EarlyWarning

Tests de Validation

Thomas Kowalski

Juillet 2018

Table des matières

1 Tests de l'interface d'édition de listes de contacts

1.1 Ajout d'un contact

Scénario

— Démarrer le serveur EarlyWarning ;
— Lancer un navigateur Internet et se rendre à l'adresse correspondante (par défaut : `IP:6001` *le port peut-être modifié dans le fichier de configuration* `earlywarning.xml`*, section* `contacts` → `port` ;
— Ajouter un nouveau contact en remplissant les champs *Nom* et *Téléphone* ;
— Valider en cliquant sur *Ajouter*.

Comportement attendu

Le nouveau contact est ajouté dans la liste de contacts correspondante.
Pour le vérifier, on peut ouvrir le fichier de la liste de contacts (on peut trouver la correspondance entre le nom de la liste et le fichier dans `earlywarning.xml`, section `contacts` → `lists` → `<le nom de la liste choisie>` et vérifier que le nouveau contact est bien présent.

1.2 Ajout d'un contact à la liste d'appel

Scénario

— Démarrer le serveur EarlyWarning ;
— Lancer un navigateur Internet et se rendre à l'adresse correspondante (par défaut : `IP:6001` *le port peut-être modifié dans le fichier de configuration* `earlywarning.xml`*, section* `contacts` → `port` ;
— Ajouter un contact à la liste d'appel en le faisant glisser de gauche à droite.

Comportement attendu

Le contact sélectionné est ajouté à la liste d'appel correspondante à la position choisie.
Pour le vérifier, on peut ouvrir le fichier de la liste de contacts correspondant.

Déroulement et résultat

Succès On ajoute un contact (`Philippe` / `0692877305`) et on actualise la page, il est bien présent dans la liste. On redémarre l'application, le contact est toujours présent.

1.3 Réordonnement de la liste d'appel

Scénario

— Démarrer le serveur EarlyWarning ;
— Lancer un navigateur Internet et se rendre à l'adresse correspondante (par défaut : `IP:6001` *le port peut-être modifié dans le fichier de configuration* `earlywarning.xml`*, section* `contacts` → `port` ;
— Réordonner la liste d'appel.

Comportement attendu

La position du contact sélectionné a été modifiée dans la liste d'appel choisie..
Pour le vérifier, on peut ouvrir le fichier de la liste de contacts correspondant.

Tests supplémentaires

On peut également vérifier qu'un contact *prioritaire* est toujours situé en haut de la liste d'appel. Pour le vérifier, on peut réordonner la liste (qui devrait toujours laisser le contact prioritaire tout en haut) et vérifier que le fichier a toujours ce contact en première position.

Déroulement et résultat

Succès On fait glisser un contact vers la liste d'appel puis on actualise la page, il est toujours présent. On redémarre l'application, le contact est toujours dans la liste d'appel.

2 Tests de la passerelle Asterisk

2.1 Test d'un appel simple

La classe `ValidationTestAsterisk` *permet de faciliter la mise en place de ce test.*
Pour mettre en place ce test, on s'intéresse à cette méthode :

```
@Test
public void validationTest() throws Exception {
    List<String> callList = new ArrayList<>();
    callList.add("<numero 1>");
    callList.add("<numero 2>");
    String code = "1256";
    String message = "demo-thanks";

    Tester.run(callList, code, message);
}
```

Pour lancer le test, ajouter autant de numéros que souhaité à `callList` grâce à `callList.add("numero");`.
Les variables `code` et `message` permettent respectivement de personnaliser le code de confirmation à entrer pour arrêter les appels et le son (son nom sur le serveur Asterisk) à jouer (il correspond au message de détail joué à l'appelé).
Il suffit ensuite de lancer le test unitaire correspondant, en utilisant la commande

```
mvn test -Dtest=gateway.ValidationTestAsterisk
```

Comportement attendu
Les numéros de la liste sont appelés en boucle jusqu'à ce qu'un appelé confirme la réception du message en entrant le code spécifié dans `code`.
A chaque fois, il est accueilli avec un message de bienvenue, doit écouter le message d'avertissement, entrer le code et a droit à un certain nombre d'erreurs de code (tel que précisé dans la configuration).

2.2 Test d'un appel avec *trigger*

Pour vérifier le fonctionnement de la réception des *triggers* et des appels correspondant, il suffit de passer un paramètre à l'application :

```
java -jar AlarmePrecoce.jar --testcalls
```

Comportement attendu
Deux *triggers* sont ajoutés à la pile. Ce sont respectivement ceux écrits (et émis) par `TriggerV2Sender3` et `TriggerV2Sender2`.
Les appels sont passés aux listes correspondantes (ou, si elles n'existent pas, à la liste par défaut), tant que personne ne confirme la réception du message.
Les appels sont passés de manière synchrone : le premier *trigger* est traité jusqu'à confirmation de la réception, puis le second. Les appels ne sont pas passés en parallèle pour les deux *triggers*.

Déroulement et résultat
Succès On ajoute deux triggers depuis la page de test sur WebObs, les appels sont bien émis l'un après l'autre jusqu'à validation par l'utilisateur.

2.3 Non-démarrage en cas d'identifiants Asterisk invalides

Motivation
L'initialisation de l'application est écrite de façon à vérifier les identifiants de l'Asterisk Manager Interface au démarrage. Si ceux-ci sont incorrects, l'application ne démarre pas.

Scénario

- Modifier le fichier de configuration `earlywarning.xml` et modifier les champs `ami_user` et `ami_password` de façon à les rendre incorrects ;
- Démarrer l'application EarlyWarning.

Comportement attendu

L'application ne démarre pas et affiche un message d'erreur.

```
FATAL [main] 2018-07-16 10:59:05,605 - fr.ipgp.earlywarning.utilities.ConfigurationValidator
    ValidationException on parameter 'gateway.asterisk.settings':
    Asterisk Manager Interface credentials are incorrect.
```

Déroulement

On modifie le fichier de configuration afin d'entrer volontairement des identifiants AMI faux. On redémarre l'application, celle-ci refuse de s'initialiser avec le message attendu.

3 Tests de la passerelle Charon

Motivation

Le système de *failover* utilise une passerelle Charon, alarme anti-intrusion de l'observatoire. On souhaite pouvoir tester le bon fonctionnement de celle-ci.

Scénario

Pour tester le bon fonctionnement, il faut modifier le fichier de configuration, entrée `gateway` → `active` et de remplacer `asterisk` par `charon`.

On peut ensuite utiliser l'option `-testcalls` afin d'émettre deux appels de test.

Comportement attendu

L'application démarre et appelle deux fois le numéro de téléphone d'astreinte, en donnant un message d'avertissement générique (*Veuillez vous rendre sur WebObs...*), jusqu'à confirmation de la part de la personne d'astreinte.

Remarque après ce test, il faut redémarrer l'application EarlyWarning après modification du fichier de configuration afin qu'elle utilise à nouveau la passerelle Asterisk.

Déroulement

Après émission d'un *trigger* par l'interface WebObs, l'application Alarme Précoce communique comme prévu avec le module Charon afin de lui faire émettre deux appels à la suite. Une fois les deux appels émis, l'application continue à fonctionner normalement.

4 Tests du système de *failover* utilisant la passerelle Charon

Motivation
Il est possible, pour des raisons matérielles, que la passerelle utilisée par le serveur Asterisk soit inaccessible, ou que celle-ci ne puissent pas émettre les appels comme prévu.
Le système de *failover* vise à minimiser le risque lié à ces problèmes en prévoyant une deuxième passerelle téléphonique : celle de l'alarme anti-intrusion de l'observatoire.
Celle-ci offre moins de possibilités que la passerelle Asterisk, mais est toujours accessible.

Remarque La fonctionnalité de validation de la configuration est faite pour empêcher le démarrage de l'application EarlyWarning en cas de serveur Asterisk inaccessible. Afin d'effectuer ces tests, il faut donc changer le comportement du validateur en passant le paramètre `-novalidate`.

Scénario A : pas de liaison Asterisk $\longleftrightarrow$ passerelle AudioGuides

- Débrancher le câble Ethernet de la passerelle AudioGuides ;
- Lancer un des tests de la passerelle Asterisk, le plus simple étant d'utiliser l'option `-testcalls`.

Scénario B : pas de liaison passerelle AudioGuides $\longleftrightarrow$ réseau téléphonique

- Débrancher le câble téléphonique de la passerelle AudioGuides ;
- Lancer un des tests de la passerelle Asterisk, le plus simple étant d'utiliser l'option `-testcalls`.

Scénario C : serveur Asterisk indisponible pour des raisons logicielles

- Arrêter Asterisk en utilisant `pkill -9 asterisk`
- Lancer un des tests de la passerelle Asterisk, le plus simple étant d'utiliser l'option `-testcalls`

Comportement attendu
L'application EarlyWarning tente d'émettre l'appel en utilisant la passerelle Asterisk, qui répond avec une erreur. Au bout d'un certain nombre d'essais (tel que précisé dans le code), elle abandonne et remplace sa passerelle Asterisk par une instance de la passerelle Charon.
Elle recommence alors son cycle d'appels avec la nouvelle plate-forme, jusqu'à confirmation d'un opérateur.
Lorsque le problème concernant la passerelle Asterisk (matériel ou logiciel) est résolu, l'application utiliser à nouveau la passerelle par défaut.

Déroulement
On effectue les trois tests proposés.

(A) Succès Après émission d'appels via l'interface WebObs, l'application tente d'émettre les appels en utilisant Asterisk, mais reçoit des erreurs de sa part, et passe donc les appels grâce à la passerelle Charon. Lorsque la connexion est rétablie, l'application utilise à nouveau la passerelle Asterisk.

(B) Succès Après émission d'appels via l'interface WebObs, l'application tente d'émettre les appels en utilisant Asterisk, mais reçoit des erreurs de sa part, et bascule vers la passerelle Charon. Lorsque la connexion est rétablie, l'application utilise à nouveau la passerelle Asterisk.

(C) Succès Après émission d'appels via l'interface WebObs, l'application tente d'émettre les appels en utilisant Asterisk, mais reçoit des erreurs de sa part, et bascule vers la passerelle Charon. Lorsque le service Asterisk est redémarré, l'application utilise à nouveau la passerelle Asterisk.

5 Tests du système haute-disponibilité

5.1 Introduction

Avant d'effectuer ces tests, le système haute-disponibilité doit être configuré, avec une instance principale et une instance secondaire.

5.2 Test de l'unicité de l'appel dans le cas normal

Scénario

— Démarrer EarlyWarning sur les deux serveurs ;
— Émettre un *trigger* ;
— Suivre la procédure d'appel standard.

Comportement attendu

Les deux instances reçoivent le *trigger* (vérifier dans les journaux), mais l'instance secondaire envoie une requête à l'instance principale qui lui répond ; l'instance secondaire n'émet donc aucun appel (ou e-mail, SMS, *etc.*).

Déroulement

Succès L'instance secondaire a vérifié l'existence de la première, a reçu une réponse positive et n'a pas émis d'appel.

5.3 Test de l'utilisation de l'instance secondaire si la première est indisponible

Scénario

— Démarrer EarlyWarning sur les deux serveurs ;
— Arrêter EarlyWarning sur le serveur principal ;
— Émettre un trigger ;
— Suivre la procédure d'appel standard.

Comportement attendu

Lors de la réception du *trigger* par l'instance secondaire, elle vérifie la présence de l'instance principale, qui ne répond pas (le serveur refuse la connexion). L'instance secondaire prend donc la responsabilité de passer l'appel et d'émettre les autres alertes (e-mails, SMS, *etc.*).

Déroulement

Succès L'instance secondaire a bien appelé les numéros de sa liste d'appel.

5.4 Test du basculement vers l'instance principale en cas de redémarrage

Ce test peut être effectué directement après le précédent, on est alors dans un cas où l'instance principale a été indisponible et où on la redémarre.

Scénario *Après le test précédent...*

— Redémarrer l'instance principale (`systemctl start earlywarning.service`) ;
— Émettre un *trigger* ;
— Suivre la procédure d'appel standard.

Comportement attendu

Les deux instances reçoivent le *trigger*, l'instance secondaire vérifie à nouveau la présence de l'instance principale qui, cette fois, lui répond. L'instance secondaire arrête le traitement du *trigger*, il n'est traité que par l'instance principale qui effectue le traitement attendu.

Déroulement

Succès L'instance secondaire a bien vérifié la présence de l'instance principale, qui lui a répondu par l'affirmative. Elle a arrêté le traitement du *trigger* ; l'instance principale a quant à elle suivi la procédure prévue.

6 Tests des différents scénarios d'appel possibles

6.1 Introduction

Avant d'effectuer ces tests, il convient de régler les listes d'appel correspondantes aux appels émis par les testeurs de *triggers* par l'interface Web et d'y ajouter au moins deux numéros, afin de pouvoir vérifier le fonctionnement "en boucle".

6.2 Déroulement normal

Scénario

— Démarrer EarlyWarning avec l'option `-testcalls` ;
— Attendre que l'appel soit émis ;
— Répondre au téléphone dès que possible ;
— Écouter le message de bienvenue et le valider avec le code donné ;
— Écouter le message d'avertissement et le confirmer avec le code de confirmation ;
— Raccrocher.

Comportement attendu

L'appel est émis, le message de bienvenue est joué, la confirmation du message de bienvenue fonctionne, le message d'avertissement est joué, la confirmation du message d'avertissement par l'entrée du code fonctionne. Une fois le message validé, aucun autre appel n'est émis pour ce *trigger*.

Déroulement

Succès Le trigger est émis par WebObs, l'appel se déroule comme prévu et l'application continue à fonctionner.

6.3 Pas de réponse du premier appelé, appels en boucle

Scénario

— Démarrer EarlyWarning avec l'option `-testcalls` ;
— Attendre que l'appel soit émis ;
— Ne pas répondre au téléphone ;
— Attendre que le deuxième appel soit émis ;
— Ne pas répondre au téléphone ;
— Attendre que le troisième appel soit émis ;
— Décrocher et valider le message.

Comportement attendu

L'appel est émis et sonne pendant un certain temps (tel que réglé dans la configuration). Au bout de ce temps, un appel est émis vers le deuxième numéro de téléphone de la liste.
L'appel est émis vers le deuxième numéro de téléphone, puis est raccroché.
(Dans le cas où la liste ne comporte que deux numéros de téléphone,) un troisième appel est émis vers le premier numéro de téléphone. En décrochant, le scénario du déroulement normal fonctionne.

Déroulement

Succès Le trigger est émis par WebObs, les deux premiers numéros sont appelés puis abandonnés, lors de la validation du troisième, l'application arrête d'appeler et continue à fonctionner normalement.

6.4 Appel raccroché avant l'entrée du code

Scénario

— Démarrer EarlyWarning avec l'option `-testcalls` ;
— Attendre que l'appel soit émis ;
— Décrocher dès que possible ;
— Confirmer le message de bienvenue ;
— Ecouter ou raccrocher pendant le message d'avertissement, dans tous les cas raccrocher avant l'entrée complète du code.

Comportement attendu
Lors du raccrochage, les appels continuent d'être émis dans l'ordre de la liste d'appel, comme dans le scénario *Pas de réponse du premier appelé, appels en boucle.*

Déroulement
Succès Après l'émission d'un trigger via WebObs, on laisse les téléphones sonner. Au bout de la liste, c'est à nouveau le premier numéro qui est appelé.

7 Tests du service

7.1 Démarrage automatique

Scénario

— Redémarrer le serveur.

Comportement attendu
Lors du redémarrage, Asterisk et l'application Alarme Précoce sont automatiquement démarrés.
Pour le vérifier :

```
> systemctl status earlywarning.service

  earlywarning.service - Alarme Precoce de l'OVPF
  Loaded: loaded (/etc/systemd/system/earlywarning.service; enabled; vendor preset: enabled)
  Active: active (running) since Wed 2018-07-25 11:08:01 UTC; 5s ago
Main PID: 6594 (bash)
   Tasks: 16 (limit: 4915)
  CGroup: /system.slice/earlywarning.service
       |- 6594 /bin/bash /root/Alarme/EarlyWarning.sh
       |- 6595 java -jar EarlyWarning.jar
```

Déroulement
Après redémarrage du serveur (`reboot`), le service est correctement démarré.