

# Tutoriel Asterisk/Openais

---

Evènements distribués avec Asterisk et le module res\_ais

Florent Fossard

06/09/2009



Ce document mis à disposition selon les termes de la [licence Creative Commons Paternité-Partage des Conditions Initiales à l'Identique 2.0 France](https://creativecommons.org/licenses/by-sa/2.0/fr/)

## Table des matières

Introduction.....	3
Installation.....	3
o Dépendances .....	3
o Corosync et Openais.....	3
o Asterisk .....	5
o Vérification .....	6
Configuration des échanges d'évènements .....	6
Les autres serveurs.....	8
Test de fonctionnement .....	8
Limitations de cette solution.....	9

## Introduction

Grâce au module `res_ais`, disponible avec la branche 1.6.1 d'Asterisk, il est possible d'échanger entre plusieurs serveurs Asterisk des messages sur les modifications d'état des voicemail et device state. `Res_ais` utilise Openais, une implémentation libre du SAForum AIS (Service Availability Forum Application Interface Specification), qui fonctionne avec le moteur de clustering Corosync, libre lui aussi.

Ce document décrit l'installation et le paramétrage d'Openais et d'Asterisk pour permettre l'échange de messages indiquant des modifications de devstate, il est basé sur la documentation originale de Russell Bryant :

[http://svnview.digium.com/svn/asterisk/branches/1.6.1/doc/distributed\\_devstate.txt](http://svnview.digium.com/svn/asterisk/branches/1.6.1/doc/distributed_devstate.txt)

La procédure est écrite pour Debian Lenny, mais elle fonctionne aussi sur Centos 5.3 avec quelques adaptations mineures.

## Installation

Considérons une installation simple de Debian Lenny, avec au moins "système standard" sélectionné au choix des logiciels, et le réseau paramétré pour l'accès à internet. Les manipulations sont effectuées avec le compte root.

### o Dépendances

Installation des dépendances pour Asterisk:

```
apt-get install -y gcc g++ make bison libncurses5 \  
libncurses5-dev zlib1g zlib1g-dev libssl-dev
```

Installation des en-têtes du kernel:

```
apt-get install -y linux-headers-$(uname -r)  
cd /usr/src  
ln -s linux-headers-$(uname -r) linux
```

Pour la suite, Corosync a besoin du paquet Groff et Openais de pkg-config:

```
apt-get install -y groff pkg-config
```

### o Corosync et Openais

Installation de corosync:

```
wget ftp://ftp%40corosync%2Eorg:downloads@corosync.org/downloads/corosync-  
1.0.0/corosync-1.0.0.tar.gz  
tar xvzf corosync-1.0.0.tar.gz  
cd corosync-1.0.0  
./configure --disable-nss  
make install  
mv /etc/corosync/corosync.conf.example /etc/corosync/corosync.conf
```

```
sed -i "s/bindnetaddr: 192.168.1.1/bindnetaddr: 10.78.99.0/"
/etc/corosync/corosync.conf # la deuxième adresse doit être celle de votre
réseau
```

#### Installation d'Openais:

```
cd ..
wget ftp://ftp%40openais%2Eorg:downloads@openais.org/downloads/openais-
1.0.0/openais-1.0.0.tar.gz
tar xvzf openais-1.0.0.tar.gz
cd openais-1.0.0
./configure
make PREFIX=/usr
make install PREFIX=/usr
mkdir /etc/ais
mv ./conf/openais.conf.example /etc/ais/openais.conf
sed -i "s/bindnetaddr: 192.168.1.1/bindnetaddr: 10.78.99.0/"
/etc/ais/openais.conf # la deuxième adresse doit être celle de votre réseau
```

#### Création du groupe et de l'utilisateur approprié pour lancer ais:

```
groupadd ais
adduser --system --no-create-home --ingroup ais ais
```

#### Test rapide d'Openais:

```
aisexec -f
(quitter avec ctrl+c)
```

#### Vous devriez avoir un résultat proche de ceci :

```
corosync [MAIN ] Corosync Cluster Engine ('trunk'): started and ready to provide service.
corosync [MAIN ] Successfully configured openais services to load
corosync [MAIN ] Successfully read main configuration file '/etc/corosync/corosync.conf'.
corosync [TOTEM] Initializing transmit/receive security: libtomcrypt SOBER128/SHA1HMAC
corosync [TOTEM] The network interface [10.78.99.251] is now up.
corosync [SERV ] Service initialized 'openais cluster membership service B.01.01'
corosync [SERV ] Service initialized 'openais event service B.01.01'
corosync [SERV ] Service initialized 'openais checkpoint service B.01.01'
corosync [SERV ] Service initialized 'openais availability management framework B.01.01'
corosync [SERV ] Service initialized 'openais message service B.03.01'
corosync [SERV ] Service initialized 'openais distributed locking service B.03.01'
corosync [SERV ] Service initialized 'openais timer service A.01.01'
corosync [SERV ] Service initialized 'corosync extended virtual synchrony service'
corosync [SERV ] Service initialized 'corosync configuration service'
corosync [SERV ] Service initialized 'corosync cluster closed process group service v1.01'
corosync [SERV ] Service initialized 'corosync cluster config database access v1.01'
corosync [SERV ] Service initialized 'corosync profile loading service'
corosync [MAIN ] Compatibility mode set to whitetank. Using V1 and V2 of the synchronization
engine.
corosync [CLM ] CLM CONFIGURATION CHANGE
corosync [CLM ] New Configuration:
corosync [CLM ] Members Left:
corosync [CLM ] Members Joined:
corosync [CLM ] CLM CONFIGURATION CHANGE
corosync [CLM ] New Configuration:
corosync [CLM ]          r(0) ip(10.78.99.251)
corosync [CLM ] Members Left:
corosync [CLM ] Members Joined:
corosync [CLM ]          r(0) ip(10.78.99.251)
corosync [TOTEM] A processor joined or left the membership and a new membership was formed.
corosync [MAIN ] Completed service synchronization, ready to provide service.
```

#### Création d'un script de démarrage pour openais/corosync :

```
vim /etc/init.d/aisexec
```

```
#!/bin/sh
### BEGIN INIT INFO
```

```

# Provides:          openais
# Required-Start:    $remote_fs
# Required-Stop:     $remote_fs
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# Short-Description: openais initscript
# Description: Script de demarrage openais/corosync a placer dans
/etc/init.d/

PATH=/sbin:/usr/sbin:/bin:/usr/bin
DESC="Service openais/corosync"
NAME=aisexec
DAEMON=/usr/sbin/$NAME
DAEMON_ARGS=""
SCRIPTNAME=/etc/init.d/$NAME
PROCESSNAME=corosync

# Exit if the package is not installed
[ -x "$DAEMON" ] || exit 0

case "$1" in
  start)
    echo "Starting $DESC" "$NAME"
    $DAEMON
    ;;
  stop)
    echo "Stopping $DESC" "$NAME"
    pkill $PROCESSNAME
    ;;
  *) echo "Usage: $SCRIPTNAME {start|stop}" >&2
    ;;
esac

```

Rendre le script exécutable:

```
chmod +x /etc/init.d/aisexec
```

## o Asterisk

Installation d'Asterisk à partir des sources :

```

cd /usr/src
wget
http://downloads.asterisk.org/pub/telephony/asterisk/releases/asterisk-
1.6.1.6.tar.gz
tar xvzf asterisk-1.6.1.6.tar.gz
./configure
make menuselect # dans « Resource Modules », res_ais est coché.
make
make install
make samples
make config

```

Attention, un bug empêche Asterisk d'envoyer la bonne valeur aux autres serveurs après modification d'un état : <https://issues.asterisk.org/view.php?id=15624>

Ce bug est corrigé dans la version 1.6.1.6, n'utilisez pas de version antérieure.

## o Vérification

Vérifier que le processus corosync est présent après démarrage d'openais:

```
/etc/init.d/aisexec start
ps aux | grep corosync
root      24339  0.2  2.8 14872 14648 ?          SLs1 13:36   0:00 corosync
```

Vérifier qu'Asterisk a bien pris en compte res\_ais:

```
/etc/init.d/asterisk start
asterisk -vvvvvvvr
vm-test3*CLI> ais clm show members
```

La commande doit renvoyer quelque chose ressemblant à ça:

```
vm-test1*CLI>
=====
=== Cluster Members =====
=====
===
=== -----
=== Node Name: 10.78.99.251
=== ==> ID: 0xfb634e0a
=== ==> Address: 10.78.99.251
=== ==> Member: Yes
=== -----
===
=====
```

Si la commande "ais" n'existe pas dans la CLI ou que la commande "ais clm show members" retourne une liste vide, vous avez probablement raté une étape.

Vérifier qu'Asterisk a chargé le module res\_ais.so :

```
vm-test1*CLI> module show like ais
Module                Description                Use
Count
res_ais.so            SAForum AIS                0
1 modules loaded
```

## Configuration des échanges d'évènements

Nous allons maintenant configurer Asterisk pour qu'il échange des informations avec les autres serveurs.

Configuration d'un "canal" d'échange de device state dans Asterisk:

```
cat > /etc/asterisk/ais.conf << EOF
[device_state]
type=event_channel
publish_event=device_state
subscribe_event=device_state
EOF
```

Création d'un dialplan simple pour tester le fonctionnement:

**Attention** : si vous avez déjà un dialplan configuré, la commande suivante va le remplacer, modifiez-le plutôt avec un éditeur de texte !

```
cat > /etc/asterisk/extensions.conf << EOF
[general]
autofallthrough=yes

[devstate_test]
exten => 1234, hint, Custom:mystate
exten => set_inuse,1,Set(DEVICE_STATE(Custom:mystate)=INUSE)
exten => set_not_inuse,1,Set(DEVICE_STATE(Custom:mystate)=NOT_INUSE)
exten => check,1,NoOp(Custom:mystate is \${DEVICE_STATE(Custom:mystate)})
EOF
```

Un reload de la configuration d'Asterisk suffit pour prendre en compte les modifications du fichier extensions.conf, mais un « restart » est requis pour les modifications du fichier

```
/etc/asterisk/ais.conf :
/etc/init.d/asterisk restart
```

On pourrait à la place d'un restart complet d'Asterisk décharger puis recharger le module res\_ais.so, mais Asterisk ne semble pas le digérer :

```
vm-test1*CLI> module unload res_ais.so
Disconnected from Asterisk server
Executing last minute cleanups
[root@localhost asterisk-1.6.1.6]# /usr/sbin/safe_asterisk: line 146: 27858
Erreur de segmentation (core dumped) ...
Asterisk ended with exit status 139
Asterisk exited on signal 11.
```

Nous devons maintenant voir le canal Ais configuré :

```
asterisk -vvvvvvv
vm-test1*CLI> ais evt show event channels
vm-test1*CLI>
=====
=== Event Channels =====
=====
===
=== -----
=== Event Channel Name: device_state
=== ==> Publishing Event Type: device_state
=== ==> Subscribing to Event Type: device_state
=== -----
===
=====
```

Et notre devstate « mystate » qui nous servira pour les tests :

```
vm-test1*CLI> core show hints
vm-test1*CLI>
-- Registered Asterisk Dial Plan Hints --
          1234@devstate_test          : Custom:mystate
State:Idle          Watchers 0
-----
- 1 hints registered
```

## Les autres serveurs

Openais communique en multicast avec les autres nœuds, vous n'avez donc besoin d'indiquer nul part les adresses des autres serveurs.

Configurez vos autres serveurs de la même manière que le premier.

## Test de fonctionnement

Vous avez configuré au moins 2 serveurs Asterisk pour s'échanger des évènements, testons maintenant le fonctionnement.

Vérifier que les serveurs se voient :

```
vm-test1*CLI> ais clm show members
vm-test1*CLI>
=====
=== Cluster Members =====
=====
===
=== -----
=== Node Name: 10.78.99.251
=== ==> ID: 0xfc634e0a
=== ==> Address: 10.78.99.251
=== ==> Member: Yes
=== -----
===
=== -----
=== Node Name: 10.78.99.252
=== ==> ID: 0xfb634e0a
=== ==> Address: 10.78.99.252
=== ==> Member: Yes
=== -----
===
=====
```

Modifier la valeur de notre device state sur un des serveurs :

```
vm-test1*CLI> console dial set_inuse@devstate_test
-- Executing [set_inuse@devstate_test:1] Set("Console/dsp",
"DEVICE_STATE(Custom:mystate)=INUSE") in new stack
<< Hangup on console >>
DEBUG[2947]: ais/evt.c:184 ast_event_cb: Got an event to forward
DEBUG[2947]: ais/evt.c:239 ast_event_cb: Returning here (event_free)
DEBUG[2947]: ais/evt.c:184 ast_event_cb: Got an event to forward
DEBUG[2947]: ais/evt.c:239 ast_event_cb: Returning here (event_free)
vm-test1*CLI>
```

Dans le même temps sur l'autre serveur :

```
vm-test2*CLI>
DEBUG[2378]: ais/evt.c:184 ast_event_cb : Got an event to forward
DEBUG[2378]: ais/evt.c:188 ast_event_cb: Returning here
DEBUG[2378]: ais/evt.c:184 ast_event_cb: Got an event to forward
DEBUG[2378]: ais/evt.c:188 ast_event_cb: Returning here
vm-test2*CLI>
```

Vérifier que la valeur a bien changé sur les deux serveurs :

```
vm-test1*CLI> core show hints
vm-test1*CLI>
  -= Registered Asterisk Dial Plan Hints -=
                1234@devstate_test      : Custom:mystate
State:InUse      Watchers  0
-----
- 1 hints registered

vm-test2*CLI> core show hints
vm-test2*CLI>
  -= Registered Asterisk Dial Plan Hints -=
                1234@devstate_test      : Custom:mystate
State:InUse      Watchers  0
-----
- 1 hints registered
```

Le statut est passé à « InUse » sur les deux serveurs, le but est atteint.

Vous pouvez le remettre à « idle » avec l'autre extension du dialplan :

```
vm-test1*CLI> console dial set_not_inuse@devstate_test
```

## Limitations de cette solution

Je vois deux inconvénients à cette solution basée sur Openais :

- L'utilisation du multicast pour la communication entre les serveurs, cela restreint la portée du système au réseau local.
- L'absence d'authentification des « participants », n'importe qui peut modifier n'importe quel device state sur tous les serveurs, il faut donc gérer la sécurité « à côté », avec par exemple des règles de pare-feu.

Une solution encore en test, basée sur XMPP, évite ces deux points négatifs :

<https://issues.asterisk.org/view.php?id=15757>