Extensions

Une **extension** est une **instruction** du plan de numérotation exécutée pour le poste ou **groupe de postes** destinataire(s) correspondant à son **nom**.

Ce **nom** est défini par une **séquence de touches** (numéro de téléphone ou alphanumérique).

Syntaxe:
```
exten => nom_extension
```

Syntaxe complète de l'instruction:
```
exten => nom_extension, priorité, application()
```

Exemple: répondre à l'appel numérotant le poste 123
```
exten => 123, 1, Answer()
```

Exemple: répondre à l'appel numérotant tout poste commençant par 123 et comportant 5 chiffres
```
exten => _123XX, 1, Answer()
```

- Les protocoles de VoIP tels H323 ou SIP encouragent l'emploi de noms d'extensions alphanumériques ou adresses de messagerie.

Extensions non spécifiques

Certaines extensions nommées ne sont pas spécifiques à un terminal, ni même un groupe de terminaux.

"s": start

L'**extension "s"**, pour début (*start*), gère automatiquement **tous** les **appels entrants** dans un contexte sans extension de destination spécifique (souvent utilisé dans les macros).

Exemple: sur tout appel entrant, décrocher et jouer le fichier son demandant de numéroter une extension

```
exten => s, 1, Answer()
exten => s, 2, Background(enter-ext-of-person)
```

"i": invalid

L'**extension "i"**, pour **entrée invalide** (*invalid*), recevra tous les appels dont l'extension numérotée ne figurera pas *dans le menu d'un serveur vocal*.

Exemple: sur entrée invalide, joue le fichier son correspondant et raccroche

```
exten => i, 1, Playback(pbx-invalid)
exten => i, 2, Hangup()
```

"t": timeout

L'**extension "t"**, pour **délai dépassé** (*timeout*), sera appelée lorsque l'appelant aura mis trop de temps à composer une entrée demandée par une application (exemple: `Background()`). La temporisation par défaut est de **10 secondes**.

Exemple: à la chute du délai de temporisation, jouer le fichier son vm-goodbye et raccrocher

```
exten => t, 1, Playback(vm-goodbye)
exten => t, 2, Hangup()
```

Priorités

La **priorité** définit l'**ordonnancement des instructions** relatives à **une extension**. Elles sont numérotées **à partir de 1** dans un **ordre croissant**.

➢En cas de "trou" dans la numérotation des priorités, le PBX Astérisk arrêtera l'exécution des instructions.

Exemple: répondre sur la numérotation de 123 puis raccrocher

```
exten => 123, 1, Answer()
exten => 123, 2, Hangup()
```

Les priorités peuvent comporter du **texte indicatif** entre parenthèses, qui servira de nom de référence: ce sont des **priorités nommées**:

```
exten => 123, 1(décrocher), Answer()
exten => 123, 2(raccrocher), Hangup()
```

Les **priorités** peuvent être **non numérotées**: la **priorité n** (next) prendra le numéro incrémenté de 1 de la priorité précédente, rendant le plan de numérotation plus souple.

Exemple: répondre au canal qui sonne sur numérotation de 123, établir la communication vers le terminal SIP d'extension 123 puis raccrocher en fin de communication

```
exten => 123, 1, Answer()
exten => 123, n, Dial(SIP/123)
exten => 123, n, Hangup()
```

Applications

Une **application** exécute une **action spécifique** sur le **canal en cours**. Analogue à une fonction, elle peut recevoir des **arguments** entre **parenthèses** "`()`" séparés par des virgules ",".

Syntaxe:
```
application(argument1, argument2,...)
```

La plupart des applications ne fonctionne que si le canal *n'est pas* en train de sonner. Il faut donc au préalable faire *répondre* Asterisk (`Answer()`), qui crée alors le demi-canal appelant-proxy.

Tout plan de numérotation devrait donc débuter ainsi:

```
[incoming]
exten => s, 1, Answer()
exten => s, n, application2()
exten => s, n, application3()

exten => 123, 1, application1()
...
```

La liste des applications Asterisk est accessible en ligne de commande:
```
~# asterisk -r
CLI> core show applications
CLI> core show application Dial                    ;Manuel de l'application Dial (sensible à la casse)
```

Exemple d'appel

Tout terminal téléphonique VoIP (ipphone, softphone) devant s'inscrire sur un proxy pour passer des appels, devra nécessairement être déclaré dans la configuration d'Asterik.

Dans le cas où l'on dispose de deux ipphones SIP, on renseignera le fichier */etc/asterisk/sip.conf* avant d'écrire le plan de numérotation dans le fichier */etc/asterisk/extensions.conf* selon les règles précédentes.

Exemple de configuration des ipphones uid01 et uid10 pour des appels bidirectionnels:

/etc/asterisk/sip.conf:

```
[uid01]                          ;Nom de compte utilisé pour l'enregistrement du téléphone
type=friend                      ;Support bidirectionnel des appels
allow=all                        ;Liste des codecs autorisés
username=station01               ;Nom d'utilisateur en cas d'authentification
secret=bonjour                   ;Mot de passe à l'enregistrement
qualify=yes                      ;Détermination de la latence réseau par Asterisk
host=dynamic                     ;Adressage IP dynamique
context=from_01                  ;Contexte du plan de numérotation traitant les appels de uid01

[uid10]
type=friend
allow=all
username=station10
secret=bonjour
qualify=yes
host=dynamic
context=from_10                  ;Contexte du plan de numérotation traitant les appels de uid10
```

Exemple d'appel

Exemple de plan de numérotation gérant les numéros à deux chiffres:

/etc/asterisk/extensions.conf:

```
[from_01]                                   ;Déclaration du contexte de uid01
include => interne                          ;Inclusion du contexte de numérotation interne

[from_10]                                   ;Déclaration du contexte de uid10
include => interne                          ;Inclusion du contexte de numérotation interne

[interne]                                   ;Déclaration du contexte interne
exten => _XX, 1, Answer()                   ;Asterisk établit le demi-canal appelant-proxy
exten => _XX, n, Dial(SIP/uid${EXTEN},30,gjr)   ;Numérotation du canal SIP référencé 'uidXX'.
                                            ;Le destinataire est connu grâce à la variable ${EXTEN}
exten => _XX, n, Hangup()                   ;L'appel se termine, on raccroche
```

Visualisation de l'appel uid01 vers uid10 dans la console Asterisk:

```
~# asterisk -r
CLI> core set verbose 3
        -- Executing [10@from_01:1] Answer("SIP/uid01-08b054b8", "") in new stack
        -- Executing [10@from_01:2] Dial("SIP/uid01-08b054b8", "SIP/uid10|30|gjr") in new stack
        -- Called uid10
        -- SIP/uid10-08b071e8 is ringing
        -- SIP/uid10-08b071e8 answered SIP/uid01-08b054b8
        -- Native bridging SIP/uid01-08b054b8 and SIP/uid10-08b071e8
        -- Executing [10@from_01:3] Hangup("SIP/uid01-08b054b8", "") in new stack
== Spawn extension (from_01, 10, 3) exited non-zero on 'SIP/uid01-08b054b8'
```