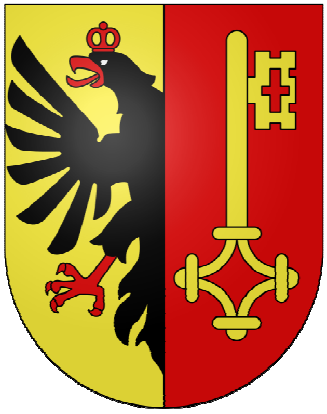
Réception/Transmission des Alarmes sur un réseau IP



Travail de Bachelor réalisé en vue de l'obtention du Bachelor HES

par :

Christophe JORDAN

Conseiller au travail de Bachelor :

Gérard INEICHEN, Enseignant HEG

Genève, le 22 Octobre 2010

Haute École de Gestion de Genève (HEG-GE)

Filière Informatique de Gestion

Déclaration

Ce travail de Bachelor est réalisé dans le cadre de l'examen final de la Haute école de gestion de Genève, en vue de l'obtention d'un Bachelor d'Informatique de Gestion. L'étudiant accepte, le cas échéant, la clause de confidentialité. L'utilisation des conclusions et recommandations formulées dans le travail de Bachelor, sans préjuger de leur valeur, n'engage ni la responsabilité de l'auteur, ni celle du conseiller au travail de Bachelor, du juré et de la HEG.

« J'atteste avoir réalisé seul le présent travail, sans avoir utilisé des sources autres que celles citées dans la bibliographie. »

Fait à Genève le 22 Octobre 2010

Christophe JORDAN

Remerciements

Mr Gérard INEICHEN pour son soutien et ces conseils précieux ainsi que pour nous avoir fourni des conditions de travail optimales.

Mr Pascal EMONET pour la mise à disposition du matériel nécessaire au bon déroulement du projet.

Mr David BILLARD pour son support pour les questions de base de données

Mr Jonathan MALFOY pour son aide et sa collaboration tout au long du travail.

Et tout le reste de mon entourage qui a participé de près ou de loin à la réalisation de ce mémoire.

Résumé

L'État de Genève, plus précisément la division Réseaux/Télécoms dispose d'un système de transmission d'alarmes vocales réalisé voici une dizaine d'années. Ce système est fermé, non extensible, et dépendant du hardware qui est devenu obsolète. De plus quelques nouvelles fonctionnalités sont envisagées, afin d'étendre ce système à de multiples applications avec gestion partiellement déléguée aux utilisateurs.

C'est pourquoi il convient d'étudier et réaliser un système capable de reprendre différentes alarmes (tant des contacts secs tels que des thermostats ou des centrales de détection feu, que des éléments informatiques (réponse erronée d'une application, état snmp incorrect, traps snmp) ou encore des boîtiers IP de senseurs divers), de les traiter en fonctions de critères d'urgence et d'alarmer les bonnes personnes, en tenant compte de paramètres tels que, par exemple, les vacances des collaborateurs.

Par alarmer, il faut comprendre appeler les personnes désignées, leur jouer un texte préenregistré indiquant le type d'alarme, et leur proposer d'entrer en conférence téléphonique pour décider des mesures à prendre, ou quitter le système en cas d'impossibilité de prise en charge.

La réalisation se fera en favorisant les logiciels libres, tant du côté web (php, mysql) que du côté téléphonie (asterisk) et devra s'intégrer autant que faire se peut dans le contexte des infrastructures existantes à Réseaux/Télécoms. Le travail se déclinera en deux travaux de bachelor distincts, celui de Jonathan Malfoy et ce travail.

Table des matières

Introduction **1**

1. Les alarmes **2**

1.1 Les sources d'alerte **2**

1.1.1 Le protocole SNMP *2*

1.1.2 Contact sec *2*

1.1.3 Disponibilité des sondes *2*

1.1.4 Programme informatique *2*

1.2 Les différents états d'une alarme **2**

1.2.1 Alarme *2*

1.2.2 Normal *2*

1.2.3 Dérangement *3*

1.3 Les critères d'urgence **3**

1.3.1 Type d'alarme *3*

1.3.2 Périodes de transmission *3*

2. La norme DC-09 : 2007 ..... **4**

2.1 Type de message selon la norme DC-09 ..... **4**

2.2 Champ x.data **5**

2.2.1 Définition *5*

2.2.2 Les balises *5*

3. Critères de test et exécution de scripts **6**

3.1 Scripts **6**

3.2 Réponse **6**

4. Daemons **7**

4.1 Daemon Execute **8**

4.1.1 Vérifier le type de jour *8*

4.1.2 Récupération des tests *8*

4.1.3 Mise en processus *9*

4.1.4 Réactivation de tests *9*

4.1.5 Exécution des tests *9*

4.1.6 Analyse du code de retour *10*

4.1.7 Mise à jour des tests *10*

4.2 Daemon Critère **11**

4.2.1 Récupérer les alertes *11*

4.2.2 Mise en processus *11*

4.2.3 Transmission des alertes *11*

4.2.4 Mise à jour de l'alerte *11*

5. Journalisation complète des actions 13
5.1 Protocole SYSLOG 13
5.1.1 L'application RSYSLOG 13
5.1.2 Formatage des informations pour l'application 14
6. Réception des alarmes 15
6.1 Réception d'alarmes selon le standard DC-09 15
6.2 Réception d'alarmes « autres » 15
7. Base de données : Alarmes 16
7.1 Tables 17
7.1.1 Table TCriteresTest 17
7.1.2 Table TAlertes 18
7.1.3 Table TAlertesATransmettre 18
7.1.4 Table TCalendrier 18
7.1.5 Table TPeriodes 19
7.1.6 Table histo 19
7.1.7 Table utilisateur 19
7.2 Triggers 20
7.2.1 Trigger verifierCompteur 20
7.2.2 Trigger ajouterTemps 20
7.3 Stockage des alertes dans la BDD 21
7.3.1 Les alertes détectées 21
7.3.2 Les alertes transmises 21
8. Interface Web 22
8.1 Gestion des critères/tests 23
8.2 Gestion du calendrier 25
8.3 Gestion des utilisateurs 26
8.4 Identification LDAP / GINA 27
8.4.1 GINA 27
8.4.2 LDAP 28
8.5 Historique 28
8.6 Droits d'accès 29
8.6.1 Les différents rôles 29
8.6.2 La gestion par groupe 29
Conclusion 30
Bibliographie 31
Annexe 1 Manuel d'installation 32
Annexe 2 Manuel d'utilisation 38

Liste des Tableaux

Tableau 1 Paramètres du message DC-09 4
Tableau 1 Définition des balises 5

Liste des Figures

Figure 1 Exemple de périodes 3
Figure 2 Schéma global 7
Figure 3 Mise en processus 9
Figure 4 Structogramme Daemon Execute 10
Figure 5 Structogramme Daemon Critère 12
Figure 6 Réception des alarmes 15
Figure 7 Schéma base de données 16
Figure 8 Schéma interface web 22
Figure 9 Gestion d'un critère partie 1 23
Figure 10 Gestion d'un critère partie 2 24
Figure 11 Liste des fériés 25
Figure 12 Gestion d'un férié 25
Figure 13 Liste utilisateurs 26
Figure 14 Gestion d'un utilisateur 26
Figure 15 Page d'authentification GINA 27
Figure 16 Fenêtre d'authentification LDAP 28
Figure 17 Historique 28

Introduction

Ce travail traitera la réception d'alarmes sur un réseau IP consistant à récupérer des signaux d'alarmes décrits dans le résumé (contacts sec et autres, en DC-09 :2007 si possible).

La transmission d'alarmes consistera à envoyer des signaux reçus des différentes sources si possible selon la norme DC-09 :2007 pour qu'ils soient traités (cf. Travail Bachelor de Jonathan MALFOY « Traitement d'alarmes et mise en conférence téléphonique »). Pour transmettre les signaux sous un seul et unique format, il faut les convertir selon la norme DC-09 (pour toutes les sondes qui ne le font pas directement).

Les technologies utilisées seront le C++ pour le daemon servant à réceptionner les alarmes et les stocker dans une base de données MySQL. PHP & MySQL seront utilisés pour la partie interface web et gestion du DC-09.

Ce travail est donc séparé en plusieurs parties

- Recherche et documentation des différentes technologies utilisées
- Réception des alarmes & exécution des scripts de test
- Transmission des alarmes en fonction des périodes de transmission (si possible selon la norme DC-09)
- Création d'une interface web pour la gestion des outils, avec authentification.

1. Les alarmes

1.1 Les sources d'alerte

Les sources d'alerte peuvent être nombreuses et variées mais toutes doivent être capables d'envoyer des messages sur le réseau ou d'être testée via le réseau ou serveur.

1.1.1 Le protocole SNMP

Le SNMP, en français « protocole simple de gestion de réseau », est un protocole de communication qui permet aux administrateurs réseau de gérer les équipements du réseau, de superviser et de diagnostiquer des problèmes réseaux et matériels à distance. Lorsqu'un certain événement se produit sur un équipement, un paquet UDP est envoyé à un serveur. C'est le concept de « traps SNMP ».

1.1.2 Contact sec

Un contact sec est une alarme qui contient seulement deux états. Ouvert et fermé. Les contacts secs doivent être reliés sur un émetteur Ethernet pour pouvoir transmettre leur état sur le réseau via le protocole SNMP.

1.1.3 Disponibilité des sondes

Pour vérifier la disponibilité des sondes, on teste leur présence sur le réseau (cf. 1.2 les différents états d'une alarme).

1.1.4 Programme informatique

Pour vérifier, par exemple, si un DNS est actif, si une page web répond ou encore si un daemon existe.

1.2 Les différents états d'une alarme

1.2.1 Alarme

Equipement opérationnel, alarme détectée

1.2.2 Normal

Equipement opérationnel, rien n'à signaler

1.2.3 Dérangement

Equipement indisponible, l'alarme sera donnée selon les critères paramétrés pour cette alarme (ex : après 3 tests sans réponse, déclenchement de l'alarme cf. 5.1)

1.3 Les critères d'urgence

Il est nécessaire de définir des critères d'urgence pour éviter les dérangements inutiles des personnes concernées car les équipements envoient des alertes en continu sur le réseau. Par exemple, si une simple ouverture de porte génère une alerte, et il serait inutile de la traiter durant les heures d'ouverture. Ces critères sont administrés depuis l'interface de gestion web (cf. 8.1 Interface web pour la gestion des critères/tests).

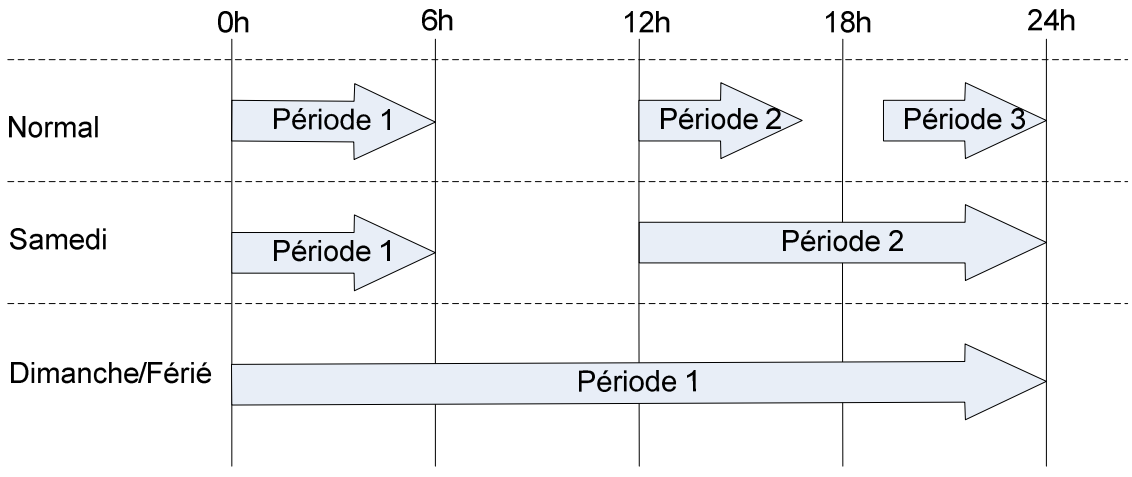
1.3.1 Type d'alarme

Certains types d'alarmes sont plus urgents que d'autres. Une alarme incendie devra être traitée immédiatement et en priorité, alors qu'un dérangement d'un climatiseur peut attendre le matin si la température ne monte pas trop.

1.3.2 Périodes de transmission

On peut déclencher une urgence selon un certain horaire. Par exemple, les portes d'un bâtiment ne doivent **pas** être ouvertes avant 6h et après 20h pendant la semaine, l'après-midi le samedi et toute la journée les dimanches et jours fériés. Il peut y avoir jusqu'à 3 périodes de transmission configurées par jour. Si lors du déclanchement de l'alerte nous sommes dans une des périodes de transmission, alors le traitement de ce dernier sera immédiat.

Figure 1
Exemple de périodes



2. La norme DC-09 : 2007

La norme DC-09 : 2007 a été établie par SIA Digital Communication Standard – Internet Protocol Event pour uniformiser les signaux d'alarmes IP. Le signal a une structure fixe (cf. 2.1). Le champ x.data est un champ libre sans contrainte de longueur et qui permet, si besoin est, de transmettre des informations supplémentaires (le campus, le bâtiment, l'étage, le détecteur actionné, etc..) grâce a des balises définies (cf. 2.2.2).

Remarque : Les équipements DC-09 n'ayant pas été fournis, la norme DC-09 n'a pas pu être implémentée. Dans l'hypothèse d'une future implémentation, voici les grandes lignes concernant la norme DC-09.

2.1 Type de message selon la norme DC-09

<LF><CRC><0LLL><"id"><seq><Rrcvr><Lpref><#acct>[<pad>|data...][x.data...]
<timestamp><CR>

Tableau 1

Paramètres du message en DC-09

Paramètres	Commentaire
LF	Caractère ASCII Line Feed (0A en hexadécimal).
CRC	Contrôle de redondance cyclique.
LLL	*Longueur du message (3 digits hexadécimaux, précédés par le caractère zéro).*
"id"	Code identifiant le format d'entrée
Seq	Numéro de séquence (0001-9999, pas incrémenté lors d'un renvoi).
Rcvr	Identifiant du récepteur, si utilisé (1-6 digits hexadécimaux, précédés d'un R).
Pref	Préfixe (1-6 digits hexadécimaux, précédés d'un L).
acct	Identifiant du transmetteur (3-16 digits hexadécimaux, précédés par un #).
Pad	Caractères de remplissage (uniquement quand le message est crypté).
data	Champ de données, libre, sans limite de longueur, syntaxe selon format d'entrée.
x.data	Paramètre d'extension, identifiable par la « balise » suivant le caractère « [». Sa valeur est le texte compris entre la balise et le crochet de fermeture «] ».
timestamp	Horodatage, heure de transmission du message, requis si le message est crypté (20 caractères, format "_HH:MM:SS,MM-DD-YYYY")
CR	Caractère ASCII Carriage Return (0D en hexadécimal)

Source : eca-vaud.ch (Complément technique à la règle de prescription, p. 15)

2.2 Champ x.data

2.2.1 Définition

Les paramètres d'extension (x.data) sont utilisés pour transmettre les différents paramètres optionnels (Bâtiment concerné, étage, heure de survenance, adresse MAC, etc..).

Ces paramètres sont codés selon la syntaxe « [Xccc...cc] » où X est la balise d'identification du paramètre (cf. 2.2.2) et ccc...cc sa valeur (texte de longueur variable compris entre la balise et le crochet de fermeture). Cette syntaxe s'applique à chaque paramètre d'extension. Les paramètres d'extension peuvent ainsi être mis à la suite l'un de l'autre, dans n'importe quel ordre, chaque paramètre étant identifié par une balise univoque. La syntaxe « < [><balise><valeur><]> » doit être respectée pour chacun de ces paramètres. La chaîne <valeur> ne doit pas contenir les caractères « [», « | » ou «] ».

2.2.2 Les balises

D'après le standard DC-09, seules les lettres de G à Z sont utilisables pour attribuer des balises et certaines de ces lettres sont réservées par ce dernier (en gras). Dans notre cas nous avons défini un certain nombre de balises dont nous allons avoir besoin (cf. Tableau 2).

Tableau 2

Définition des balises

Balise	**Affectation**	**Format**
H	Heure de survenance	_HH :MM :SS,MM-DD-YYYY
L	Etage	Variable
M	**Adresse MAC**	**12 caractères hexadécimaux**
O	Bâtiment	Variable
P	**Données de *Programmation***	***Libre (usage futur)***
R	Salle/Local	Variable
S	Site (ex : Battelle)	Variable
T	Trigger de déclenchement	Variable
V	**Données de Validation**	**Libre (usage futur)**

Source : eca-vaud.ch (Complément technique à la règle de prescription, p. 16)

3. Critères de test et exécution de scripts

Le système parcourt la table des critères de tests à effectuer (cf. 7.1.1) et exécute les tests à la fréquence inscrite dans la table. Ces tests ont plusieurs critères importants :

- Un identifiant unique
- Un nom d'alarme et du script à exécuter
- Une fréquence d'exécution
- La réponse attendue (par défaut 0)

3.1 Scripts

Tous les scripts 'Shell' doivent être placés dans le dossier « /usr/share/scriptalarme » du serveur pour que le système puisse les exécuter. Lors de l'insertion d'un nouveau test depuis l'interface de gestion des tests (cf. 8.1), l'utilisateur devra inscrire le nom du script qu'il aura préalablement transmis à l'administrateur pour que ce dernier le copie dans le dossier des scripts.

3.2 Réponse

Les codes de retour des scripts devront être normalisés pour que le système puisse analyser les réponses. Le standard appliqué pour ce travail est le suivant :

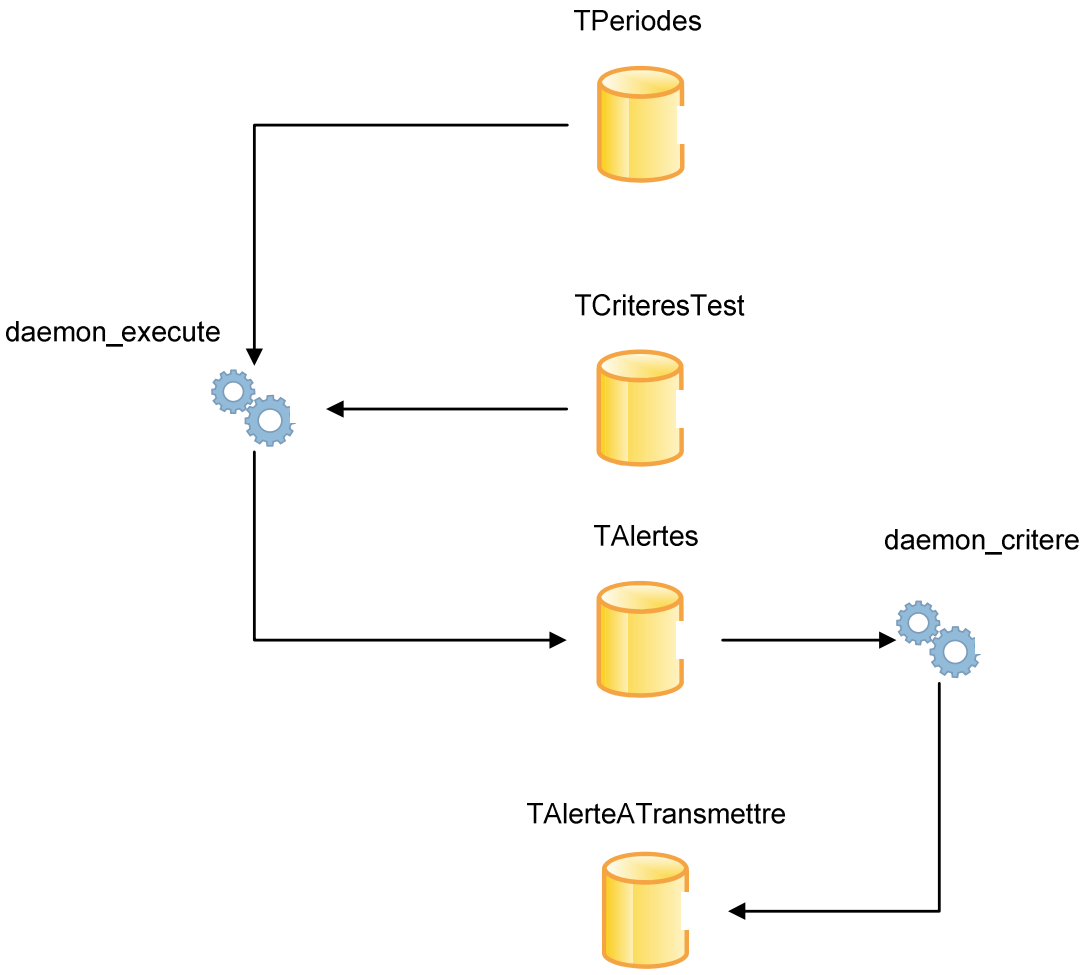
- 0 : Exécution réussie
- 1 : Erreur d'exécution
- .. : d'autres codes de retour peuvent être ajoutés.

4. Daemons

Pour ce travail il a fallu créer 2 daemons :

- Daemon Execute
- Daemon Critere (à transmettre)

Figure 2
Schéma global



4.1 Daemon Execute

Ce daemon possède plusieurs responsabilités :

- Vérifier le type de jour
- Récupérer les tests
- Mise en processus
- Réactivation de tests
- Exécution des scripts
- Analyse du code de retour des scripts
- Mise à jour des tests

4.1.1 Vérifier le type de jour

Le daemon vérifie constamment le type du jour actuel parmi les trois possibilités suivantes :

- Jours ouvrables (Lundi au Vendredi)
- Samedi
- Fériés (jours/périodes défini(e)s dans le calendrier de l'application gérés depuis l'interface web)

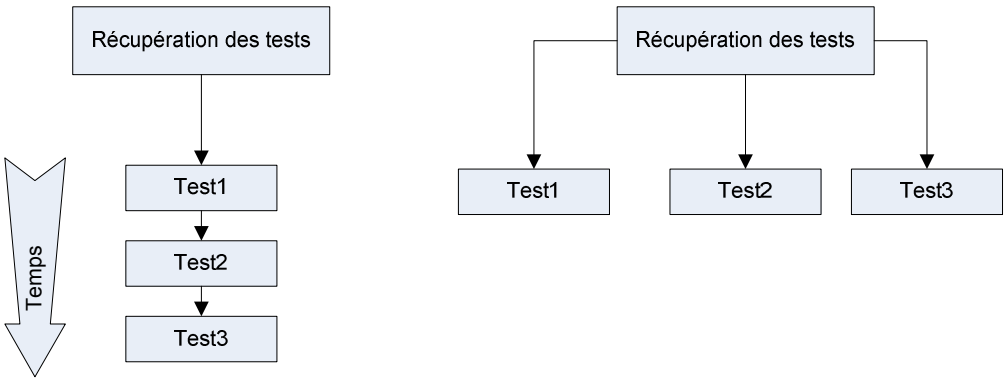
4.1.2 Récupération des tests

Le daemon récupère tous les tests présents dans la table TCriteresTest ainsi que leurs périodes de transmission stockées dans la table TPeriodes pour définir l'heure de transmission de l'alarme en cas d'alerte.

4.1.3 Mise en processus

Pour chacun des tests récupérés, un processus sera créé pour le traité. Cela nous évitera d'attendre sur un test en amont qui ne répond plus. Sur l'exemple de gauche si dessous (sans la mise en processus) si le Test2 ne répond plus il bloque la chaîne et le Test3 ne pourra pas s'exécuter. Cela pourrait être plus qu'ennuyeux si ce dernier est une sonde incendie… Avec le diagramme de droite, nous n'avons aucune dépendance entre les tests. Ils ont chacun leur processus pour traiter le test.

Figure 3
Mise en processus



4.1.4 Réactivation de tests

Après avoir récupéré les tests, le daemon vérifie pour tous les tests désactivés si la date de réactivation a été atteinte ou dépassée. Dans ce cas le test est réactivé et la date de réactivation remise à 0 (0000-00-00 00:00:00). Il faut définir une date de réactivation si l'on veut que la réactivation soit automatique. Si on désactive un test sans définir de date de réactivation (par défaut 0000-00-00 00:00:00) alors le test restera désactivé jusqu'à ce que le test soit réactivé manuellement (depuis l'interface web).

4.1.5 Exécution des tests

Pour chaque test, le daemon va vérifier s'il est actif et si la date et l'heure de la prochaine exécution a été atteinte/dépassée. Ensuite il vérifie que le script est bien présent dans le dossier des scripts sur le serveur. Dans ce cas, le daemon exécute le script et stocke la valeur de retour.

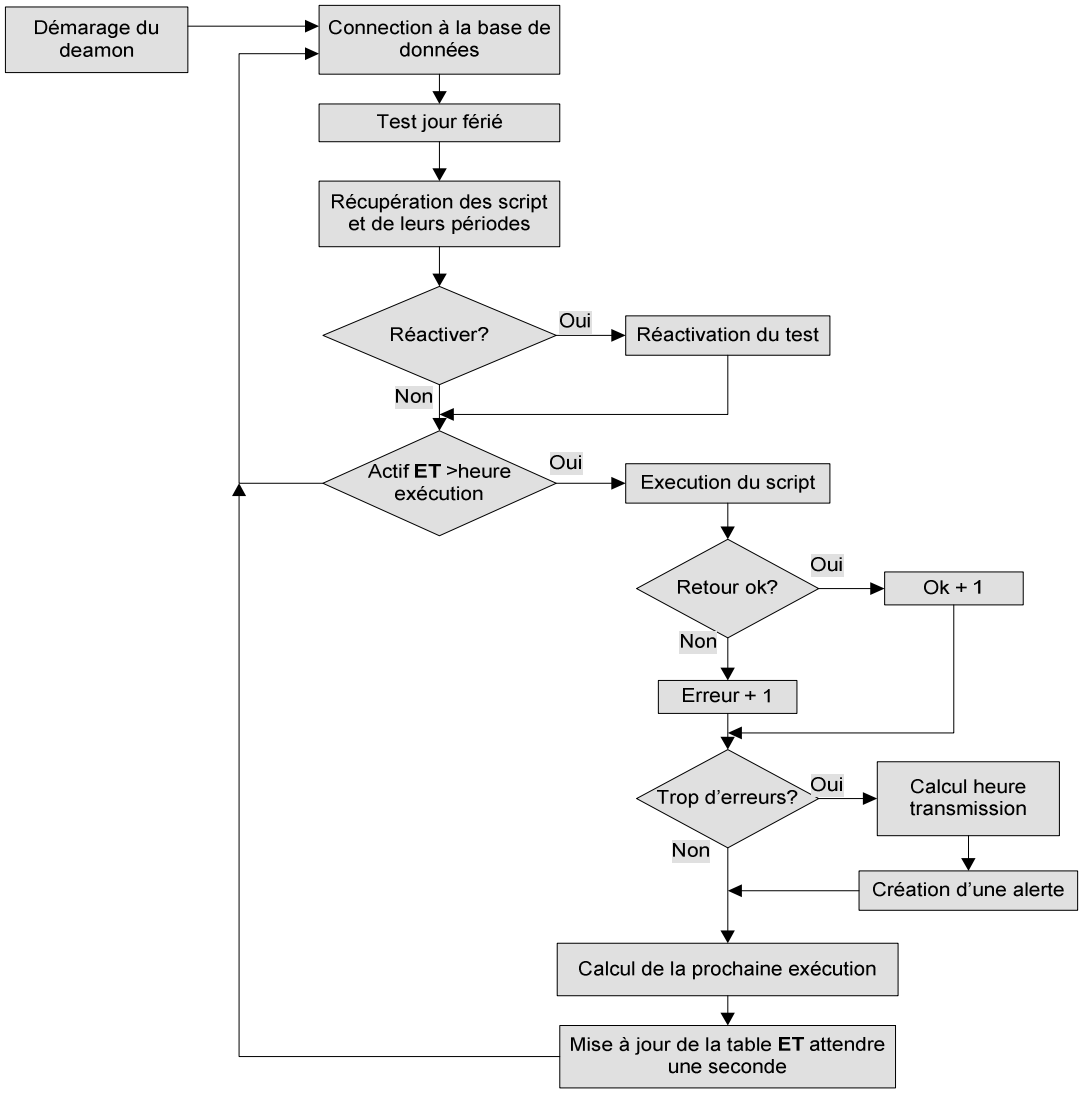
4.1.6 Analyse du code de retour

Une fois que le script a été exécuté et le code de retour stocké, on le compare avec la réponse attendue (stocké dans les critères du test, par défaut 0). Si celui-ci correspond, on incrémente le compteur de réponse positive. Dans le cas contraire on incrémente le compteur d'erreurs. Si le nombre d'erreurs atteint le nombre limite (défini dans le test) le daemon doit insérer une alerte dans la table TAlertes. Pour cela il faut calculer, en fonction des périodes de transmission, l'heure à laquelle l'alerte sera transmise pour être traitée par le daemon de Jonathan MALFOY.

4.1.7 Mise à jour des tests

Une fois le script exécuté, le daemon met à jour l'heure de la dernière exécution (maintenant) et l'heure de la prochaine exécution (maintenant + la fréquence en seconde du test).

Figure 4
Structogramme Daemon Execute



4.2 Daemon Critère

Ce daemon possède plusieurs responsabilités :

- Récupérer les alertes
- Mise en processus
- Transmission des alertes
- Mise à jour de l'alerte

4.2.1 Récupérer les alertes

Le daemon récupère toutes les alertes stockées dans la table TAlertes n'ayant pas encore été transmises (transmis = 0). Le champ transmis ne permet de déclencher qu'une seule mise en conférence. Si suite à la mise en conférence l'alerte est quittancée, le champ transmis repassera à 0 et la prochaine heure de transmission sera calculée par le trigger 'ajouterTemps' (cf. 7.2.2.).

4.2.2 Mise en processus

Pour chacune des alertes récupérées, un processus sera crée et lui sera attribué pour traiter l'alerte. Cela permet des traiter en parallèle toutes les alertes en évitant les problèmes de file d'attente.

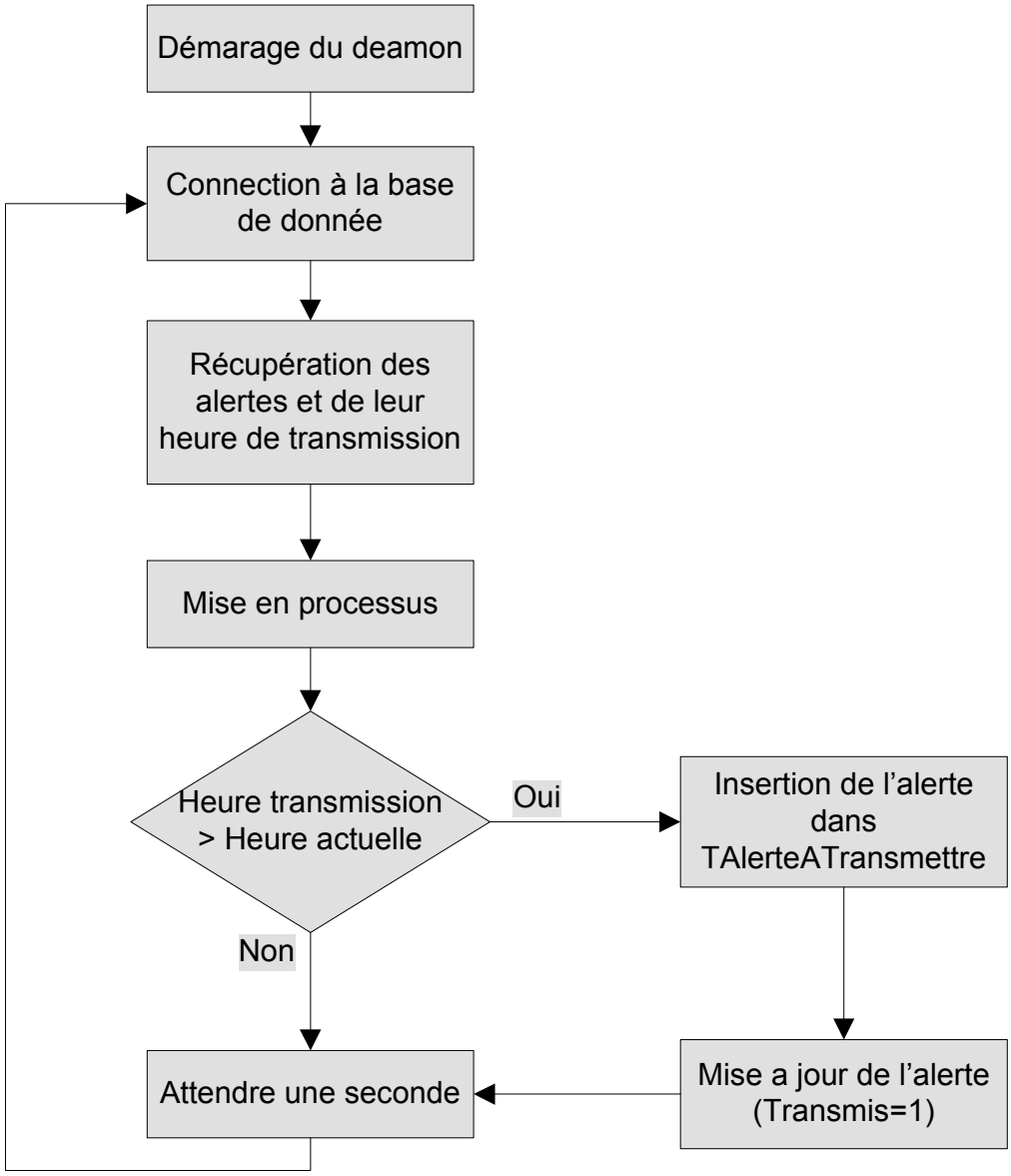
4.2.3 Transmission des alertes

Le daemon vérifie si l'heure de transmission a été atteinte/dépassée. Dans ce cas l'alerte est copiée dans la table TAlertesATransmettre. Une fois dans cette table, l'alerte sera traitée par le daemon de Jonathan MALFOY qui se chargera de mettre en conférence les personnes concernées par cette alerte.

4.2.4 Mise à jour de l'alerte

Une fois l'alerte transférée dans la table TAlertesATransmettre, le daemon met à jour l'alerte. Il passe le champ transmis à 1 pour signifier qu'elle a déjà été transmise. L'alerte ne sera supprimée que si le test atteint le nombre de réponses positives consécutives pour couper l'alarme (fait automatiquement par le trigger 'verifierCompteur' de la table TCriteresTest cf. 7.2.1).

Figure 5
Structogramme Daemon Critère



5. Journalisation complète des actions

Les alarmes nécessitent un traitement rigoureux de la journalisation. En effet, si une panne réseau ou une erreur système devait venir interférer le processus de traitement et de mise en conférence, les conséquences pourraient être relativement importantes selon la nature de l'alarme touchée. C'est pourquoi l'administrateur doit pouvoir retrouver facilement la faille ainsi que l'alarme en cause. Le moyen le plus sûr et le plus fiable est d'utiliser le protocole SYSLOG, déjà présent sur les systèmes Unix.

5.1 Protocole SYSLOG

Syslog est un protocole définissant un service de journaux d'événements d'un système informatique. C'est aussi le nom du format qui permet ces échanges. En tant que protocole, Syslog se compose d'une partie client et d'une partie serveur. La partie client émet les informations sur le réseau, via le port UDP 514. Les serveurs collectent l'information et se chargent de créer les journaux.

L'intérêt de Syslog est donc de centraliser les journaux d'événements, permettant de repérer plus rapidement et efficacement les défaillances d'ordinateurs présents sur un réseau.[1]

5.1.1 L'application RSYSLOG

Rsyslog est un daemon de journalisation de type syslogd ayant pour fonctionnalités principales le support de MySQL, syslog/tcp, RFC 3195, les listes d'expéditeurs autorisés, le filtrage sur n'importe quelle partie du message et un contrôle très fin du format de sortie[2]

[1] Définition - http://fr.wikipedia.org/wiki/Syslog

[2] Définition - http://wiki.monitoring-fr.org/integration/rsyslog

5.1.2 Formatage des informations pour l'application

Afin de mieux traiter et trier la journalisation de l'application, la mise en place d'un formatage spécifique pour les messages a été appliquée sur l'ensemble des données journalisées. L'application utilise trois types de messages syslog :

- INFO : Les messages de ce type sont utilisés pour la journalisation des événements faisant partie du processus normal du système. Par exemple : *« [INIT] Processus père lancé »*

  « [INIT] Initialisation de la connexion bdd »

- WARNING : Ces messages sont générés lorsqu'un événement sortant du déroulement normal de l'application a été produit comme, par exemple, une perte temporaire de la connexion mysql. Cela est un avertissement mais ne provoque pas un arrêt du daemon. Exemple :

  « [TRAITEMENT] Fichier inexistant »

  « [TRAITEMENT] Réponse incorrect »

- ERROR : Ces événements se déclarent lorsqu'un disfonctionnement a été détecté dans le système qui pourrait nuire au bon déroulement de l'application ainsi qu'à l'intégrité des données. En général, un événement ERROR stoppe le daemon.

Tous ces messages syslog sont préfixés par deux mots-clefs.

- [INIT] : Les événements se produisent lors d'une initialisation. Par exemple, lors de la création du processus père, l'initialisation de la connexion à la base de données.

- [TRAITEMENT] : Les événements se produisent pendant le traitement d'un critère. Par exemple, lors de la mise à jour de sa prochaine date d'exécution ou dès la mise à jour de ces compteurs.

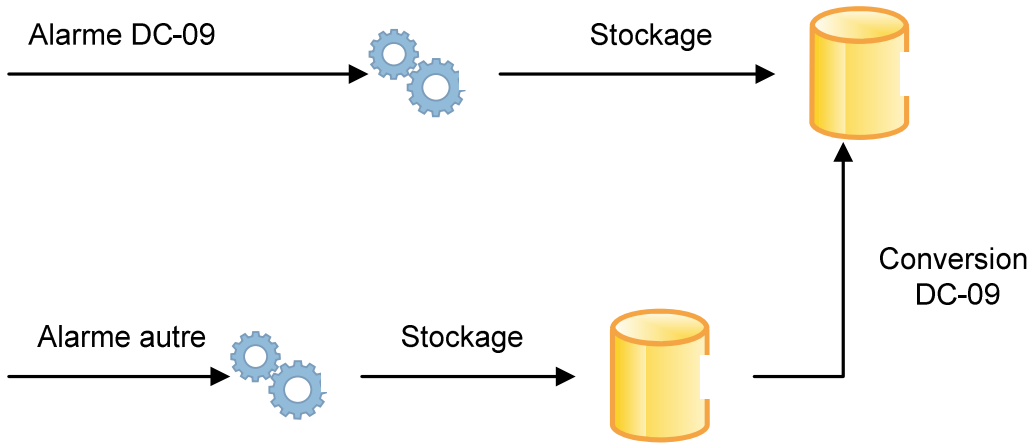
6. Réception des alarmes

Tous les équipements du réseau IP vont envoyer leur alarme sur le daemon de réception (contact sec, rainbow, snmp, etc...). Dans ces différents équipements nous devons différencier 2 groupes distincts :

- Les équipements envoyant leurs alarmes selon le standard DC-09
- Les équipements « autres »

Remarque : N'ayant pas eu d'équipement pour voir des messages DC-09, cette partie a été abandonnée. Nous n'avons donc pas de conversion des alarmes 'simples' en DC-09.

Figure 6
Réception des alarmes



6.1 Réception d'alarmes selon le standard DC-09

Des équipements transmettent leurs signaux d'alarmes selon la norme DC-09 (par exemple le Rainbow). Pour ces signaux, il suffit de prendre les données souhaitées (selon les paramètres rentrés dans l'interface web de gestion du DC-09) et les stocker dans la base de données pour être traitées.

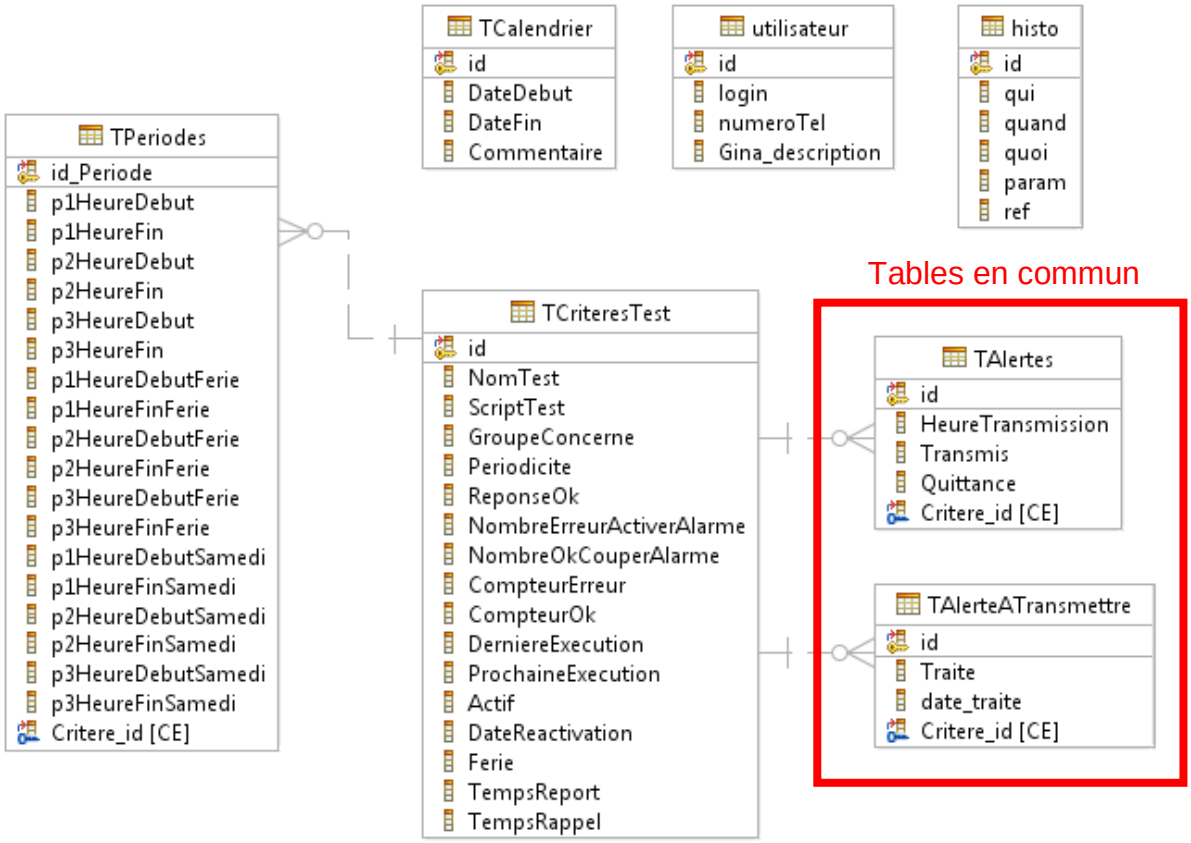
6.2 Réception d'alarmes « autres »

Pour les équipements transmettant dans un autre format que le DC-09. Il faut sélectionner les données « utiles » et correspondant à un des paramètres du DC-09 pour être converties et stockées selon la norme DC-09.

7. Base de données : Alarmes

La base de données ‘Alarmes’ contient les critères/tests à exécuter, leur période de transmission, le calendrier des jours fériés, les alertes en attente de transmission, les alertes transmises, les utilisateurs et l’historique pour les interfaces de gestion. Une fois qu’une alerte stockée dans la table TAlertes a atteint son heure de transmission celle-ci est copiée dans la table TAlertesATransmettre pour être traitée par le daemon de Jonathan MALFOY qui s’occupera de mettre en conférence les personnes concernées.

Figure 7
Schéma base de données



7.1 Tables

La base de données ‘Alarmes' possède sept tables. Certaines utilisées pour ce travail :

- TCriteresTest
- TAlertes *(en commun avec le travail de Jonathan Malfoy)*
- TAlertesATransmettre *(en commun avec le travail de Jonathan Malfoy)*
- TCalendrier
- TPeriodes
- histo
- utilisateur

7.1.1 Table TCriteresTest

Cette table permet aux utilisateurs d’insérer les paramètres de nouveaux tests que le système doit exécuter selon les paramètres rentré par ce dernier. Elle contient les champs suivants :

- Un identifiant **unique**
- Un nom de test et le nom du script présent sur le serveur
- Le groupe concerné pour administrer le test
- La fréquence du test en **secondes**
- Le nombre d’erreurs pour déclencher l’alerte
- Le nombre de réponses positives **consécutives** pour couper l’alerte
- Les compteurs de réponses positives et négatives
- L’état d’activité du test (0 inactif, 1 actif)
- La dernière et la prochaine exécution du script (calcul automatique)
- Une date de réactivation si jamais on désactive le test momentanément
- Le temps de report (en secondes) lors de la quittance d’une alerte
- Le temps avant rappel (en secondes) si personne ne répond

7.1.2 Table TAlertes

Cette table possède toutes les alertes détectées par le système. Elle contient les champs suivants :

- Le numéro du critère concerné
- L'heure de transmission pour la mise en conférence
- Un champ 'Transmis' pour savoir s'il a déjà été transféré dans la table TAlertesATransmettre pour être traité par Jonathan Malfoy
- Un champ 'Quittance' pour savoir si une alarme a été quittancée et le trigger calculera la prochaine mise en conférence si le problème persiste et que l'alarme est toujours active.

7.1.3 Table TAlertesATransmettre

Cette table reçoit les alarmes à traiter. C'est le travail du daemon à Jonathan MALFOY de récupérer ces alarmes, de mettre en conférence la liste de personnes concernées, de quittancer l'alarme si quelqu'un s'en occupe. Cette table contient les champs suivants :

- Le numéro du critère concerné
- 'traite' pour savoir si l'alarme a déjà été mise en conférence
- La date de traitement pour savoir quand il faudra relancer la mise en conférence si personne n'a quittancé l'alarme.

7.1.4 Table TCalendrier

Cette table contient toutes les dates à considérer comme fériées pour les heures de transmission des alarmes. Cette table est gérée depuis le module de gestion du calendrier de l'interface web (cf. 8.2). Elle contient les champs :

- Une date de début
- Une date de fin qui peut être identique au jour de début dans le cas d'un jour férié unique (ex : Jeune Genevois)
- Un commentaire (ex : Jeune Genevois)

7.1.5 Table TPeriodes

Cette table contient les périodes pour chacun des tests. Chaque test possède jusqu'à 3 périodes de transmission configurées pour chacun des états possibles (Normal, Samedi, Férié). Dans le cas d'une alarme critique de type incendie par exemple on n'aura qu'une seule période configurée qui couvre toute la journée (00 :00 :00 -> 23 :59 :59) car on ne peut pas se permettre d'attendre 7h pour avertir qu'un incendie s'est déclaré.

7.1.6 Table histo

Cette table permet d'avoir un historique de toutes les actions effectuées sur l'interface web (connexions, erreurs, modification, etc..). Lors du premier login si l'utilisateur n'est pas présent dans la table, le système va copier ces informations et créer un utilisateur. Cette table contient les champs suivant :

- L'identifiant de l'utilisateur concerné
- La date de l'événement
- La partie concernée par le message
- L'action qui a provoqué l'historisation
- Un numéro de référence

7.1.7 Table utilisateur

Cette table contient les utilisateurs s'étant connectés à l'interface web.

- Un identifient unique
- Un login
- Le numéro de téléphone
- Une description GINA

7.2 Triggers

Plusieurs triggers on été mis en place au sein de la base de donnée :

- ‘verifierCompteur’ sur la table TCriteresTest
- ‘ajouterTemps’ sur la table TAlertes

7.2.1 Trigger verifierCompteur

Il permet de vérifier le compteur de bonnes réponses consécutives (compteurOk). Si ce dernier atteint le nombre de bonnes réponses pour couper l’alarme, il supprime l’alarme présente dans la table TAlertes ainsi que dans la table TAlertesATransmettre. Ceci évite que le daemon de Jonathan MALFOY remette en conférence alors que le critère d’alarme n’est plus rempli. Le trigger remet également à zéro les compteurs d’erreurs et de bonnes réponses.

7.2.2 Trigger ajouterTemps

Il permet de calculer la prochaine mise en conférence quand une alarme est quittancée. Pour cela il récupère le type du jour dans la table TCriteresTest (0 = Normal, 1 = Férié, 2 = Samedi). Ensuite, en fonction du type de jour récupéré, il récupère les périodes de transmission correspondant et le temps de report pour cette alarme si elle est quittancée. Quand une alarme est quittancée le trigger ajoute le temps de report à l’heure actuelle, vérifie si cela tombe dans une des périodes de transmission dans le cas contraire il met l’heure de début de la prochaine période. Si le report dépasse la dernière période ou nous fait changer de jour alors on met l’heure de début de la première période du lendemain en tenant compte du type de jour (ouvrable, samedi, férié/dimanche)

7.3 Stockage des alertes dans la BDD

Il y a deux types d'alertes :

- Les alertes détectées (en attente de transmission)
- Les alertes transmises

7.3.1 Les alertes détectées

Une fois qu'un critère de test a atteint le nombre maximum d'erreurs tolérées, le système calcule l'heure de transmission de l'alerte en fonction de ses périodes et du type de jour (Ouvrable, Samedi, Férié), il stocke le numéro du critère avec son heure de transmission dans la table TAlertes. Une fois l'heure de transmission atteinte le système transfert l'alerte dans la table TAlertesATransmettre et met à jour l'alerte en mettant le champ transmis à 1. Si lors de la mise en conférence, gérée par Jonathan MALFOY, l'alerte est quittancée alors une nouvelle heure de transmission est calculée et le champ transmis est remis a 0. Si le problème est corrigé avant la prochaine transmission l'alerte sera supprimée vu que le nombre de réponses positives consécutives pour couper l'alerte sera atteint.

7.3.2 Les alertes transmises

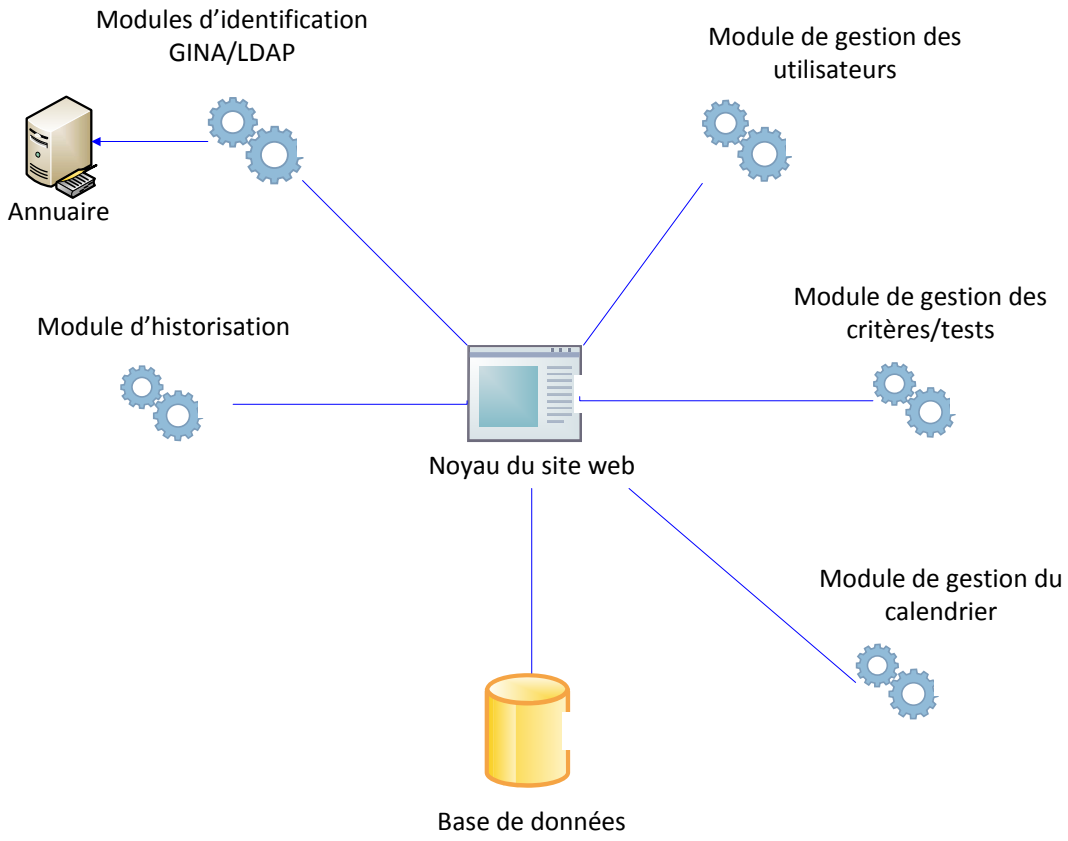
Une fois le critère inséré dans la table TAlertesATransmettre, le daemon de Jonathan MALFOY traite l'alerte immédiatement. Il se charge de mettre en conférence les personnes concernées par ce critère, de vérifier si une personne quittance l'alarme et de rappeler ces dernières si nul ne répond. Une fois l'alerte quittancée, il met à jour la table TAlertes pour calculer la prochaine transmission et supprime l'alerte dans la table TAlertesATransmettre. Ainsi, si le problème n'est pas corrigé d'ici la nouvelle heure de transmission, le système relancera la mise en conférence.

8. Interface Web

Le système possède 5 modules :

- Un module pour la gestion des critères/test
- Un module pour la gestion du calendrier
- Un module pour la gestion des utilisateurs
- Un module d'identification LDAP et GINA
- Un module d'historisation

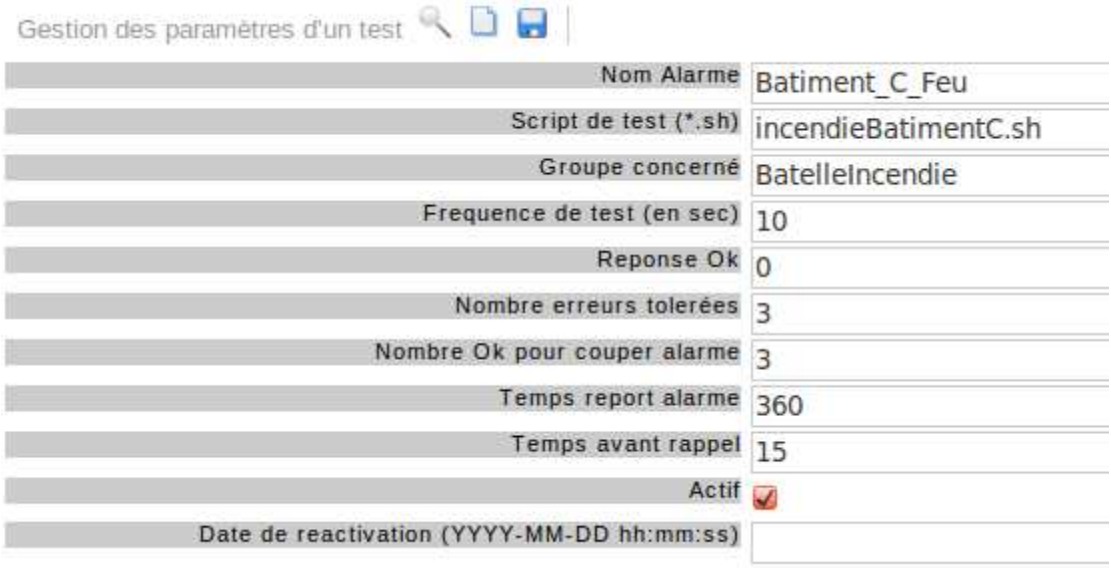
Figure 8
Schéma interface web



8.1 Gestion des critères/tests

L'interface permet aux administrateurs d'ajouter/modifier/supprimer des tests que le système devra effectuer. La première partie des paramètres concernent la configuration 'globale' du test. Il faut inscrire le nom du test, le nom du script à exécuter, le groupe concerné par ce test et qui pourra le modifier (ex : BatelleIncendie), la fréquence d'exécution en secondes, la réponse attendue (par défaut 0), le nombre d'erreurs pour déclencher l'alarme, le nombre de réponses positives **consécutives** pour couper la condition d'alarme, le temps de report quand une alarme est quittancée en secondes, le temps en secondes avant rappel si personne ne quittance l'alarme, activer/désactiver le test. Le champ pour la date de réactivation n'est à remplir que si l'on désactive le test et que l'on désire qu'il se réactive automatiquement. Cela nous permet, par exemple, de désactiver les tests d'un bâtiment pendant la durée des travaux.

Figure 9
Gestion des critères partie 1



La deuxième partie concerne les périodes de transmission de l'alarme. Ces périodes permettent d'éviter de déclencher une alarme et de mettre en conférence les personnes concernées en pleine nuit pour une alarme 'mineure'. On peut avoir jusqu'à 3 périodes par jour. Il y a 3 périodes pour chacun des 3 types de jours possibles :

- Normal : Horaire du lundi au vendredi hors jours fériés
- Férié : Horaire pour le dimanche et les jours fériés définis dans le calendrier
- Samedi : Horaire pour les samedis hors jours fériés

Figure 10
Gestion des critères partie 2

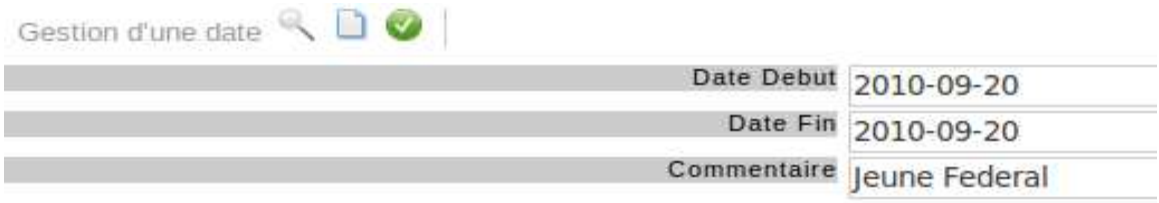
8.2 Gestion du calendrier

La partie gestion du calendrier permet aux administrateurs de gérer les jours fériés. Le calendrier est global et donc appliqué à **tous** les tests. Si l'on souhaite avoir un calendrier pour chaque test, il faut ajouter une table intermédiaire entre la table TCriteresTest et TCalendrier. L'administrateur peut ajouter un nouveau férié, modifier et supprimer un férié existant. Un utilisateur normal de l'application ne pourra que consulter les jours fériés configurés par l'administrateur.

Figure 11
Liste des fériés

Un jour férié contient une date de début ainsi qu'une date de fin et un commentaire. Pour des fériés d'un jour (ex : Jeune Fédéral) il suffit d'indiquer la même date en début et fin. Le commentaire est un champ texte libre pour expliquer pourquoi c'est un jour férié.

Figure 12
Gestion d'un férié



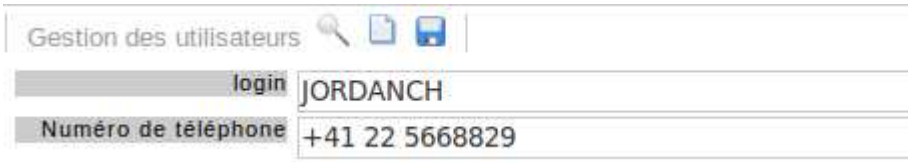
8.3 Gestion des utilisateurs

La partie gestion des utilisateurs permet de gérer les utilisateurs s'étant connectés au moins une fois à l'interface web. Cette partie n'est accessible qu'aux administrateurs. Elle leur permet d'ajouter, de modifier ainsi que de supprimer des utilisateurs.

Figure 13
Liste des utilisateurs

Quand un administrateur ajoute ou modifie un utilisateur, on arrive sur la page suivante. Elle lui permet de modifier le login ainsi que son numéro de téléphone et d'enregistrer les modifications.

Figure 14
Gestion d'un utilisateur



8.4 Identification LDAP / GINA

L'ensemble du système de gestion est restreint aux ayant droits. A la première connexion, l'utilisateur doit s'authentifier au moyen d'un protocole défini selon son type de compte (GINA ou LDAP)

8.4.1 GINA

Gina est un protocole d'identification propriétaire de l'État de Genève. L'utilisateur est redirigé vers une page dédiée et s'authentifie à l'aide d'un couple login/mot de passe de son compte GINA.

Figure 15

Page d'authentification GINA

8.4.2 LDAP

LDAP (Lightweight Directory Access Protocol) est un protocole permettant l'interrogation et la modification des services d'annuaire. Dans le cadre de l'application de gestion, LDAP va authentifier l'utilisateur avec le service d'annuaire Active Directory. A la connexion, l'utilisateur doit rentrer son login (préfixé de son domaine, par exemple, « GE-EM ») et son mot de passe à partir d'une simple fenêtre de connexion html.

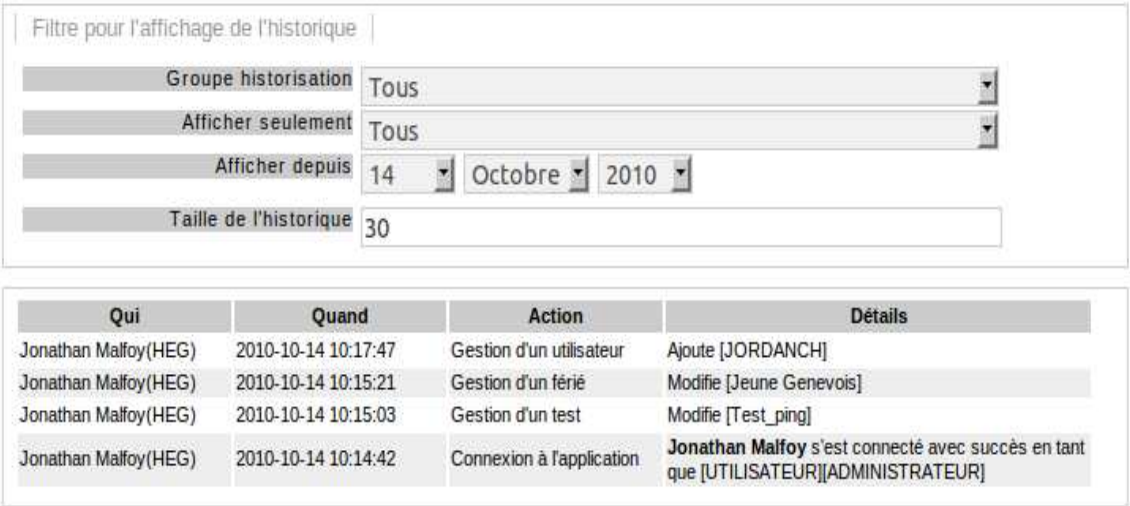
Figure 16

Fenêtre d'authentification LDAP

8.5 Historique

Toute actions effectuée depuis l'interface web est insérée dans la table 'histo'. On y stocke l'auteur, la date et l'heure, l'action effectuée et les détails de l'action effectuée. Par exemple si l'administrateur Jonathan MALFOY se connecte à l'interface, modifie un test ainsi qu'un férié et qu'il ajoute un utilisateur il y aura les enregistrements suivants.

Figure 17
Historique



Qui	Quand	Action	Détails
Jonathan Malfoy(HEG)	2010-10-14 10:17:47	Gestion d'un utilisateur	Ajoute [JORDANCH]
Jonathan Malfoy(HEG)	2010-10-14 10:15:21	Gestion d'un férié	Modifie [Jeune Genevois]
Jonathan Malfoy(HEG)	2010-10-14 10:15:03	Gestion d'un test	Modifie [Test_ping]
Jonathan Malfoy(HEG)	2010-10-14 10:14:42	Connexion à l'application	**Jonathan Malfoy** s'est connecté avec succès en tant que [UTILISATEUR][ADMINISTRATEUR]

8.6 Droits d'accès

Les opérations d'ajout, de suppression et de modification des informations de chaque module de l'interface de gestion ne doivent pas être attribuées à n'importe quel utilisateur authentifié. C'est pourquoi un système de gestion par rôles et groupes a été introduit. Les rôles et les groupes auxquels appartient chaque utilisateur est récupéré dans l'annuaire du compte de l'utilisateur.

8.6.1 Les différents rôles

L'application peut être administrée selon deux rôles :

- **L'administrateur** : L'utilisateur possédant les droits d'administrateur a le contrôle total de l'interface. Il peut non seulement, effectuer des ajouts, suppression et modifications sur l'ensemble des données, mais aussi voir en tout temps l'historique d'utilisation des utilisateurs.
- **L'utilisateur de base** : Un utilisateur de base est limité. Il ne peut gérer que les critères concernant ces groupes (récupérés dans l'annuaire). Par exemple s'il appartient au groupe « Effraction_Battelle », il pourra gérer tous les critères de test qui ont comme groupe concerné « Effraction_Battelle ».

8.6.2 La gestion par groupe

Chaque critère de test possède un groupe concerné (le groupe pouvant l'administrer). Lors de la création d'un test, l'administrateur peut spécifier le groupe d'utilisateurs ayant les droits d'administration sur ce test. Chaque utilisateur faisant partie de ce groupe pourra modifier le test ainsi que ses périodes de transmission. Le groupe d'administration ne pourra pas supprimer la liste. Le principal avantage de déléguer les droits d'administration sur les listes est que l'utilisateur n'aura pas à faire appel à l'administrateur global pour chaque modification 'mineure' d'un test. Par exemple, si l'on veut modifier la fréquence d'exécution du test de 12 à 15 secondes.

Conclusion

Pour conclure, la recherche et la documentation des nouvelles technologies étaient très instructives et intéressantes. Malheureusement, la norme DC-09 :2007 n'a pas pu être intégrée au projet car le simulateur d'équipements envoyant des signaux en DC-09 n'a pas pu être obtenu. Maintenant grâce à l'utilisation de logiciels libres, la maintenabilité du programme est garantie et donc une mise à niveau pour intégrer la norme DC-09 est tout à fait envisageable par la suite en rajoutant un module de conversion des alarmes 'standard' en message DC-09. Cette norme permettrait d'avoir un format unifier de toutes les alarmes.

L'immersion dans le monde des alarmes sur IP a été très passionnante. Comment tester une sonde ? Les différents états d'une alarme ? Quels critères a-t-on besoin pour un test ? Comment gérer les différents niveaux de criticités des alarmes ? Toutes des questions qu'il a fallu étudier durant tout le projet afin d'avoir, au final, un programme au plus près des besoins que l'on nous a fournis.

La création de l'interface web aussi fut fort intéressante. Pour permettre aux utilisateurs de modifier les tests les concernant et aux administrateurs d'avoir un accès total, une gestion des droits a été mise en place. Pour accéder à l'interface une page authentification a été mise en place. Une fois authentifié, le programme lui attribue ces droits.

Bibliographie

Documentation MySql en ligne, http://dev.mysql.com/doc/refman/

La norme ANSI/SIA DC-09-2007, http://www.hacker-soft.net/tools/Assessment/dc09_20070319.pdf

Un complément technique sur l'application de la norme DC-09 dans le canton de Vaud www.**eca**-vaud.ch/prevenir/pdf/451170_complement_technique.pdf

L'encyclopédie « Wikipedia » en ligne pour certaines définitions, http://fr.wikipedia.org

Le forum de http://www.commentcamarche.net/ pour divers problèmes de codage.

Annexe 1
Manuel d'installation

L'installation a été testée sur le système d'exploitation **Debian 5.0.5 64 bits**

Configuration d'Asterisk 1.4

Installez les paquets suivants :

« libmime-lite-perl », requis pour l'envoi de mail

« libasterisk-agi-perl », prise en charge du langage PERL pour les scripts AGI

Fichier */etc/asterisk/extensions.conf*

Ajoutez la macro suivante dans le fichier de configuration :

```
[macro-dialout_callmanager]
exten => s,1,Dial(SIP/${ARG1}@cmtr001)
exten => s,n,Dial(SIP/${ARG1}@cmtr004)
exten => s,n,Dial(SIP/${ARG1}@cmtr005)

exten => s,n,Congestion
```

Ainsi que l'extension suivante :

```
[out]
exten => _0[2-9]XXXXXXXX,1,Macro(dialout_callmanager,${EXTEN})
exten => _00[2-9]XXXXXXXX,1,Macro(dialout_callmanager,${EXTEN:1})
exten => _8XXXX,1,Macro(dialout_callmanager,02238${EXTEN})
exten => _6[1-8]XXX,1,Macro(dialout_callmanager,02254${EXTEN})
exten => _5XXXX,1,Macro(dialout_callmanager,02232${EXTEN})
exten => _7XXXX,1,Macro(dialout_callmanager,02232${EXTEN})
```

Ensuite copier les deux extensions suivantes nécessaires pour l'enregistrement audio et la mise en conférence :

```
[record_conferenceWav]
exten => s,1,Answer()
exten => s,n,Wait(2)
exten => s,n,Playback(${INTRO_RECORD})
exten => s,n,Record(${FILE_RECORD},,120)
exten => s,n,Playback(beep)
exten => s,n,Wait(2)
exten => s,n,Hangup()

[conference_alarme]
exten => s,1,Set(CHANNEL(language)=fr)
exten => s,n,Answer()
exten => s,n,Wait(2)
exten => s,n,Playback(${INTRO_CONF})
exten => s,n,WaitExten(10)
exten => #,1,AGI(logalarme.agi|${NOM_ALARME}|${LOGIN}|est entre en conference)
exten => #,n,MeetMe(${NUM_CONF},cdMFX)
exten => #,n,Hangup()
exten => *,1,AGI(quittance.agi|${ID_CRITERE})
exten => *,n,AGI(logalarme.agi|${NOM_ALARME}|${LOGIN}|a quittance)
exten => *,n,MeetMeAdmin(${NUM_CONF},K)
exten => *,n,Playback(${QUITTANCE_CONF})
exten => *,n,Hangup()
exten => t,1,Playback(${GOODBYE_CONF})
exten => t,n,Hangup()
exten =>
h,1,DeadAGI(sendmail.agi|${NOTIFICATION}|${EMAIL}|${NOM_ALARME}|${LOGIN})
exten => h,n,MeetMeCount(${NUM_CONF}|count)
exten => h,n,Gotoif,$[${count} = 0}]?103
exten => h,103,DeadAGI(end.agi|${NOM_ALARME})
```

Les mots en rouge peuvent être modifiés. Par défaut, # est utilisé pour rentrer en conférence et * pour quittancer.

Fichier */etc/asterisk/sip.conf*

S'assurer que la méthode d'envoi DTMF est bien rfc2833

```
dtmfmode=rfc2833
```

Scripts AGI

Copiez les scripts AGI (disponible dans le répertoire « agi-bin » du dossier d'installation) dans le répertoire */usr/share/asterisk/agi-bin* et mettre les droits root (chown) sur tous les fichiers.

Si besoin est, vous pouvez changer la **configuration de l'envoi de mail** dans le fichier « *sendmail.agi* » ainsi que **l'envoi de quittance** dans le fichier « *quittance.agi* » (Attention, si vous changez le port et l'adresse, il ne faudra pas oublier de le modifier aussi dans le daemon de quittance)

Installation des daemons

Installez la librairie de prise en charge mysql pour le langage c « libmysqlclient-dev »

Daemon alarme

Dans le fichier *« daemon_alarmes.c »* modifiez les constantes de connexion aux bases de données.

```
#define SERVER

#define USER

#define PASS

#define BDDALARMES

#define BDDLISTES
```

Éditez le fichier « *Makefile* » et changez la ligne de compilation selon le répertoire d'installation du daemon.

Compilez le daemon avec la commande make all.

Daemon quittance

Dans le fichier *« daemon_quittance.c »* modifiez les constantes de connexion aux bases de données.

#define SERVER

#define USER

#define PASS

#define BDD

#define PORT (port de connexion socket)

Éditez le fichier « *Makefile* » et changez la ligne de compilation selon le répertoire d'installation du daemon.

Compilez le daemon avec la commande make all.

Daemon exécute

Dans le fichier *« main.cpp »* modifiez les constantes de connexion aux bases de données.

#define SERVER

#define USER

#define PASS

#define BDD

#define CHEMIN (répertoire d'exécution des scripts)

#define CHEMINSIMPLE (pour tester l'existence des scripts)

Compilez le daemon avec la commande

g++ main.cpp –lmysqlcient –o daemon_execute

Daemon critère

Dans le fichier *« main.cpp »* modifiez les constantes de connexion aux bases de données.

```
#define SERVER

#define USER

#define PASS

#define BDD
```

Compilez le daemon avec la commande

```
g++ main.cpp –lmysqlcient –o daemon_critere
```

Monit

Ajoutez les lignes suivantes dans le fichier de configuration /etc/monit/monitrc sur les deux serveurs :

```
check process heartbeat with pidfile /var/run/heartbeat.pid
      .start program = "/etc/init.d/heartbeat start"
       stop  program = "/etc/init.d/heartbeat stop"
       depends asterisk, mysql

check process asterisk with pidfile /var/run/asterisk/asterisk.pid
       start program = "/etc/init.d/asterisk start"
       stop program  = "/etc/init.d/asterisk stop"
       if failed port 5060 type udp then restart

check process mysql with pidfile /var/run/mysqld/mysqld.pid
       start program = "/etc/init.d/mysql start"
       stop program = "/etc/init.d/mysql stop"
       if failed port 3306 protocol mysql then restart
```

Puis ces lignes sur le serveur principal (un pour chaque daemon):

```
check process service with pidfile /var/run/nom_du_daemon.pid
        start program = "/etc/init.d/nom_du_daemon"
      stop program = "/sbin/start-stop-daemon –stop –pidfile
/var/run/nom_du_daemon.pid "
       if failed port 80 protocol tcp then start
```

Changez les mots en rouge pour chaque daemon et selon le répertoire d'installation des daemons.

Attention, le daemon_alarme doit impérativement se lancer en root (sudo) du à une contrainte Asterisk sur les fichiers d'appel.

Répétez l'opération sur l'autre serveur mais en changeant la dernière ligne :

```
check process service with pidfile /var/run/nom_du_daemon.pid
        start program = "/etc/init.d/nom_du_daemon"
       stop program = " /sbin/start-stop-daemon –stop --pidfile /var/run/
nom_du_daemon.pid "
        if failed host ip_du_serveur_principal port 80 protocol tcp then start
        if host ip_du_serveur_principal port 80 protocol tcp then stop
```

Interface de gestion

Copier les répertoires gest_listes et gest_test vers le répertoire /var/www/

Exécutez (en root) les scripts SQL contenu dans le répertoire /sql du dossier d'installation.

Lancement des daemons

Utilisez la commande **nohup** pour lancer les daemons. Exemple :

```
nohup ./daemon_quittance &
```

Annexe 2
Manuel d'utilisation

Les droits

Mis à part l'administrateur global qui possède tous les privilèges, les droits sont attribués comme suit :

	Gestions des utilisateurs	Gestion des Tests	Gestion du calendrier	Historique
Consultation		Administrateur du test, Utilisateur*	Utilisateur	
Ajout				
Modification		Administrateur du test, Utilisateur*		
Suppression				

***** : ne peut consulter/modifier que les tests pour lesquels il est concerné.

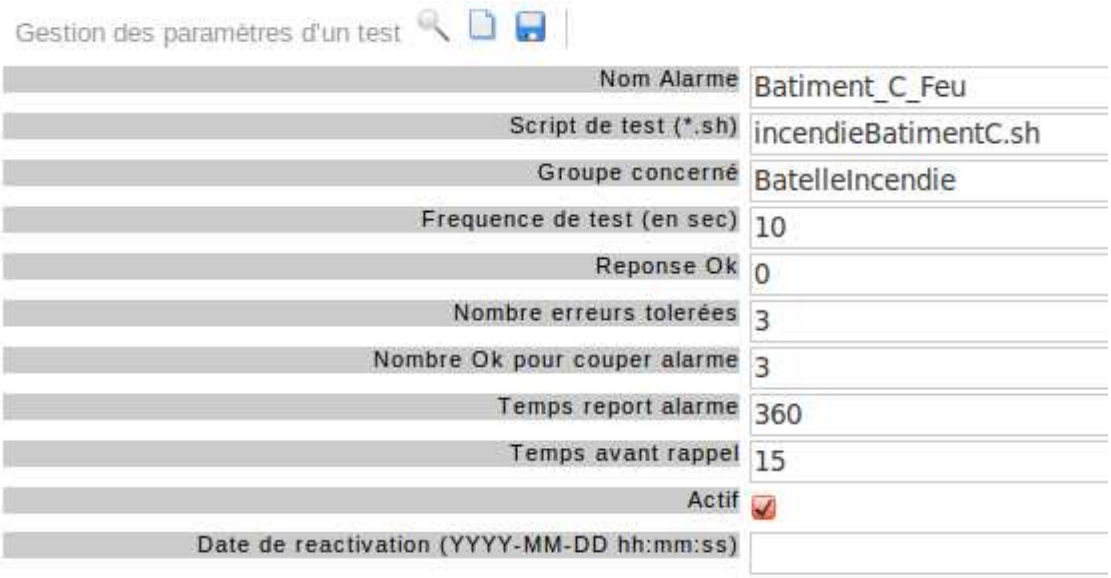
Gestion des critères de test

L'interface permet aux administrateurs d'ajouter / modifier / supprimer des tests que le système devra effectuer. Les utilisateurs de base peuvent uniquement modifier les tests si son groupe est le groupe concerné du test.

Gestion des Tests

Nom Alarme	Script	Groupe concerné	Fréquence	Reponse ok	Actif?	Reactivation	Modifier	Supprimer
Test1	test.sh	LST-HES-HEG-IG-ETU	10	0	Non	0000-00-00 00:00:00		
Test2	test2.sh	Effraction	10	0	Non	0000-00-00 00:00:00		
Test_ping	test3.sh	ADMINISTRATEUR	10	0	Non	0000-00-00 00:00:00		
test_jordan	test3.sh	ADMINISTRATEUR	5	0	Non	0000-00-00 00:00:00		
Test4	test3.sh	ADMINISTRATEUR	10	0	Non	0000-00-00 00:00:00		
Test_Malfoy	test3.sh	ADMINISTRATEUR	4	0	Non	0000-00-00 00:00:00		

Les premiers paramètres concernent la configuration ‘globale’ du test. Il faut inscrire le nom du test, le nom du script à exécuter, le groupe concerné par ce test et qui pourra le modifier (ex : BatelleIncendie), la fréquence d’exécution en secondes, la réponse attendue (par défaut 0), le nombre d’erreurs pour déclencher l’alarme, le nombre de réponses positives consécutives pour couper la condition d’alarme, le temps de report quand une alarme est quittancée en secondes, le temps en secondes avant rappel si personne ne quittance l’alarme, activer/désactiver le test. Le champ pour la date de réactivation n’est à remplir que si l’on désactive le test et que l’on désire qu’il se réactive automatiquement. Le bouton permet d’enregistrer le nouveau test, le bouton permet de remettre les champs à leur valeur par défaut et le bouton permet de retourner à la liste test.



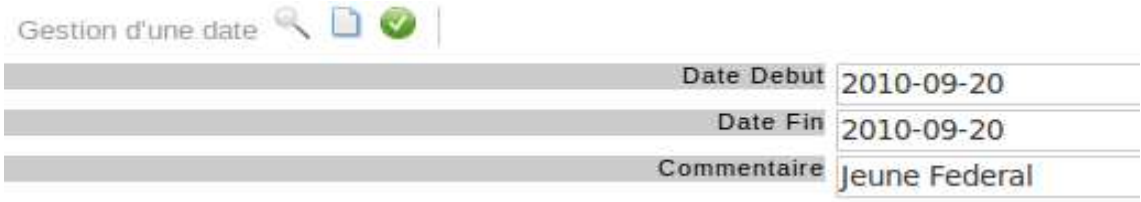
La deuxième partie des paramètres concerne les périodes de transmission de l'alarme. Ces périodes permettent d'éviter de déclencher l'alarme et une mise en conférence des personnes concernées en pleine nuit pour une alarme 'mineure'. On peut avoir jusqu'à 3 périodes par jour. Dans cet exemple une seule période est configurée. Elle couvre toute la journée (typiquement pour une alarme feu). Il y a 3 périodes pour chacun des 3 types de jours possibles :

- Normal : Horaire du lundi au vendredi hors jours fériés
- Férié : Horaire pour le dimanche et les jours fériés définis dans le calendrier
- Samedi : Horaire pour les samedis hors jours fériés

Gestion du calendrier

L'ajout, la modification et la suppression d'un férié ne peuvent être effectués que par des administrateurs. L'interface permet d'ajouter un nouveau jour férié grâce au bouton , de modifier un jour férié déjà inséré avec le bouton et de supprimer un jour férié avec le bouton .

Un jour férié contient une date de début ainsi qu'une date de fin et un commentaire. Pour des fériés d'un jour (ex : Jeune Fédéral) il suffit d'indiquer la même date en début et en fin. Lors d'une modification d'un férié, le bouton permet d'enregistrer les modifications. Lors d'un ajout, c'est le bouton qui permet d'enregistrer le nouveau férié. Le bouton permet de remettre les champs à leur valeur par défaut et le bouton permet de retourner à la liste des jours fériés.



Gestion des utilisateurs

La partie gestion des utilisateurs permet des gérer les utilisateurs s'étant connectés à l'interface web. Cette partie n'est accessible qu'aux administrateurs. Elle permet de modifier , supprimer ainsi qu'ajouter un utilisateur.

Quand un administrateur ajoute ou modifie un utilisateur, on arrive sur la page suivante. Elle lui permet de modifier le login ainsi que le numéro de téléphone et d'enregistrer les modifications. Lors d'une modification, le bouton permet d'enregistrer les modifications tandis que lors d'un ajout c'est le bouton qui permet d'enregistrer le nouvel utilisateur.