mod_dptools: bind_digit_action

About

Bind a key sequence or regex to an action.

Description

The bind_digit_action application is very similar to bind_meta_app but is better and more flexible. It can be used to replace bind_meta_app in all cases.

bind_digit_action uses the concept of ''realms'' for binding various actions. A realm is similar to a dialplan context where the dialed number means something different depending upon which context the call is in. (See the example below.)

You may bind one or more digits to an action, and you may also use [[regex|regular expressions]]. Note that you cannot "capture" digits in bind_digit_action,.

To clear or remove the digit bindings, use clear_digit_action.

Usage

```
<action application="bind_digit_action" data="<realm>,<digits|~regex>,<string>[,<value>][,<dtmf target leg>][,
<event target leg>]"/>
```

realm — the digit realm, somewhat similar to a dialplan context

digits — the digits to match

OR

~regex — the regular expression to match

string — the dialplan or api command, e.g. "exec:<dialplan app>", e.g. "api:<api app>"

Optional:

value — contains the arguments to the command string

dtmf target leg — aleg, bleg, peer, both; defaults to aleg

event target leg — aleg, bleg, peer, both; defaults to aleg

Channel Variables

bind_digit_digit_timeout

Interdigit timeout, in milliseconds. Default = 1500 msec. This sets the time to wait between individual dialled digits. Mostly only useful in conjunction with play_and_get_digits in same dialplan extension.

Example: set to 10000 to wait 10 seconds between digits.

bind_digit_input_timeout

Overall timeout, in milliseconds. Default = 0 msec. This sets the overall time to wait for the entire digit sequence to be entered.

Notes

```
<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"/>
```

Setting on the B leg

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'}
```

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 regexes that "grab everything" unless you are equipped to handle that scenario.

Using bind_digit_action with a conference

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.

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.

Examples

Simple Example With Two 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"/>
```

-- or --

```
<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"/>
```

Advanced Example: Call Recording On/Off Switch

The following 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. 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

SETUP RECORDING

```
<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 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 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>
```

Now we need to apply this to our inbound and outbound calls. Outbound calls (A leg) are simple, just add an execute extension:

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

For inbound calls (or the B leg) you need to do a little bit more. Add this to the Local_Extension after the bind_meta_app calls:

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

Note that in this example we have taken '"*2"' away from bind_meta_app.

Catching the Digits Dialled

Sometimes it is useful to know what digit(s) the caller dialled, especially when using a regular expression to capture digits. This is done with a channel variable called [[Variable_last_matching_digits|last_matching_digits]]. The channel variable 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>
```

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"/>
```

put 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);
```

See Also

- mod_dptools: digit_action_set_realm
- [[Variable_last_matching_digits]]
- clear_digit_action— clear digit bindings