action application="set" data="inherit_codec=true"/>
```

inherit_codec=true (only applicable when late-negotiation is enabled) will take the codec negotiated when the B leg answers and pass it to the A leg so it also uses the same codec (if the A leg supports it); otherwise, it will use whatever it can from its own list

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.

```
<action application="export" data="codec_string=${ep_codec_string}"/>
```

Proxy Media

Codec Negotiation can not be made if proxy media is enabled and answer application not executed.

Warnings

If your sdp contains codecs with different ptime preferences and ptime is specified in sdp, sofia will not send a ptime attr.

```
2010-10-18 11:47:52.234322 [WARNING] sofia_glue.c:213 Codec G723 payload 4 added to sdp wanting
```

Examples

Modifying the codec when using proxy media mode

Here is a dialplan example that will allow you to change the codec preference order and still enjoy the low cpu usage of proxy_media mode. It is similar to subst() in opener/opensips/kamailio textops module.

```
<context>
  <extension name="sdp_mangler">
    <!-- Set some defaults for this call -->
    <condition field="destination_number" expression="^\(d+$" break="on-false">
      <action application="set" data="transfer_to=${destination_number} XML fail"/>
    </condition>

    <!-- Here we check for G729 if found we are going make sure we have it first in preference order -->
    <condition field="${switch_r_sdp}" expression="/(.*)(m=audio \d+ RTP\AVP) (?=[ \d]+18|18[ \d]+)">
      <action application="set" data="codec_mangle= 18"/>
      <action application="set" data="transfer_to=${destination_number} XML public_proxy"/>
    </condition>

    <!-- Here we check for PCMU to add it back into the SDP if we want -->
    <condition field="${switch_r_sdp}" expression="/(.*)(m=audio \d+ RTP\AVP) (?=[ \d]+0|0[ \d]+)">
      <action application="set" data="codec_mangle=${codec_mangle} 0"/>
    </condition>

    <!-- Here we rebuild our SDP Moving G729 up front-->
    <condition field="${switch_r_sdp}" expression="/(.*)(m=audio \d+ RTP\AVP) ([ \d]+)(.*)/s" break="on-false">
      <action application="set" data="switch_r_sdp=${1}$2${codec_mangle} 101$4"/>
    </condition>

    <condition>
      <action application="transfer" data="${transfer_to}"/>
    </condition>
  </extension>
</context>
```

Rewriting SDP

switch_r_sdp:

```
<action application="set">
  <![CDATA[switch_r_sdp=(sdp here)
  ]]>
</action>
```

Disable G.729b on outbound

```
<extension name="disable-annexB" continue="true">
  <condition field="${switch_r_sdp}" expression="/(.*)(m=audio \d+ RTP\AVP) (.*) ( 18 ) (.*)/s">
    <action application="export" data="sip_append_audio_sdp=a=fmtp:18 annexb=no"/>
  </condition>
</extension>
```

Codec Negotiation when proxy media enabled

Correct

```
<extension name="test_proxy_media">  
  <condition field="source" expression="mod_sofia">  
    <action application="answer"/>  
    <action application="playback" data="ivr/ivr-welcome_to_freeswitch.wav"/>  
  </condition>  
</extension>
```

Not correct

```
<extension name="test_proxy_media">  
  <condition field="source" expression="mod_sofia">  
    <action application="playback" data="ivr/ivr-welcome_to_freeswitch.wav"/>  
  </condition>  
</extension>
```

See Also

- [Codecs](#)
- [Proxy Media](#)
- [Trans Coding Issues](#)
- [SDP Manipulation](#)
- [Codec Related in Channel Variables](#)