Contents

- 1 Description
- 2 Usage
  - 2.1 Inline Execution
- 3 Examples
- 4 bind_meta_key variable
- 5 See Also

Description

bind_meta_app binds an application to the specified call leg(s). During a bridged call, the DTMF sequence on the bound call leg will trigger the execution of the application. The call leg that is **not** bound will **not** hear the DTMF sequence being dialed. Once bound to a call leg, the application binding will survive for the lifetime of the call leg.

Important: This feature will not work when bypass_media=true, because the endpoints will be communicating directly with each other, and the key presses will not be sent to FreeSWITCH.

Key can also only be 0-9 ... * or # will revert to 0

For a more advanced digit binding, see the bind_digit_action application.

Usage

```
<action application="bind_meta_app" data="KEY LISTEN_TO FLAGS APPLICATION[::PARAMETERS]"/>
```

Explanation of parameters

- KEY is the button you want to respond to after the * button is pressed. If you wanted to respond to *1, you would put 1 in place of KEY. You are limited to a single digit.
- LISTEN_TO is which call leg(s) to listen on. Acceptable parameters are **a**, **b** or **ab**.
- FLAGS modifies the behavior. The following flags are available:
  - a - Respond on A leg
  - b - Respond on B leg
  - o - Respond on opposite leg
  - s - Respond on same leg
  - i - Execute inline (see below)
  - 1 - Unbind this meta_app after it is used one time
- APPLICATION is which application you want to execute.
- PARAMETERS are the arguments you want or need to provide to the APPLICATION. You must put :: after the APPLICATION for these arguments to be handled properly.

After the * button is pressed, the KEY event is expected within 5 seconds. In case of timeout, the KEY is passed further to the B leg.

Inline Execution

The **i** flag means to execute the application inline, that is, immediately. Normally the application is queued up to be executed in the session's thread asynchronously, similar to how ESL queues up and application.

Examples

```
<action application="bind_meta_app" data="1 a s execute_extension::dx XML features"/>
```

When *1 is pressed on the A call leg, the execute_extension application is executed upon the A call leg. The extension that is executed is the dx extension in the XML dialplan under the features context.

```
<action application="bind_meta_app" data="2 a s record_session::$${base_dir}/recordings/${caller_
```

When *2 is pressed on the A call leg, the session audio starts recording on the A call leg. It saves the audio to the $$(base_dir)/recordings/${caller_id_number}.$(strftime(%Y-%m-%d-%H-%M-%S)}.wav file. (i.e. /usr/local/freeswitch/recordings/1234.2008-04-09-10-11-12.wav). **Important note**: In default diaplan example record_session is listening for DMTF on the B leg, so callee can only activate the recording.

bind_meta_key variable

You may also customise the meta key. It will default to '*' but you can use the bind_meta_key channel variable to select a different meta key:

```
<action application="set" data="bind_meta_key=#"/>
<action application="bind_meta_app" data="2 a s foo"/>
```

See Also

- Dialplan
- Misc. Dialplan Tools bind digit action