mod_dptools: bind_digit_action

0. About

Bind a key sequence or regular expression to an action to match on incoming DTMF tones.

1. Description

The `bind_digit_action` application is very similar to `mod_dptools: bind_meta_app` but is better, more flexible, and can be used to replace it in all cases.

> Be aware that when using `bind_digit_action` you may inadvertently break other apps that rely upon DTMFs. This is because any digits that match a pattern are *consumed* by `bind_digit_action`, and thus will not be sent through.
>
> However, **any non-matching digits will be sent through**. Therefore you should make sure that you don't have any regular expressions (or regexes) that "grab everything" unless you are equipped to handle that scenario.

`bind_digit_action` uses the concept of **realm**s for binding various **action**s.

- A **realm** is similar to a dialplan context where the dialed number means something different depending upon which context the call is in. (Could also be perceived as a state in a finite state machine.)
- You may bind one or more digits or a regular expression (also called regex) to an **action**.

> Note that you cannot "capture" digits in `bind_digit_action`.
>
> TODO Is this still valid? Section **3.4 Capture dialed digits** describes exactly this scenario, and there's even a channel variable for it.

2. Syntax

```
<action application="bind_digit_action" data="<realm>,<binding>,<command_string>[,<value>][,<dtmf_target_leg>][,
<event_target_leg>]"/>
```

Parameter	Description	Example
`<realm>`	Somewhat similar to a dialplan context (see XML Dialplan) or a state in a finite state machine. To clear or remove the digit bindings, use `mod_dptools: clear_digit_action`. To switch between realms, use `mod_dptools: digit_action_set_realm`.	`get_passcode`
`<binding>`	A `<binding>` can either be • a **sequence of digits to match** or • a **PCRE-compatible regular expression** that must start with a **~** TODO Only assuming PCRE-compatibility because other module descriptions mention it, and one of FreeSWITCH dependencies is the PCRE library.	`1234567` `~7\d{3}`

<table>
<tr><td><code><command_string></code></td><td><table>
<tr><th><code><command_string></code></th><th>Description</th></tr>
<tr><td><code>exec:<dialplan_app></code></td><td>To execute a dialplan application. This can have an optional flag 'H' which disables putting the other leg on hold during this operation. Can have an optional flag 'i' which makes the command blocking (it will execute in the thread of of the digit callback). Setting 'i' means 'H' has no effect.</td></tr>
<tr><td><code>api:<api_command></code></td><td>To issue a FreeSWITCH API command.</td></tr>
</table></td><td><code>exec:execute_extension,att_xfer XML default,both,self</code>

<code>exec[H]:execute_extension,start_record XML default</code>

<code>api:hupall</code></td></tr>
<tr><td><code><value></code></td><td>OPTIONAL Contains the arguments to the <code><command_string></code></td><td></td></tr>
<tr><td><code><dtmf_target_leg></code></td><td>OPTIONAL Possible values:
<code>self</code> (the default)
<code>peer</code>
<code>both</code>
Any other value is treated as <code>self</code>.</td><td><code>peer</code></td></tr>
<tr><td><code><event_target_leg></code></td><td>OPTIONAL See <code><dtmf_target_leg></code> above</td><td><code>self</code></td></tr>
</table>

Priority flag for <binding>parameter

TODO Write this up properly. The following text is directly from the commit comment.

<binding> can have a flag, 'P' (for "Priority") to bind_digit_action so that a match is returned as soon as it is found, without waiting for the inter-digit timeout to expire.

This can be very useful where the system needs to be more responsive to the user.

By default, if multiple bindings are enabled and one or more use a regex, switch_ivr_dmachine_check_match waits for the inter-digit timeout to expire before returning a match. This ensures overlapping patterns, such as "^\d{4}$" and "^\12{3}$" can both be reliably matched

When the 'P' flag is specified with bind_digit_action, whose action is exec, a match is returned as soon as the condition is satisfied, regardless of whether or not a longer match may be possible if further digits were entered.

For example:

<action application="bind_digit_action" data="test,~^*\d{2}$, exec[P]:execute_extension,myextn,self,self"/>

<action application="bind_digit_action" data="test,~^#\d{2}$", exec:execute_extension,myotherextn,self,peer"/>

The first example causes a match to be returned immediately after the 2nd digit is received, whereas the second example defaults to waiting for the inter-digit timeout to expire before returning.

In cases where the 'P' flag is used with a regex and string, and both are matched, the more explicit, string match will be returned.

For example:

<action application="bind_digit_action" data="test,~^*\d{2}$, exec[P]:execute_extension,myextn,self,self"/>

<action application="bind_digit_action" data="test,*12, exec[P]:execute_extension,myotherextn,self,self"/>

If "*12" is matched, myotherextn is executed, because "*12" is more explicit/specific than "^*\d{2}$"

If the 'P'(riority) flag is not used, behaviour is unchanged from previous versions. This ensures backward compatibility.

3. Examples

3.0 Usage

```
<action application="bind_digit_action" data="my_digits,11,exec:execute_extension,att_xfer XML default,both,
self"/>
<action application="bind_digit_action" data="my_digits,11,api:hupall"/>
```

3.1 Two-realm example 1

TODO Is this right? "*The initial realm is `test1` (set by `mod_dptools: digit_action_set_realm`), and realm `myrealm` is never used, but shows how to use multiple realms.*"

```
<action application="bind_digit_action" data="myrealm,500,exec:playback,ivr/ivr-welcome_to_freeswitch.wav"/>
<action application="bind_digit_action" data="test1,456,exec:playback,ivr/ivr-welcome_to_freeswitch.wav"/>
<action application="bind_digit_action" data="test1,##,exec:execute_extension,mix_welcome_to_freeswitch"/>
<action application="digit_action_set_realm" data="test1"/>
```

3.2 Two-realm example 2

- The initial realm is `cool` (set by `mod_dptools: digit_action_set_realm`),
- dialling 500 or any number starting with 7 will start playing the `ivr/ivr-welcome_to_freeswitch.wav` file, and
- any sequence of numbers that start with the digit 1 will switch between realms (or contexts or states).

```
<action application="bind_digit_action" data="cool,500,exec:playback,ivr/ivr-welcome_to_freeswitch.wav"/>
<action application="bind_digit_action" data="cool,~7\d{3},exec:playback,ivr/ivr-welcome_to_freeswitch.wav"/>
<action application="bind_digit_action" data="cool,~1\d+,exec:digit_action_set_realm,rad"/>
<action application="bind_digit_action" data="rad,~1\d+,exec:digit_action_set_realm,cool"/>
<action application="digit_action_set_realm" data="cool"/>
```

3.3 Call Recording

TODO From the comment section:

Denis 2016.01.26

Call Recording On/Off Switch - work only with 1 bind, you can not make several binds - after enable|disable record and switch context we got new bindings with new realm.

The following dialplan example demonstrates how to use `bind_digit_action` to create an on/off sequence for call recording. We'll use **`*2`** as the key sequence that toggles the record, but you can use whichever key sequence you see fit.

We need three different extensions for this operation:

- The `SETUP_RECORDING` extension
- The `START_RECORDING` extension
- The `STOP_RECORDING` extension

3.3.1 Extension definitions

SETUP_RECORDING extension

```
<extension name="setup bind_digit_action_recording">
  <condition field="destination_number" expression="^SETUP_RECORDING$">
    <action application="log" data="INFO Configuring bind_digit_action to do recording on this session..."/>
    <action application="bind_digit_action" data="start_recording,*2,exec:execute_extension,START_RECORDING XML
default"/>
    <action application="bind_digit_action" data="stop_recording,*2,exec:execute_extension,STOP_RECORDING XML
default"/>
    <action application="digit_action_set_realm" data="start_recording"/>
  </condition>
</extension>
```

START_RECORDING extension

```
<extension name="bind_digit_action Start Recording">
  <condition field="destination_number" expression="^START_RECORDING$">
    <action application="log" data="INFO Starting recording..."/>
    <action
      application="set"
      inline="true"
      data="rec_file=$${base_dir}/recordings/${strftime(%Y-%m-%d-%H-%M-%S)}_${destination_number}
_${caller_id_number}.wav"
    />
    <action application="record_session" data="${rec_file}"/>
    <action application="digit_action_set_realm" data="stop_recording"/>
  </condition>
</extension>
```

STOP_RECORDING extension

```
<extension name="bind_digit_action Stop Recording">
  <condition field="destination_number" expression="^STOP_RECORDING$">
    <action application="log" data="INFO Stop recording [${rec_file}]"/>
    <action application="stop_record_session" data="${rec_file}"/>
    <action application="digit_action_set_realm" data="start_recording"/>
  </condition>
</extension>
```

3.3.2 Application

Now we need to apply this to our inbound and outbound calls.

3.3.2.1 Outbound calls

Outbound calls (A leg) are simple, just add an execute extension:

Outbound

```
<!-- ... -->
<action application="execute_extension" data="SETUP_RECORDING XML default"/>
<action application="bridge" data="<your bridge stuff here>"/>
<!-- ... -->
```

3.3.2.2 Inbound calls

For inbound calls (or the B leg) you need to do a little bit more.

Add this to the Local_Extension after the `bind_digit_action` calls:

```
<!-- ... -->
<action application="set" data="bridge_pre_execute_bleg_app=execute_extension"/>
<action application="set" data="bridge_pre_execute_bleg_data=SETUP_RECORDING XML default"/>
<!-- ... -->
```

3.4 Capture dialed digits

Sometimes it is useful to know what digit(s) the caller dialed, especially when using a regular expression to capture digits. This is done with a channel variable called `last_matching_digits` that is updated with each match.

This dialplan snippet demonstrates how it works:

```
<extension name="Bind a regex">
  <condition field="dialed_number" expression="^(9921)$">
    <action application="bind_digit_action" data="my_digits,~^\d+,exec:execute_extension,LOG_DIGITS XML default"
/>
    <action application="digit_action_set_realm" data="my_digits"/>
  </condition>
</extension>

<extension name="Display digits dialled">
  <condition field="dialled_number" expression="^LOG_DIGITS$">
    <action application="log" data="INFO Called dialled ${last_matching_digits}"/>
    <action application="say" data=" en number iterated ${last_matching_digits}"/>
  </condition>
</extension>
```

3.5 Lua example

While on a call, you can dial 555, 556, or 557.

Put this in your dialplan extension "Local_Extension":

```
 <action application="lua" data="test.lua"/>
```

and this in ${scripts_dir}/test.lua file

```
wav1 = "ivr/ivr-welcome_to_freeswitch.wav"
wav2 = "ivr/ivr-you_are_number_one.wav"

session:execute("bind_digit_action","myrealm,555,exec:playback," .. wav1);

session:execute("set","playback_delimiter=#");
session:execute("set","playback_sleep_val=100");
session:execute("bind_digit_action","myrealm,556,exec:playback," .. wav1 .. "#" .. wav2);

session:execute("bind_digit_action","myrealm,557,exec:playback,file_string://" .. wav1 .. "!" .. wav2);
```

4. Channel Variables

See Channel Variables Catalog to see all channel variables.

Channel variable name	Description

`bind_digit_digit_timeout`	INTEGER
	
	Inter-digit timeout, in milliseconds. This sets the time to wait between individual dialed digits.
	
	Default value is 1500 milliseconds.
	
	Mostly only useful in conjunction with `mod_dptools: play_and_get_digits` in the same dialplan extension.
	
	See section **3.1** in `mod_dptools: bind_digit_action`.
	
	## Usage
	
	Example
	
	`<!-- To wait 10 seconds between digits: -->`
	`<execute application="set" data="bind_digit_digit_timeout=10000" />`

`bind_digit_input_timeout`	INTEGER Overall timeout, in milliseconds. This sets the overall time to wait for the entire digit sequence to be entered. Default value is 0 milliseconds. Mostly only useful in conjunction with `mod_dptools: play_and_get_digits` in the same dialplan extension. See section **3.1** in `mod_dptools: bind_digit_action`. ## Usage **Example** `<execute application="set" data="bind_digit_input_timeout=10000" />`

4.1 Example use case: using `mod_dptools: play_and_get_digits` and `mod_dptools: bind_digit_action` together

TODO Still not sure why one would need to use these two together.

Quoting from [Freeswitch-users] problem with "play_and_get_digits" command (DTMF delay time):

Manish Talwar manish.talwar at nexxuspg.com
Tue Aug 25 16:56:25 MSD 2015

Hi,

I have found one small problem with "play_and_get_digits" command.

We have a IVR application and I am trying to get either "16 digit" card number or "0" for move to customer support from the user.

Please find my dialplan as follows: 2015-08-25 18:05:37.535806 [CRIT] mod_httapi.c:1148 Debugging Return Data:

```
<document type="xml/freeswitch-httapi">
<params>
</params>
<work>
<execute application="log" data="card number inputed: 2"/>
<execute application="flush_dtmf"/>
<execute application="sleep" data="2000"/>
<execute application="clear_digit_action" data="my_digit" />
<execute application="bind_digit_action" data="my_digit,~\d{16}|^[0]{1},exec:transfer,dummy_transfer" />
<execute action="http://localhost:8080/ivr/fsentercardnumber/response/" application="play_and_get_digits"
data="0 16 1 1000 # en-US/enter_card_number.mp3 '' Digits '' 1000 '' " />
<getVariable name="Digits"/>
<execute action="http://localhost:8080/ivr/fsentercardnumber/response/" application="play_and_get_digits"
data="0 16 1 20000 # en-US/menu_speak_csr.mp3 '' Digits '' 20000 '' " />
<getVariable name="Digits"/>
<execute application="clear_digit_action" data="my_digit" />
</work>
</document>
```

It is working fine as expected if user input 16 digit number continuously without any delay. After getting full 16 digit number it will match the "bind_digit_action" and move to other dialplan. But my problem is if a user enter card number slowly then its trying to match already inputed DTMF values with matching binding and received next inputed value as different DTMF values. As a example, if I tried to input 55 and then wait for 1-2 seconds then Freeswitch tried to match 55 with binding.

```
2015-08-25 18:05:39.535920 [DEBUG] switch_ivr_play_say.c:1305 Codec Activated L1 6 at 8000hz 1 channels
20ms
2015-08-25 18:05:39.755933 [DEBUG] switch_rtp.c:5819 RTP RECV DTMF 5:2080
2015-08-25 18:05:40.015948 [DEBUG] switch_rtp.c:5819 RTP RECV DTMF 5:2080
2015-08-25 18:05:41.536035 [DEBUG] mod_dptools.c:132 sofia/internal/18188535351@ 192.168.1.112 Digit NOT
match binding [55]
2015-08-25 18:05:41.536035 [DEBUG] switch_channel.c:486 RECV DTMF 5:2000
2015-08-25 18:05:41.536035 [DEBUG] switch_channel.c:582 sofia/internal/181885353 51 at 192.168.1.112
Queue dtmf digit=5 ms=250 samples=2000
2015-08-25 18:05:41.536035 [DEBUG] switch_channel.c:486 RECV DTMF 5:2000
2015-08-25 18:05:41.536035 [DEBUG] switch_channel.c:582 sofia/internal/181885353 51 at 192.168.1.112
Queue dtmf digit=5 ms=250 samples=2000
```

I don't want fresswtich tried to match the binding if there is any pause time while inputed the number as user can type slowly. I want to reset this binding delay time so that user can type slowly and we will get full 16 digits number for matching inputed value in certain time interval. Please suggest me how can I resolve this problem and increase the DTMF delay time. Thanks, Regards, Manish Talwar

Michael Collins msc at freeswitch.org
Wed Aug 26 18:42:32 MSD 2015]

Is there a reason that you're trying to use both bind_digit_action and play_and_get_digits? Normally you'd use just one or the other, and if you're explicitly asking the user for input then I'd think that you would only need play_and_get_digits.

-MC

Manish Talwar manish.talwar at nexxuspg.com
Thu Aug 27 09:16:24 MSD 2015

Hi,

Thank you for your response. Actually, we are using multiple play_and_get_digits in a single dialplan. And when user inputed any matching format of input then we need to leave all remaining play_and_get_digits commands of that dialplan. That is why I used bind_digit_action also in same dialplan.

Please suggest how do we make the changes for this concern, how do we break the dialplan when required matching input found. Basically, its works fine as our requirement but we need to increase the DTMF delay time only, whenever user inputed slowly then its take the chunk of inputed data and tried to match the bind_digit_action expression. As its break the DTMF into many chunks so its not matching expected regular expression of command. Please suggest. Thanks,

Regards,

Manish Talwar

Michael Collins msc at freeswitch.org
Fri Aug 28 21:45:06 MSD 2015

```
I found this in mod_dptools.c inside the bind_to_session function:

    if (!(dmachine = switch_core_session_get_dmachine(session, target))) {
        uint32_t digit_timeout = 1500;
        uint32_t input_timeout = 0;
        const char *var;

        if ((var = switch_channel_get_variable(channel,
"bind_digit_digit_timeout"))) {
            digit_timeout = switch_atoul(var);
        }

        if ((var = switch_channel_get_variable(channel,
"bind_digit_input_timeout"))) {
            input_timeout = switch_atoul(var);
        }

        switch_ivr_dmachine_create(&dmachine, "DPTOOLS", NULL,
digit_timeout, input_timeout, NULL, digit_nomatch_action_callback, session);
        switch_core_session_set_dmachine(session, dmachine, target);
    }


It appears that there might be two channel variables for you to try:
bind_digit_digit_timeout
bind_digit_input_timeout

These appear not to be documented at present, so I think you can help pay
it forward by tinkering with them and then reporting back what you find
out. As a guess, I'd say you're running into an issue with people who spend
more than 1.5 seconds between digits. (It appears the default is 1500ms.)
Try setting the bind_digit_digit_timeout value in the dialplan. Set it to
something absurdly long like 7500 so that you can know for sure it is
working.

Please report back with your findings so that the docs team can document
this. Bonus points if you document it yourself. :)

Hope this helps,
Michael
```

Manish Talwar manish.talwar at nexxuspg.com
Mon Aug 31 14:34:30 MSD 2015

```
Hi Michael,

Thanks for your help.

Its working fine now with setting the value of "bind_digit_digit_timeout" and "bind_digit_input_timeout"
channel variable. I have set it as 10000 and its wait till 10 sec now for matching the regular expression.

<execute application="set" data="bind_digit_digit_timeout=10000" />

<execute application="set" data="bind_digit_input_timeout=10000" />

Its working fine with me, your docs team can document this if required.

Thanks,


Regards,

Manish Talwar
```

5. Notes

5.1 Using `bind_digit_action` with a conference

Be aware that when using `bind_digit_action` you may inadvertently break other apps that rely upon DTMFs. This is because any digits that match a pattern are *consumed* by `bind_digit_action`, and thus will not be sent through.

However, **any non-matching digits will be sent through**. Therefore you should make sure that you don't have any regular expressions (or regexes) that "grab everything" unless you are equipped to handle that scenario.

The `bind_digit_action` can definitely be used with conferences, however the above rule about matching digits still applies. Make sure that any keys that you wish to be available to your callers to control the conference (i.e. the conference caller-controls) do not "match" in your `bind_digit_actions`. There should be no overlap or overload of defined digits.

If you have a PIN on your conference then you will need to make sure that the PIN code also does not "match" any of your bound digit actions, otherwise `bind_digit_action` will consume the digits that the caller dials and will *not* send them on to the conference! This results in a caller not being able to join a conference that is locked with a PIN code.

5.2 Inband vs. 2833 DTMFs

The `bind_digit_action` supports both inband and 2833 DTMFs. For an example on how to check the SDP for RFC 2833 and automatically start in-inband dtmf detection look in conf/dialplan/default.xml and locate the "global" extension. The 2833 detection is commented out by default.

TODO RFC 2833 is obsoleted by RFC 4733.

5.3 Setting on the B leg

TODO How does this relate to section 3.3? Didn't touch this section.

Here are several choices:

Set the dtmf target leg to "peer" (see above)

These 2 vars on the A leg using the set app:

```
<action application="set" data="bridge_pre_execute_bleg_app=bind_digit_action"/>
<action application="set" data="bridge_pre_execute_bleg_data=whatever"/>
```

This var with export app:

```
<action application="export" data="nolocal:execute_on_answer=bind_digit_action whatever"/>
```

These vars with export app:

```
<action application="export" data="nolocal:bridge_pre_execute_app=bind_digit_action"/>
<action application="export" data="nolocal:bridge_pre_execute_data=whatever"/>
```

or one of these vars in the dial string {}:

```
{bridge_pre_execute_app=bind_digit_action,bridge_pre_execute_data='whatever'}
{execute_on_answer='bind_digit_action whatever'}
```

6. See Also

- `mod_dptools: digit_action_set_realm` - switch realms
- `mod_dptools: clear_digit_action` - clear digit binding
- `last_matching_digits` channel variable