Extensions.conf

Synopsis

The extensions.conf file lays out the dialplan, bringing channels together with applications and services. Extensions.conf features extension matching logic and intelligent call routing logic.

Arrangement

The /etc/asterisk/extensions.conf file consists of a number of special keywords, followed by arguments presented in syntax specific to those keywords. It is structured by *context,* with each context in a specific section. Each context may contain a number of *extensions* that route calls to specific channels, applications, or other extensions within local or remote dialplans.

Contexts

Contexts are begun by placing them within brackets, on a line by themselves. All following keywords are considered to be in that context until the context is changed with another statement.

Example

```
[context]
keyword => values
keyword => values

[newcontext]
keyword => values
keyword => values
```

Each channel has an initial context (defined in it's channel configuration file.) Incoming channels can only access extensions placed in their default context. Exceptions to this can be made using the ***include*** keyword or by calling the application 'Goto' within an extension.

Keywords

The following keywords are available in /etc/asterisk/extensions.conf.

include: Includes one context within another. When a context has been included, all extensions in that context are made available to the including context. This allows a layered permissions structure for the dialplan. Include also offers the ability to include another context during a particular time. Time dependent includes can be used to include a context based on the time of day, the day of the week, the day of the month, or the month of the year. The format for a time based include is:

```
include => context|days|times|days of month|months of year
```

All conditions must be met for the include. If you specify a range of days and a range of times, it must be both during those days and during those hours for the context to be included. You can specify that a particular field be ignored by making it a '*****' . Arguments can be numeric or in common (US) abbreviations.

Examples

To include a context at all times:

```
include => provider
```

To include a context between Monday and Friday, 8 am to 5 pm.

```
include => provider|Mon-Fri|8am-5pm|*|*
```

exten: Defines an extension. Extensions may be a single line or may span multiple *exten* definitions, each indicating another step in the call' s progress.*Exten* takes a specific syntax. Each *exten* definition calls an application, optionally with arguments specific to that application.

Asterisk defines a few special extensions, which have significant meaning within the dialplan. These extensions are used to handle certain call conditions, including timing out waiting for information, receiving invalid (unknown) extensions, beginning a call, and receiving a request for operator (when the caller presses zero).

These special extensions are:

s: Defines how to route a call when no other routing information has been received. On a PRI or local FXS line, we will receive a number string to route the call by. When receiving a call from an analog (FXO) line, we won' t get any routing information, just a ring. In this case the *'s'* extension will be used.

t: Defines what to do when a call has timed out waiting for routing information. If Asterisk is waiting for digits to route a call by (such as when the *'s'* extension offers a menu of extensions and no input is received.)

i: Defines how to handle an invalid or unroutable extensions, such as the cases where an extension is dialed that does not exist in the context, or when an extension is partially dialed but times out without enough information to complete the call.

o: Defines how to handle a request for operator assistance. This extension is used when a caller presses zero during the Voicemail Application.

These extensions are configured like all others. Use the extension letter for the number field.

All single-letter extensions should be considered reserved. ' #' and ' *' can be used in

extensions, for example to put VoicemailMain on #65 or similar.

An extension definition is broken down into four parts. Each extension has a *number*, which is the number of the extension. This number may be received from the caller, received from a PRI (in the case of a DID) or pointed to from other extensions with the application ' Goto.'

Following the *number* is the *priority.* Each extension consists of one or more priorities. Priority are steps in the extension. They are generally executed in numeric order, though applications can change this. Priorities also offer the ability to handle a call differently if the ' Dial' application returns busy when attempting to ring a channel. If the channel is busy, Asterisk will, instead of going to the next priority, attempt go to the next priority plus 100. If priority 1 is a ' Dial' and the channel is busy, Asterisk will move instead to priority 102 for that extension, if it exists. If the busy path cannot be found, the Asterisk will fall back to the priority plus one.

The third part is the application to call. See the application section for more information about the available applications.

The fourth part is special argument to be passed to the applications. Some applications (such as VoicemailMain) take no arguments. In this case, the fourth section can be omitted.

A complete extension definition should look like this.

```
exten =>  number,priority,application[,arguments]
```

Examples

This attempts to ring a Zaptel channel, and offers two different paths, one if the Dial application times out and the other if it' s busy.

```
exten => 6161,1,Dial,Zap/1|20
exten => 6161,2,Voicemail,u6161
exten => 6161,102,Voicemail,b6161
```

This calls an application with no arguments.

```
exten => 8500,1,VoicemailMain
```

switch: A switch statement allows an Asterisk server to search for extensions outside of the local dialplan. The only currently implemented use is the IAX switch, though other uses of this feature are planned. The IAX switch uses the IAX protocol to find extensions on a remote dialplan and connect them. If a caller dials an extension that is not accesable in the local dialplan, and a switch statement is present in the callers present context, the extension will be tried against the remote server or servers specified. The switch statement includes as an argument a context, which will be the context within the remote dialplan that will be searched.

Examples

To look for a remote extension using the IAX protocol:

```
switch => IAX/username:password@server/context
```

Alternately, if RSA is turned on (see iax.conf) you may replace the password with a keyname, which will be the key sent to the remote server.

```
switch => IAX/username:[password]@server/context
```

Complete File Example

The following is a complete short example of an /etc/asterisk/extensions.conf file:

```
[provider]
switch => IAX/bob:passme@misery/local

[default]
exten => 6161,1,Dial,IAX/suid|20
exten => 6161,2,Voicemail,u6161
exten => 6161,102,Voicemail,b6161

exten => 8500,1,VoicemailMain

[local]
include => provider
include => default

[incoming]
include => default
exten => s,1,Playback,welcome
exten => s,2,Background,Menu

exten => 1,1,Goto,default|6161|1
exten => 2,1,Goto,default|6161|2

exten => t,1,Goto,s|2
exten => i,1,Goto,s|2
exten => o,1,Goto,default|6161|1
```