France Telecom
en partenariat avec
l'Ecole Nationale Supérieure des Mines de Paris

Gestion centralisée des solutions de supervision Spatrol

Mémoire

Novembre 2008

MASTERE SPECIALISE
en
INGENIERIE PRODUCTION
et
INFRASTRUCTURES EN SYSTEMES OUVERTS
(IPISO)

Table des matières

Contexte personnel

Avant propos

Le projet que j'ai choisi de présenter dans le cadre de ce rapport est un projet technique et structurel. J'ai estimé qu'il entrait pleinement dans le cadre d'un rapport sur l'Ingénierie Production et Infrastructure des Systèmes Ouverts, mais 2 éléments sont à prendre en compte :

- J'ai travaillé sur ce projet durant 3 ans, cette période a été dense et riche.
- Ce projet n'était pas terminé lorsque j'ai quitté France Telecom en janvier 2008. Il était néanmoins arrivé à une maturation permettant un usage par différents services. J'ai depuis évolué vers un métier différent.

Ainsi, j'ai préféré présenter un projet encore en cours d'amélioration et complet d'un point de vue technique plutôt qu'un travail terminé, mais dont la consistance aurait été moindre.

Extrait de mon *curriculum vitae*

J'ai travaillé chez France Télécom entre le 7 octobre 2002 et le 7 janvier 2008. J'ai été recruté dans le cadre de la formation IPISO[1]. Je possède initialement une formation d'ingénieur système et réseau de l'ESME Sudria obtenu en 2002. Par goût pour l'ingénierie système, j'ai été attiré par l'offre qui m'avait été faite par France Télécom pour accéder, moyennant 6 mois de formation, à un poste d'expert système au SNPI.

Formations reçues durant ces 6 mois :

Novembre 2002	BeijaFlore	Programmation Shell et Filtre
Novembre 2002	BeijaFlore	Administration Système Unix Multi-OS
Décembre 2002	BeijaFlore	Administration LVM sur AIX et HPUX
Janvier 2003	HP	Administration Système et Réseau sous HPUX I
Février 2003	BeijaFlore	Réseau sous TCP/IP, configuration et administration.
Mais 2003	IBM	MQSeries Présentation Technique
Juin 2003	BMC Software	Patrol Scripting Language
Juin 2003	IBM	MQSeries : Mise en œuvre sur plate-formes distribuées

Ces formations complétées de tutorats à l'école de la production m'ont permis d'appréhender dans les meilleures conditions mon premier poste « d'expert technique ».

Après avoir intégré DTO, début 2003, j'ai été positionné dans l'équipe gérant le support technique de niveau 3 sur les outils de supervision et sur MQSeries pour les applications nationales.

J'ai été plus particulièrement en charge du soutien et du déploiement de l'agent Patrol en version 3.4 ce qui m'a permis de monter en compétence rapidement sur ce domaine. Nous avions à l'époque un parc de 750 serveurs environ pour une équipe de 3 personnes. Durant cette période, j'ai découvert l'architecture et l'environnement de supervision. J'ai appris à maîtriser le fonctionnement des agents Patrol ainsi que celui des consoles de supervision. A l'époque, aucun concentrateur d'évènements (hyperviseur) n'était mis en place.

[1] Tous les acronymes sont traduit dans le Glossaire (page 60).

Ce service a ensuite fusionné avec les services régionaux. Nous avons eu alors au sein du PEOS (15 personnes) à gérer et à maintenir l'ensemble des équipements permettant de mener à bien la surveillance des serveurs du système d'information de France Télécom. C'est à cause de cette augmentation du parc qu'il a fallu mettre en place un outil de référentiel de supervision. Nous avions aussi en charge le support et le soutien technique aux équipes de production. Nous étions le dernier niveau de soutien sur les outils de supervision avant l'éditeur.

L'étape SPatrol

Fin 2005, mon objectif était d'industrialiser la gestion du référentiel des serveurs supervisés du système d'information. J'ai, en concertation avec l'équipe, travaillé à l'élaboration et au développement de cet outil appelé « Spatrol ». J'ai de ce fait pu mettre pleinement en application les connaissances acquises lors des formations reçues dans le cadre du programme IPISO.

C’est ce projet, dont la définition, les développements et la mise en place ont duré 3 ans que j’ai choisi de présenter.

Ma vie professionnelle après France Telecom

J’ai depuis choisi de changer de société pour évoluer vers un poste de responsable informatique dans une usine de production agro-alimentaire. Ceci est motivé par 3 raisons qui se complètent :

- Être en contact direct avec des utilisateurs et les former aux outils informatiques,
- Travailler dans le domaine de l'informatique industrielle,
- Gérer un budget informatique.

Présentation globale du projet

Qu'est ce que la supervision ?

L'informatique est au cœur de l'entreprise. On peut facilement comparer la place que joue l'informatique au sein d'une entreprise à celle que joue le système nerveux chez l'être humain. En effet, il est au centre de l'activité, et doit fonctionner pleinement et en permanence pour garantir l'activité.

C'est pour cette raison que les problèmes liés à l'informatique doivent être réduits au minimum, car une indisponibilité du système d'information peut-être la cause de pertes énormes. Il faut donc garantir la disponibilité du système en cas de panne (par des mécanismes de redondance...) mais aussi tenter de prévenir en cas de problème et, le cas échéant, garantir une remontée d'information rapide réduisant le délai d'intervention au minimum : c'est le rôle de la supervision.

Enjeu, intérêt et contexte

A mon arrivée en octobre 2002, la supervision chez France Télécom était gérée par deux entités distinctes. D'un côté, historiquement, les USEI géraient de manière indépendante les serveurs des directions régionales, de l'autre, le SNPI exploitait les applications nationales. A cette époque, chaque service avait en charge la supervision de ses machines. Les 750 serveurs exploités par le SNPI étaient supervisés au travers d'une console Patrol. Les USEI disposaient de 7 consoles pour 4000 agents.

En décembre 2003, ces deux services ont fusionné. Nous avons donc du administrer tous ces agents et les équipements associés de manière industrielle. Sans solution de référencement centralisée et automatisée, le PEOS n'aurait pu mener à bien une consolidation des différentes consoles. Nous avons donc du analyser cette problématique afin de mettre en place une solution adaptée. Dans un premier temps, une étude a été demandée à une société de service pour développer un produit en phase avec nos besoins. Finalement, en raison de la réactivité dont nous avions besoin pour la prise en compte des améliorations, il a été décidé de réaliser les développements en interne.

En premier lieu, entre janvier et décembre 2005, une solution à base de script Perl a été mise en place. Celle-ci, à base de fichiers plats, permettait de gérer un simple référentiel de serveurs. Des cartographies de supervision Patrol étaient générées automatiquement.

Dès cette période, une nouvelle solution plus industrielle était en cours d'étude. D'autre part, l'arrivée d'autres produits d'hypervision[2] (BEM, PEM) nous ont poussé à adapter l'environnement associé et à faire évoluer la version initiale. C'est donc cette version de l'outil Spatrol que je vais vous présenter dans ce document. Elle a pour but de centraliser toutes les données permettant de connecter les agents aux infrastructures de supervision. Elle permet aussi d'enrichir les alarmes émises par les agents afin d'optimiser la diffusion à chaque équipe de soutien concernée et d'améliorer le traitement de celles-ci en fournissant des données fiables. Cet outil permet d'obtenir rapidement des rapports variés sur le périmètre de supervision, le bilan des licences utilisées ainsi que des rapports sur les versions des composants installés.

Ce projet s'étend sur la période de janvier 2005 à décembre 2007.

Ressources requises

[2] Consolidation d'évènements et d'enrichissement d'alarmes

Pour héberger notre application, nous avons du nous mettre en quête d'un serveur disponible. Notre choix s'est porté vers une ancienne console qui servait fin 2004 pour l'une des USEI. Grâce aux experts système du groupe, cette machine a été réinstallée aux normes en vigueur. Il s'agissait d 'un serveur Sun Netra 1405 avec Solaris 8 en version Platon G4R0. Les produits d'infrastructure Tina et Patrol (V3.4.11) ont d'ailleurs été ajoutés. La version actuelle a finalement été livrée en juin 2005. Il ne restait plus qu'à installer les produits liés à la solution.

Risques

La phase la plus critique dans la mise la mise en place de l'application a été la consolidation initiale des données des 8 consoles de supervision. Il ne fallait perdre aucune donnée et éviter toute discontinuité de service pour les équipes de supervision. La solution devait être disponible 24 heures sur 24.

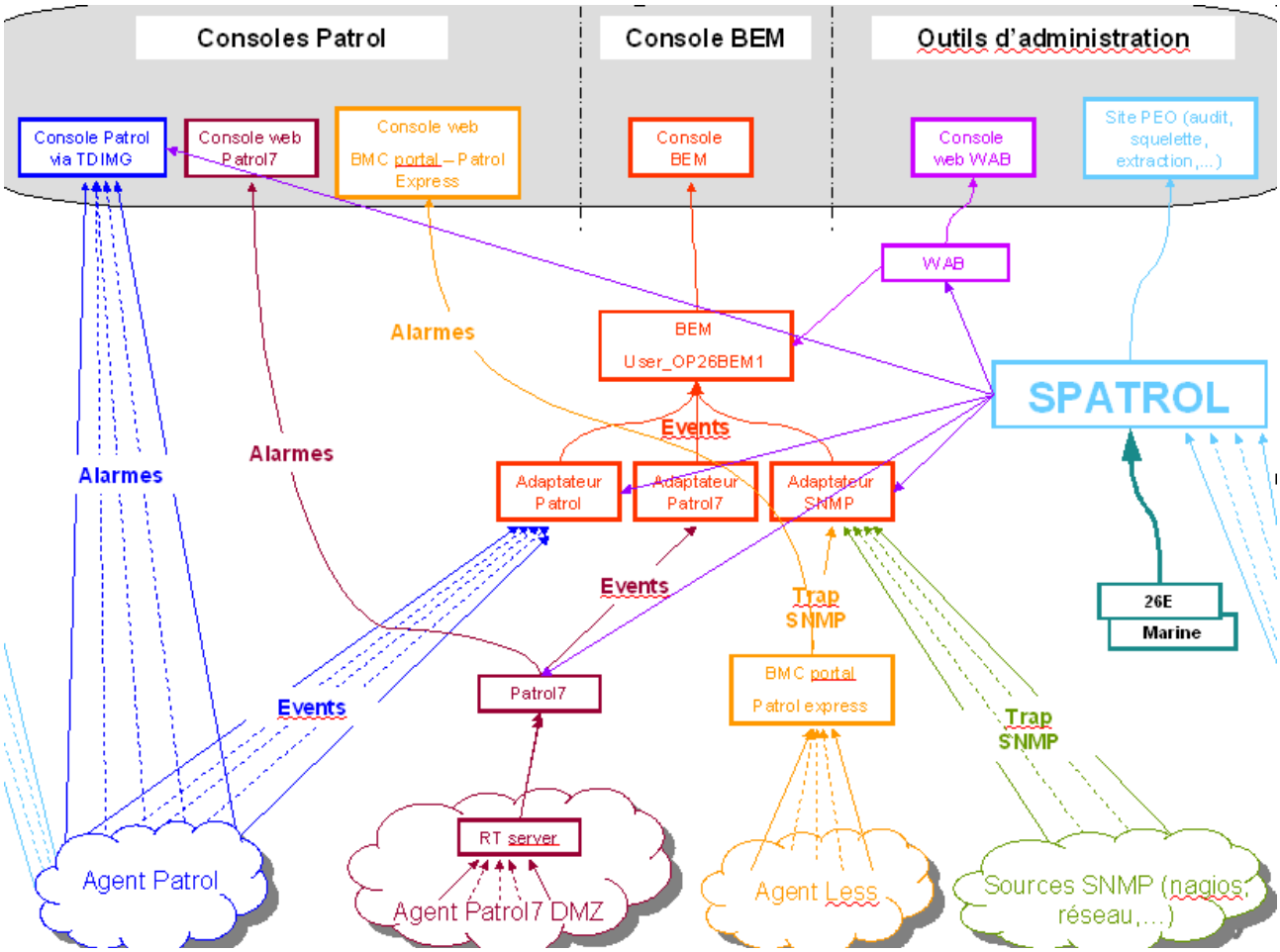
D'autre part, les données stockées sur le serveur ont un degré de confidentialité importante. Il recense en effet les adresses de tous les serveurs du SI, l'application à laquelle chaque serveur est dédié ainsi que la criticité de celle-ci.

Enfin, dès janvier 2005, l'application était en production, mais, faute de disposer d'un serveur de développement, les améliorations devaient être directement réalisées sur le serveur de production.

Contexte technique

Position dans le SI

Le schéma suivant présente une vision globale des applications liées à la supervision et la position de Spatrol par rapport à ces outils. Ces relations étant détaillées ci-dessous.



Spatrol est un des éléments clefs de la chaîne de supervision. C'est l'outil qui permet de gérer le référentiel des équipements supervisés. Il se connecte à tous les serveurs supervisés en utilisant le protocole de supervision adéquat. Il collecte un état des lieux des serveurs régulièrement et le met à disposition via une interface Web.

Il se synchronise aussi avec divers outils de la chaîne de supervision. Avec ce procédé, nous disposons des données fiables et à jour sur l'ensemble des applications utiles à la supervision. Il collecte des informations dans 26E qui est le référentiel de tout équipement du système d'information de France Télécom. Cette application référence aussi les applications et donc l'usage de chaque serveur. Elle contient aussi la criticité de celles-ci (Or, Argent, Bronze). Toutes ces informations sont recueillies par Spatrol et permettent d'enrichir les alarmes reçues par BEM[3] (remplaçant de PEM). D'autres informations sont récupérées dans l'application Marine qui centralise les données de la chaîne de soutien pour toutes les applications de France Télécom. Ces synchronisations sont automatiques et journalières.

[3] BEM : hyperviseur de BMC Software décrit page 18.

Un référentiel de supervision

Outre le référentiel simple des données permettant de se connecter aux agents, l'objectif de Spatrol est de fournir un état des lieux des serveurs. Nous avons donc écrit des scripts permettant de les interroger, en se basant sur les langages adaptés à chaque type d'agent. Nous avons de cette manière pu collecter à distance l'OS, la version, la date du dernier redémarrage et beaucoup d'autres paramètres. Certains scripts donnant énormément d'informations, mais n'étant pas collectés automatiquement peuvent être lancés simplement à la demande depuis l'interface Web.

Spatrol est capable de référencer et d'auditer les équipements suivants :

- Les agents Patrol. Ceux-ci peuvent communiquer soit directement s'ils sont connectés au RIFT (49 % du parc), soit en passant par un proxy applicatif appelé dans ce cas un RT serveur (Architecture Patrol7). Le protocole utilisé dans les deux cas est un protocole propriétaire BMC, que l'on peut utiliser via des exécutables fournis avec les produits,
- BMC portal qui est sans agent. Il s'agit de serveurs collectant à distance des informations sur d'autres serveurs en utilisant des protocoles standards. Cette solution est fortement utilisée pour les plate-formes de service qui sont bien souvent des plate-formes propriétaires sur lesquelles il est impossible d'installer un produit du marché,
- Nagios (produit du monde libre). Cet outil est utilisé pour surveiller certains serveurs Windows, principalement des frontaux de type « Metaframe ». Il permet de réaliser une supervision très normalisée basée sur le protocole WMI. Cette solution est fortement mise en place sur les applications comportant un nombre très important de serveurs Windows (e-buro light, Ebu, Métappli, ...),
- Les équipements réseaux supervisés via les « traps » SNMP qu'ils émettent,
- PEX (Patrol Express). Il s'agit d'une supervision sans agent et qui permet, via des requêtes sur des protocoles standards, de connaître l'état des serveurs.

Réalisations

Consolidation Serveur

Reprise de l'historique

Lors de la consolidation des consoles (qui a été l'évènement déclencheur du projet SPatrol), il n'existait que les consoles Patrol dans le système de supervision. Celles-ci étaient identiques et gérées par les USEI. Le SNPI (dont je faisais partie) en exploitait également une. D'une part, nous avons du réaliser un inventaire des serveurs connectés à ces consoles (A cette époque, il y avait 5500 agents Patrol, répartis sur 8 consoles); D'autre part, nous avons du récupérer les binaires d'exécution des consoles Patrol ainsi que les modules de connaissances (KM) permettant de visualiser l'état des serveurs. Nous avons donc du créer un nouveau serveur reprenant l'ensemble des Kms des différentes consoles en fonction des versions d'agents existantes.

Migration des données

Nous avons identifié les données pertinentes concernant ces agents. Le fichier /etc/hosts de chaque serveur contenant la liste des adresses IP des agents constituait la principale source d'informations. Nous avons ensuite du analyser les cartographies Patrol qui contenaient les comptes, mots de passe et ports de fonctionnement des agents. Les mots de passe étant cryptés dans les cartographies, nous les avons décrypté par dichotomie. Toutes ces données ont ensuite été intégrées à Spatrol pour être gérées de façon centralisée.

Données externes importées

26E

L'application 26E était au départ une gestion de configuration. A l'usage, ce logiciel a évolué d'un outil de recensement de matériel, à un référentiel s'apparentant à une CMDB de la méthode ITIL. Un référentiel de tous les équipements : postes de travail, imprimantes, serveurs, applications, châssis, équipements réseau, cluster, ...

Cette application permet d'assurer la gestion de parc, de l'entrée en stock des nouvelles commandes jusqu'à la sortie du parc des biens désinvestis en passant par les mouvements lors de leur exploitation.

Elle couvre 2 périmètres techniques :

- La bureautique (postes de travail, moyens d'impressions, composants logiciels, écrans...) gérée par l'entité DTF/DISU.
- Les applications IT&PFS (serveurs, partitions, conteneurs, stockage, éléments réseau, robotique, composants logiciels, applications,...) gérées par les entités DTF/DESI, DTF/DEPFS, DTF/DEI, ROSI/DPS.

Cette application offre aux utilisateurs une visibilité d'ensemble des données patrimoniales des biens du SI. Elle donne accès aux différents acteurs (Soutien, Déploiement, Gestion, Maintenance, Production, Exploitation…) du SI aux données communes de tous ces équipements. Elle permet en outre aux applications d'avoir à disposition, en tant que référentiel, des données techniques et organisationnelles fiables relatives aux biens du SI.
Enfin, elle fournit à travers son info-centre 26EBO, les tableaux de bord et des indicateurs de suivi du parc et référence les contrats de maintenance ainsi que la gestion des SLA.

26E fournit donc toutes les données dont elle est la référence concernant les équipements supervisés, et notamment les informations suivantes :

Concernant un serveur/partition :

Hostname	
IP	
Type	(Partition, Cluster, Réseau, Stockage)
État	(Dans le parc, Sortie du parc, En stock, ...)
Usage	(Production, Backup, Test, ...)
Site	
Bâtiment	
Étage	
Pièce	
Emplacement	
Travée	
OS	
Nom de l'application	

Concernant une application :

Nom	
DPSI	(Entité Exploitante)
DPSE	(Entité Support)
Top SOX	(Application impacté par la loi SOX)
Top QS	(Application demandant une Qualité de service optimum)
Code Basicat	(Code unique de l'application)
Eds	(Groupe de soutien)
Niveau de service	(Permet d'identifier la criticité de l'application qui peut être Or, Argent, Bronze).
Statut	(Production, J4[4], Développement)
Astreinte	(Application dont le niveau de service implique une astreinte)
Équipe Astreinte	(Équipe qui a la charge d'astreinte sur l'application)

Toutes ces données sont récupérées quotidiennement et mises à jour dans Spatrol. Elles permettront d'enrichir les données issues des alarmes de supervision remontées par les agents.

[4] J4: Il s'agit du jalon 4. Cela veut dire que l'application est sortie de production.

Marine (SAV2000)

Cette application est le référentiel France Télécom des équipes de soutien. On y trouve l'ensemble des groupes de soutien tels qu'ils sont utilisables dans Océane (Outil de traitement d'incident pour le groupe France Télécom). Ces données permettent en corrélation avec celles de 26E de définir la chaine de soutien.

Données exportées

Un changement de hostname, d'adresse IP, de compte dans Spatrol est automatiquement répercuté à toutes les applications qui lui sont directement liées. Voici le détail des applications dont une partie des données est issue de Spatrol.

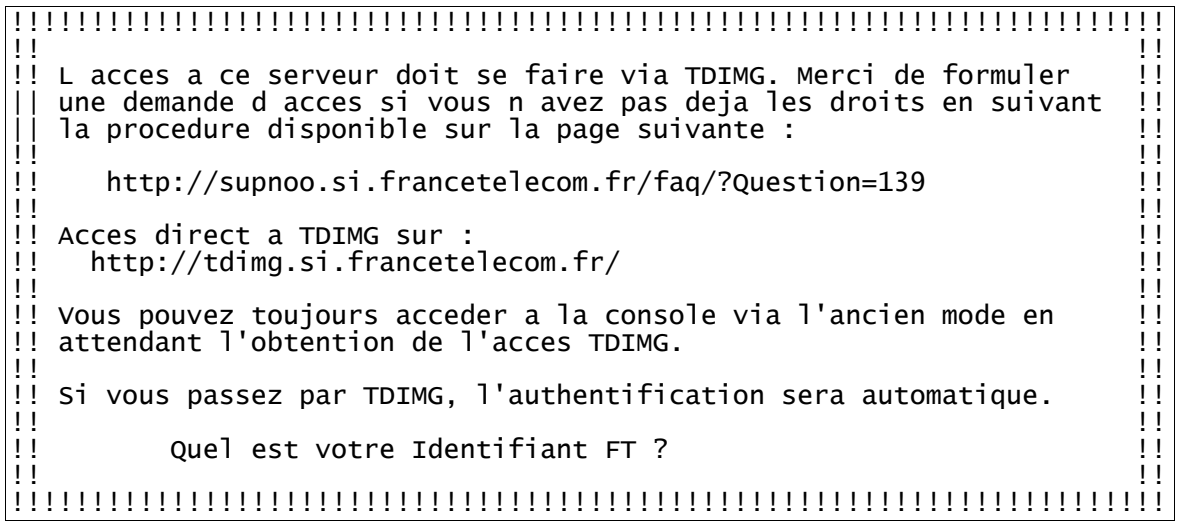
WAB

Le WAB (Web d'Administration de la BEM) permet, au niveau de la BEM, de définir la manière dont les alarmes doivent être enrichies. La majeure partie de la déclaration de cet enrichissement est faite de manière automatique sur la base des informations recueillies par Spatrol. Il est donc pour une grande partie réalisé automatiquement en fonction des sources de Spatrol, c'est à dire 26E et Marine. Une première mise à niveau est effectuée depuis Spatrol. En effet, les agents émettent des alertes sur trois niveaux de criticité alors que la BEM en gère cinq. Notons que des réglages précis au niveau des paramètres supervisés peuvent-être effectués par les équipes d'exploitation au niveau de Spatrol.

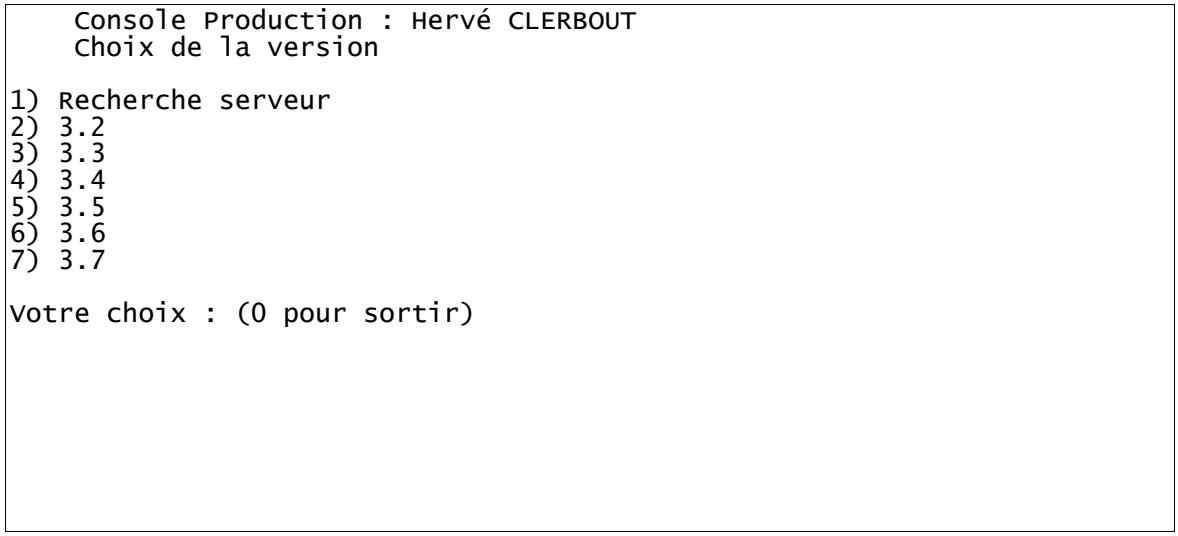
Consoles Patrol

Les consoles Patrol (Production et Backup) sont mises à jour depuis Spatrol. Les fichiers /etc/hosts de ces 2 serveurs sont automatiquement récupérés depuis les données contenues dans la base de données. Cette synchronisation permet aux équipes d'exploitation de gérer au quotidien leurs serveurs. Spatrol permet aussi de mettre à jour les cartographies de supervision. Chacune d'elle est représentée par un répertoire partagé. Chaque répertoire représente un container au sens Patrol qui lui même, peut, contenir des serveurs et des sous-containers. Lors de la génération des cartographies, les agents sont automatiquement répartis dans celle qui correspond à leur version. En effet, des agents de versions différentes ne peuvent cohabiter dans une même cartographie. Spatrol permet aussi de gérer l'authentification et le suivi des accès aux cartographies. Toutes les minutes, un script recherche les cartographies qui ont été mises à jour, les régénère et les réplique sur les 2 consoles.

Lorsqu'un utilisateur accéde à la console, il doit passer par la page d'accueil suivante lui demandant une authentification sur le domaine AD.



Une fois authentifié, il accède à un menu lui permettant de rechercher un serveur, un nom de container (souvent identique à l'application) ou d'ouvrir une cartographie. Ces cartographies sont organisées en fonction de la version de l'agent.



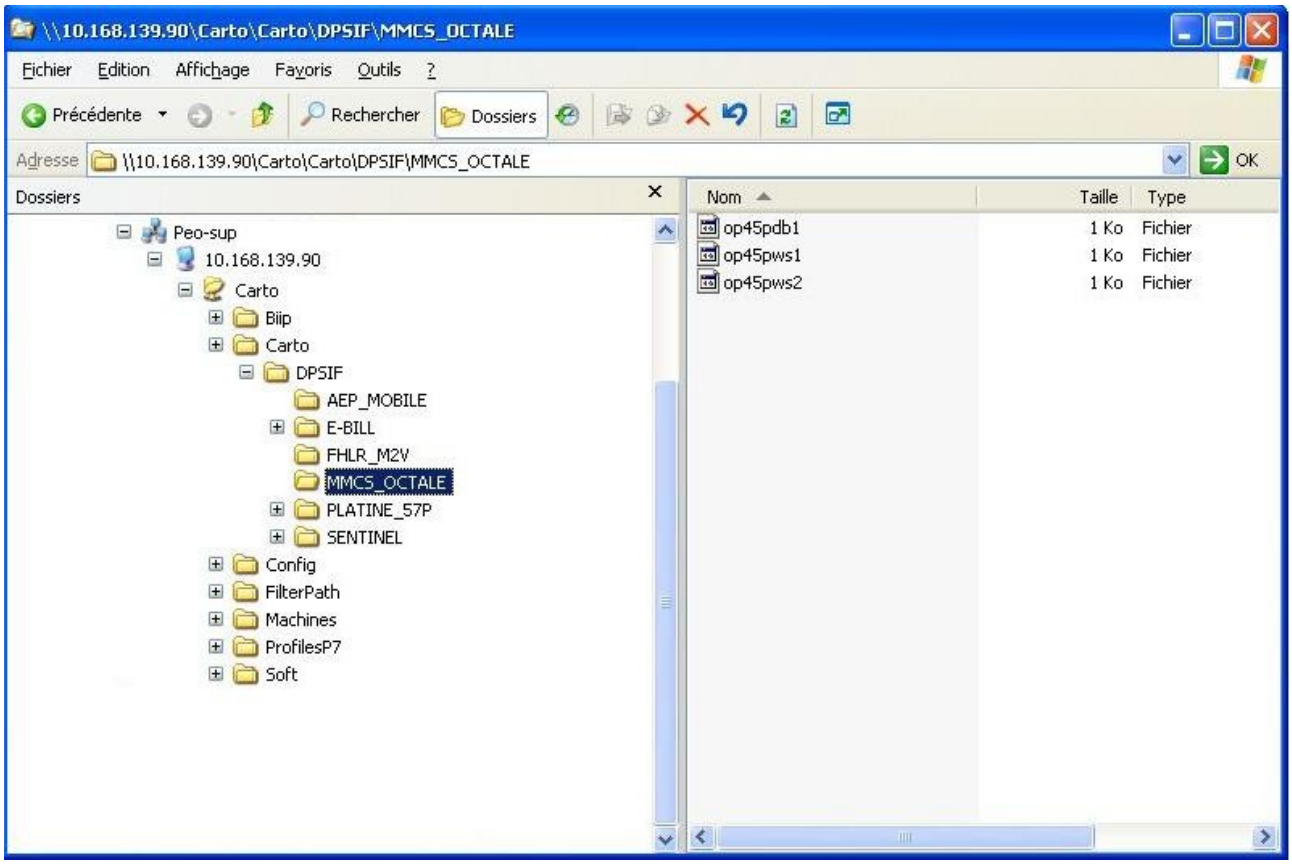
Voici par exemple la liste des cartographies Patrol en version 3.6.

```
    Console Production : Hervé CLERBOUT
    Choix de la carto en Version 3.6.

 1) Recherche serveur  26) DEM_APM             51) DPSII_DIA
 2) DEFI_CLA           27) DEM_FO              52) DPSII_IBS
 3) DEFI_CLA1          28) DEM_PCAF            53) DPSIRR_DFG
 4) DEFI_CLA2          29) DEM_TVX             54) DPSIRR_DFG_TVX
 5) DEFI_CLA_TVX       30) DEM_UNPS_PERIDISE   55) DPSIRR_DIART
 6) DEFI_CLB_UNIX      31) DERR_AUTRE          56) DPSIRR_DMSR
 7) DEFI_CLC           32) DERR_DFG            57) DPSIRR_DMSR_TVX
 8) DEFI_CLC_TVX       33) DERR_NT             58) DPSIRR_DSII
 9) DEFI_DEC           34) DERR_RES1           59) DST
10) DEFI_DEC_TVX       35) DERR_RES1_TVX       60) DSU_PEO
11) DEFI_ECRMDWH_TVX   36) DERR_RES2           61) OrangeMobile
12) DEFI_JAUNES_TVX    37) DERR_RESF           62) OrangeMobilePEM
13) DEFI_PASENANTES    38) DERR_RESF_IOSM      63) PEX1_RSX
14) DEFI_RP            39) DISU_PEO_QOS        64) PEX2_ESD3
15) DEFI_RP_TVX        40) DPSIC_BIDWH         65) REFSAM
16) DEFI_SON           41) DPSIC_BIDWH_TVX     66) URS_Lyon_AASR
17) DEH_INTRANET_TVX   42) DPSIC_CL_ESD3       67) URS_Lyon_MPE
18) DEH_OSM_U          43) DPSIC_CL_EXD2       68) WANA_BSS
19) DEH_OSM_W          44) DPSIC_COOR_TRANS    69) ZZZ_PARSIFAL
20) DEH_SUP_NANTES     45) DPSIC_CRM           70) xx34GRD
21) DEI_BEGOR          46) DPSIC_CRM_TVX       71) zzz_PEO_TEST
22) DEI_IASIL_IBU      47) DPSIF               72) zzz_PEO_UTIL
23) DEI_NETBACKUP      48) DPSIF_FE            73) zzz_Test
24) DEI_PINF           49) DPSIF_SII
25) DEI_PUBLIPHONIE    50) DPSIF_TVX

Votre choix : (0 pour sortir)
```

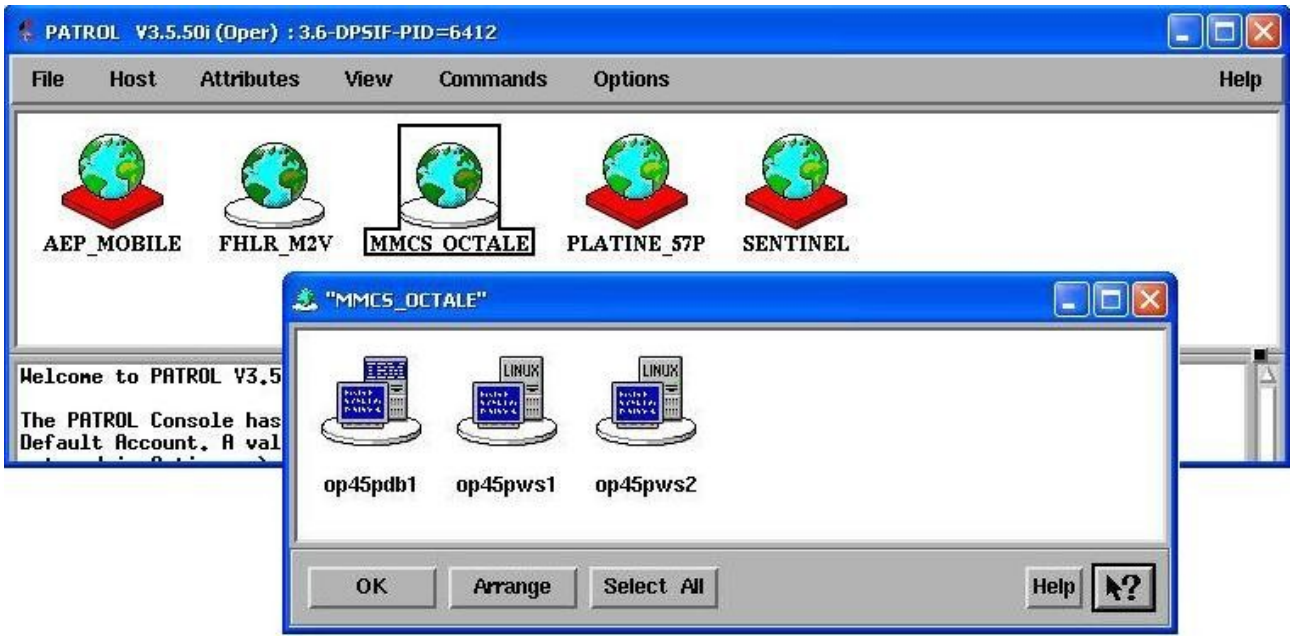
Voici une copie d'écran suite à une recherche de machine contenant « h3xx ».

```
    Console Production : Hervé CLERBOUT
    Liste des Cartos contenant la machine "h3xx"

 1) 3.2 DEFI_CLB_UNIX h3xxosc1     23) 3.6 DERR_RES2 h3xxnm12
 2) 3.2 DEFI_CLB_UNIX h3xxadr2     24) 3.6 DERR_RES2 h3xxnms01
 3) 3.2 DEFI_CLB_UNIX h3xxadr4     25) 3.6 DERR_RES2 h3xxrmp1
 4) 3.2 DEI_BEGOR h3xxpat1         26) 3.6 DERR_RES2 h3xxrm01
 5) 3.2 DEI_BEGOR h3xxrea1         27) 3.6 DERR_RES2 h3xxnms03
 6) 3.2 DERR_RES2 h3xxrem4         28) 3.6 DERR_RES2 h3xxnms02
 7) 3.2 DERR_RES2 h3xxrem3         29) 3.6 DERR_RES2 h3xxrm02
 8) 3.2 DERR_RES2 h3xxrem1         30) 3.6 PEX1_RSX H3XXOPS4
 9) 3.4 DERR_RES2 h3xxbrxl         31) 3.6 PEX1_RSX h3xxops6
10) 3.4 PEX1_RSX h3xxnav1          32) 3.6 PEX1_RSX H3XXOPS11
11) 3.5 DEFI_CLA1 h3xxc1p8         33) 3.6 PEX1_RSX h3xxops2
12) 3.5 DEFI_CLA1 h3xxc1p7         34) 3.6 PEX1_RSX h3xxops1
13) 3.5 DEFI_CLA1 h3xxc1i1         35) 3.6 PEX1_RSX h3xxops8
14) 3.5 DEFI_CLA1 h3xxc1p6         36) 3.6 PEX1_RSX H3XXOPS5
15) 3.5 DEH_SUP_NANTES H3XXAPS2    37) 3.6 PEX1_RSX h3xxaws5
16) 3.5 DEI_TVX h3xxout2           38) 3.6 PEX1_RSX h3xxops7
17) 3.5 PEX2_ESD3 h3xxc1p8         39) 3.6 PEX1_RSX h3xxops3
18) 3.5 PEX2_ESD3 h3xxc1p7         40) 3.6 PEX1_RSX h3xxaws4
19) 3.5 PEX2_ESD3 h3xxc1p6         41) 3.6 PEX1_RSX h3xxaws1
20) 3.5 PEX2_ESD3 h3xxc1i1         42) 3.6 PEX1_RSX H3XXOPS9
21) 3.6 DEI_PINF h3xxlog2          43) 3.6 PEX1_RSX h3xxops10
22) 3.6 DEI_PINF h3xxlog1          44) 3.6 PEX1_RSX h3xxaws6

Votre choix : (0 pour sortir)
```

Voici la représentation via partage netbios d'une cartographie.

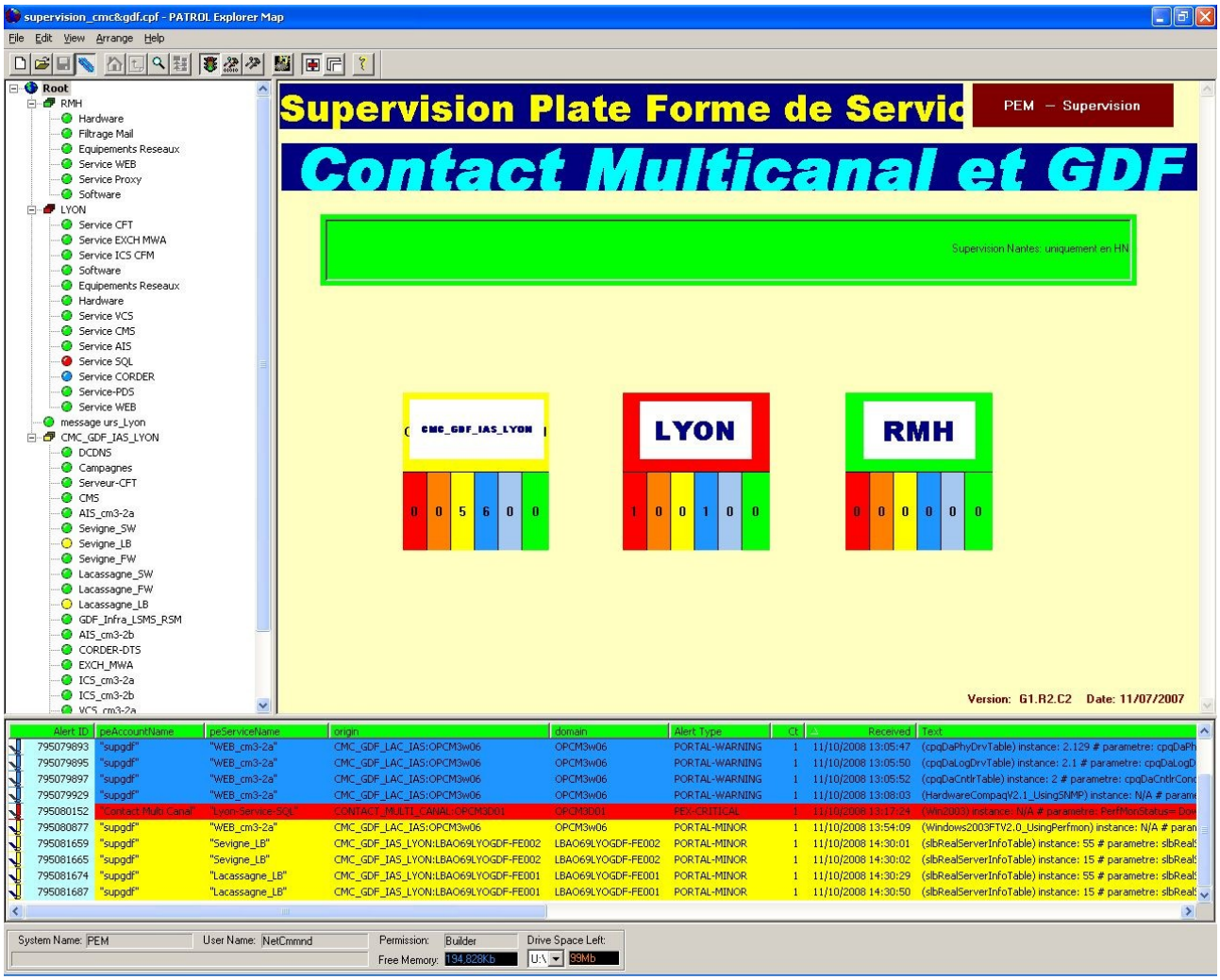


Voici la version Console de cette même cartographie.



PEM

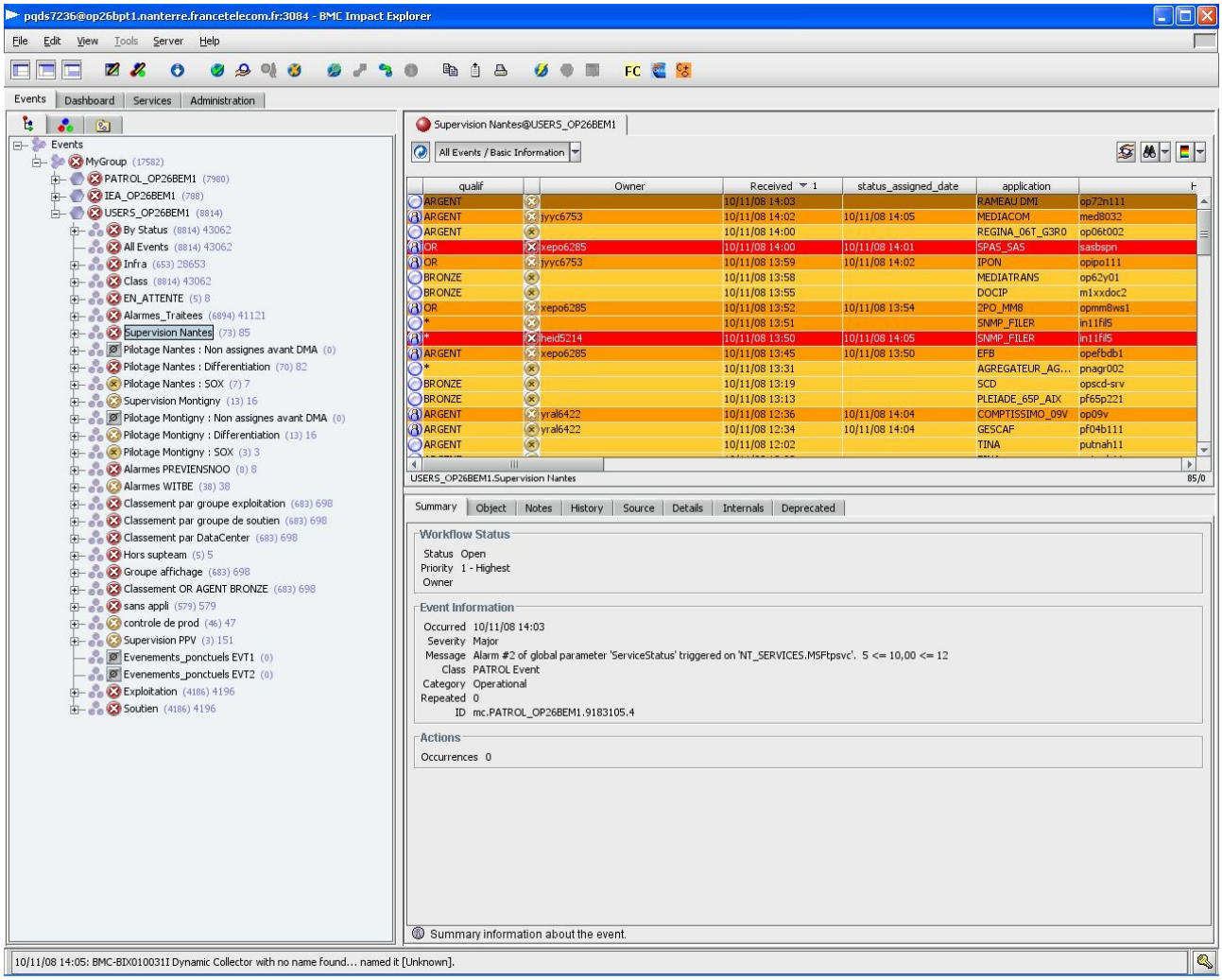
La PEM est un gestionnaire d'évènements. C'est le logiciel qui était chargé de consolider, d'enrichir tous les évènements issus des équipements supervisés par la DPSE. Il a été remplacé début 2008 par la BEM. Il s'appuyait sur une base de donnée Sybase pour ses données internes et Oracle pour le stockage des alertes[5] et les données d'enrichissement. Les agents sont connectés à cet outil via des FilterPath. Ils sont automatiquement maintenus par Spatrol. Cette gestion nécessite des interactions avancées entre Spatrol et la PEM afin de redémarrer le cas échéant les FilterPath.



[5] Une alerte est un événement enrichi.

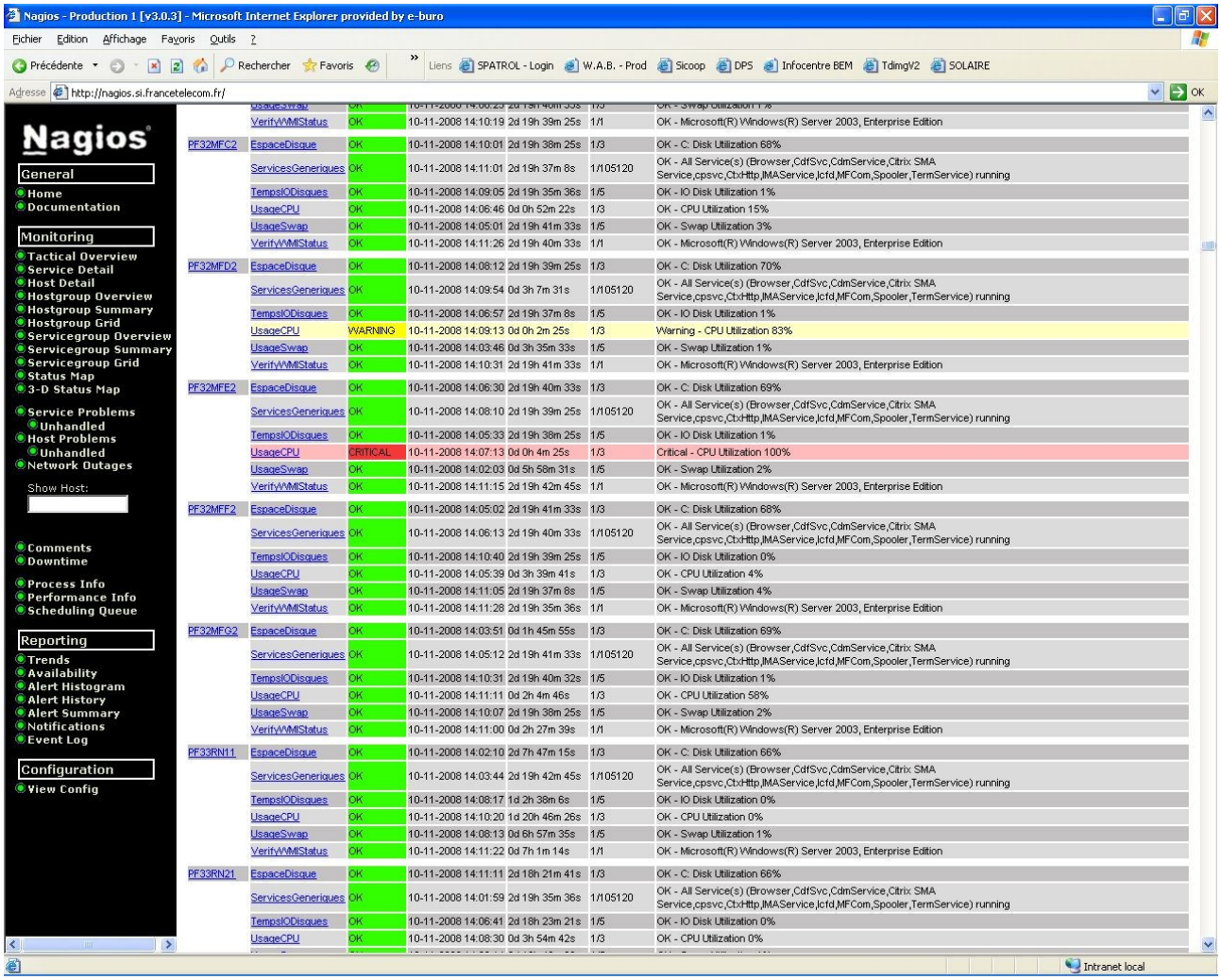
BEM

La BEM est un gestionnaire d'évènements. Ce logiciel a le même fonctionnement que la PEM, mais il a l'avantage d'être beaucoup plus distribué. En effet, en fonction des moyens disponibles en terme de nombre de serveurs, il est tout à fait possible de répartir l'application sur plusieurs serveurs, voire de dupliquer les serveurs afin d'en avoir toujours un de disponible. Les agents sont connectés à cet outil via des adapters Biip. Il existe un modèle d'adapter par type d'agent. Les fichiers de paramétrage sont automatiquement générés en fonction des affectations aux adapters définies dans Spatrol.



Nagios

Nagios est utilisé dans le cadre de la supervision pour collecter les informations sur les serveurs. Il est configuré automatiquement depuis Spatrol. C'est à ce niveau que l'on configure les services que l'on souhaite superviser. Cette supervision est très normalisée afin de limiter et de faciliter la configuration. Cette solution n'est déployée que sur quelques applications comportant un nombre conséquent de serveurs identiques.



SICOOP

SICOOP est une application Intranet permettant de maintenir une base de données contenant les mots de passe de comptes des serveurs en production des différentes entités de FT/TGPF/OPF/DTF/DESI. Les comptes et mots de passe contenus dans Spatrol sont extraits, cryptés et transmis à SICOOP. Cet échange est fait de manière automatique à l'aide de Pgp pour le cryptage et de l'envoi de mail aux experts SICOOP.

Serveur

Logiciels utilisés

Nous avions besoin de choisir les logiciels correspondants aux différentes parties de l'application.

Pour la partie présentation, il nous est apparu pertinent d'utiliser un serveur Web pour la simplicité de déploiement. Nous avons donc opté pour l'un des plus répandu, Apache. Ce moteur Web a été

complété par Php pour la couche présentation.

Tous les développements non interactifs ont été réalisés en Perl, pour sa modularité, sa simplicité d'interfaçage avec d'autres langages et ses performances.

Pour stocker les informations utiles à la supervision, nous avons choisi d'utiliser le système de gestion de base de données Mysql. Ce choix a d'abord été orienté par le coût de la solution (gratuite). D'autre part, Mysql est multi-plate-forme. Nous avons donc pu réaliser nos tests et modélisations sur nos PC puis passer sur le serveur de production sans aucune contrainte. Enfin, Mysql est supporté par Perl et Php.
En outre, Samba a été installé pour la gestion des cartographies et des adapters BEM. Ce mode de partage a été préféré à NFS car la connexion à nos environnements de bureautique est très simple.

Mysql

L'une des tâches majeures a été la conception de la base de données (cf annexe page 30-31). A l'époque, nous avions le choix entre la version 4 ou 5. La version 5.0.15 a été retenue car elle permet de mettre en place les procédures stockées, la gestion des curseurs et les triggers (cf annexe page 32-54). Le modèle joint en annexe a demandé beaucoup d'échanges entre les experts de supervision. Il a permis de modéliser toutes les données qui sont nécessaires à la supervision.

PSL

Le PSL est un genre de script permettant de faire exécuter des commandes ou des appels système aux agents Patrol. C'est grâce à ce langage que nous avons mis au point un ensemble de scripts permettant d'auditer les serveurs. Nous avons réussi à recueillir un inventaire détaillé de chaque serveur permettant ainsi à d'autres services d'accéder à une vue globale du parc à jour. Nous avons également consolidé les données concernant le nombre d'agents et de licences utilisées sur notre parc. Enfin, ces audits ont permis d'analyser de façon plus approfondie les critères supervisés ainsi que les problèmes de paramétrages des modules de connaissances (KM). Il a même été possible d'effectuer des corrections automatiques de paramétrage via ces commandes.

Perl

Nous avons décidé d'utiliser la version 5.8.7 qui était la dernière version existante. Le Perl a été utilisé dés la première version de Spatrol. En effet, ce langage est adapté pour les scripts interactifs. Dans un second temps, nous avons développé tous les scripts batch dans ce même langage. Néanmoins, il est bien adapté pour les interactions avec les bases de données hétérogènes (Mysql, Oracle, Sybase). Ce langage profite aussi d'une communauté fournissant énormément de modules (cf annexe page 55). Perl a été fortement utilisé pour les interactions avec les programmes de BMC, les appels aux applications annexes tel que Pgp, les sauvegardes, ...

Php

Pour l'interface Graphique web, nous avons eu recours à Php qui est plus adapté que Perl. Ce langage a aussi l'avantage de proposer nativement une multitude de modules permettant d'interagir avec le système. La version 5.0.5 qui a été choisie a l'avantage de gérer nativement les fichiers XML. Ce format est utilisé pour les retours d'audits des agents. Nous avons à ce titre activé le module permettant à Php d'interagir avec un serveur AD.

Samba

Ce logiciel donne accès à un système de fichiers Unix depuis un environnement Windows. La version utilisée est la 3.0.10. Il a d'abord été mis en place pour faciliter les développements. Ensuite, ce logiciel a permis de mettre à disposition différents modules (mise à jour des cartographies, gestion des sauvegardes de configuration, ...).

SSH

Ce protocole a permis à l'aide de clefs privées/publiques de lancer des traitements sur les serveurs distants. Il est fortement utilisé pour les mises à jour critiques. En effet, la mise à jour par exemple d'un fichier /etc/hosts est initiée par Spatrol, mais est faite par le serveur impacté lui-même. Cette technique permet en cas de problème sur Spatrol de ne pas corrompre les données critiques de ces serveurs.

Pgp

Ce logiciel permet de crypter des fichiers à l'aide de clefs privées/publiques. Il a servi pour les échanges des mots de passe avec SICOOP. Lors du transfert des fichiers de mot de passe, ceux-ci sont cryptés puis transmis par mail.

Ajax

Ce terme désigne une « nouvelle » approche utilisant un ensemble de technologies existantes, dont: HTML, les feuilles de styles CSS, JavaScript, le modèle objet de document (DOM), XML, XSLT, et l'objet XMLHttpRequest. Lorsque ces technologies sont combinées dans le modèle AJAX, les applications Web sont capables de réaliser des mises à jour rapides et incrémentielles de l'interface utilisateur sans devoir recharger la page entière du navigateur. Les applications fonctionnent plus rapidement et sont plus réactives aux actions de l'utilisateur. Cette technique a permis de rendre l'interface beaucoup plus conviviale.

Exécutables fournis par BMC

L'ensemble des exécutables fournis avec les produits de BMC (Agent, Console, KM, ...) ont été analysés de façon approfondie pour en tirer le meilleur parti. Il a été possible grâce à eux d'interagir à distance avec les agents, les consoles, la BEM et les équipements de l'architecture Patrol 7.

Développements effectués

Authentification LDAP

Le premier développement a été de mettre en place une authentification sur l'interface Web. Cette authentification a donc été implémentée en Php en utilisant le module adapté pour une connexion sur le domaine AD. Grâce à cette solution, nous n'avons pas eu besoin d'implémenter une gestion des comptes et profiles utilisateurs puisque nous réutilisions la base des comptes/mots de passe du domaine France Télécom. La copie d'écran suivante montre la page d'authentification à l'application Spatrol.

Cette solution a permis par la suite d'ouvrir l'accès à d'autres sous-domaines du groupe. Il est actuellement possible d'accéder au serveur pour les domaines suivants :

Orange

Wanadoo

Equant

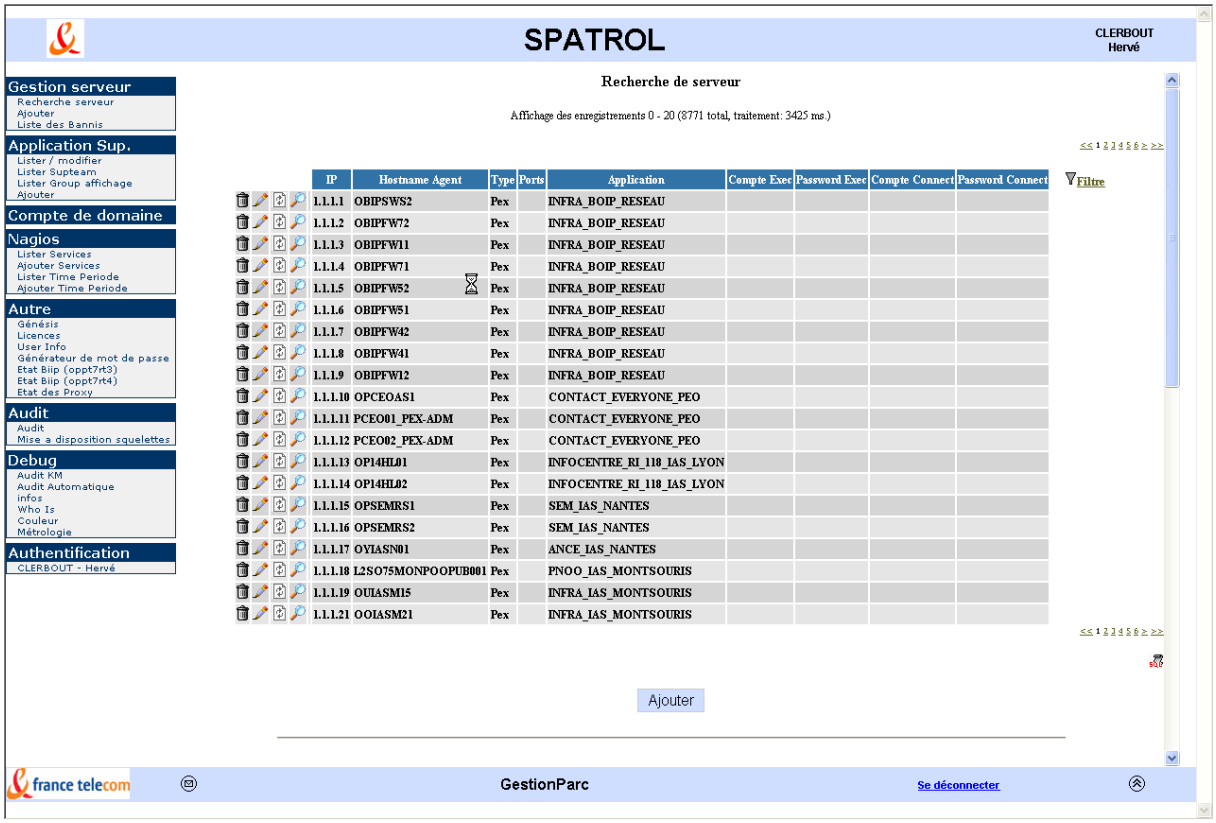
FT Marine

Sofrecom

FTT

Base des serveurs

L'interface de Spatrol permet aussi d'effectuer des recherches multi-critères sur l'ensemble des équipements qu'il contient. Cet accès permet à l'équipe qui gère tout l'environnement de supervision de disposer d'un point central de gestion et de mise à jour des informations de supervision. Lorsque nous recherchons un ou plusieurs équipements, nous bénéficions une vision globale des informations les concernant. Voici par exemple la recherche de tous les serveurs supervisés via PEX.



D'autre part, grâce aux audits effectués de manière automatique, nous pouvons à chaque instant connaître les problèmes d'accès aux serveurs. En effet, régulièrement, les serveurs sont audités et les données collectées sont stockées dans la base de données. Nous pouvons donc, sans nous connecter sur un serveur, connaître sa configuration (les KM chargés, leurs configurations respectives, ...).

Cette vision centrale est un gain de temps très important pour le PEO. Voici par exemple les caractéristiques du serveur "dr23gbo1" :

Detail du serveur dr23gbo1.

Type d'agent : Patrol
Hostname reel : dr23gbo1
IP principale : 10.102.40.21
Port : 3181
Application de supervision : GDP (212044).
Salle de supervision : Nantes.
Priorité : 1.
Compte d'execution : patroloc [redacted]
Compte de connection : AD\copatroldom [redacted]
OS information : Microsoft Windows Server 2003 (5.2 Service Pack 1).
Agent information : V3.6.50.13i (C:\PROGRA~1\BMCSOF~1\PATROL~1\).
Liste des IPs :

10.64.196.13
10.100.32.19
10.100.36.2
10.102.40.21

Liste des KMs :

AS_EVENTSPRING V2.7.00.01 (preload)
->
SW_SENTRY 8.4.02 (preload)
->

Carto :

PEX1_RSX -> GDP-PEX-TOULOUSE -> dr23gbo1
DPSIRR_DMSR -> GDP -> dr23gbo1

Date de dernier inventaire : 2008-10-19 01:16:03
Date de dernière erreur d'inventaire : 2008-11-07 10:25:42
BEM :

oppt7rt3 / 023

Liste des intervenants :

Pascal CHERRIER	2008-03-14 10:08:21	Ajout	Serveur	
Pascal CHERRIER	2008-03-14 10:08:21	Ajout	Agent	
Pascal CHERRIER	2008-03-14 10:09:02	Modification	Serveur	Resource 26E
Pascal CHERRIER	2008-03-14 10:09:02	Modification	Agent	Application
Pascal CHERRIER	2008-03-26 11:04:24	Modification	Serveur	Compte Domaine Execution Compte Domaine Connexion
Pascal CHERRIER	2008-03-26 11:18:44	Modification	Serveur	Compte Domaine Execution Compte Exec Password Exec
Pierrick ORGEBIN	2008-10-08 08:53:23	Modification	Serveur	IP

Actions diverses :
- Test AD\appl-parme
- Restart
- Stop
- Bannir

Information 26E serveur (n°0000452083) :
- Hostname : dr23gbo1
- Type : Partition
- Etat : Dans le parc
- Usage : Préproduction
- Site : /FRANCE/93/AUBERVILLIERS/AUBERVILLIERS LA MOTTE/
- Batiment : AAS
- Etage : ETG 3
- Piece : AAS32
- Os : Windows 2003 Server Enterprise

Application de la partition :
- GDP - 23G (23G).

Information 26E application (n°305484159) :
- Nom : GDP - 23G
- Entité Exploitante : /Ft/Tgpf/Opf/Dtf/Desi/Dprod/Dpse/Pex 1/Res/
- Entité Support : /Ft/Tgpf/Opf/Dtf/Desi/Dprod/Dpsirr/Sii/Msr 2/
- Top SOX
- Code Basicat : 23G
- Niveau de service : Argent
- Eds : AQZE14
- Code appli local :
- Ft Carto : 02164

Accès direct :
- Telnet
- Ssh

Mots de passe encyptés :

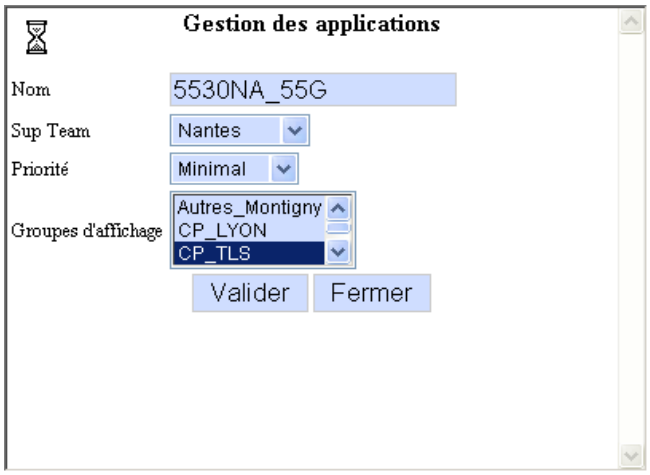
Passwd	Cryptage DES	Cryptage PEM
[redacted]	3CB736A49C12674A	A3CB736A49C12674A1
[redacted]	410548E5B034D66FDBD128D379E9A39C	A410548E5B034D66FDBD128D379E9A39C1

Gestion via le Web

L'interface Web permet de gérer l'ensemble des serveurs supervisés. Par exemple pour ajouter un nouveau serveur, il suffit de fournir l'adresse IP, le port de fonctionnement de l'agent et le couple compte/mot de passe. Ces informations sont validées par un test de connexion et automatiquement, toutes les caractéristiques sont récupérées. Des corrélations automatiques sont faites avec les données récupérées sur le serveur (hostname, version d'agent, OS, ...) et celles contenues dans les autres applications (26E, Marine, ...). Toutes ces corrélations permettent d'avoir une vue dans Spatrol et une supervision du serveur avec plus d'informations.

C'est également via cette interface que l'on déclare les applications de supervision avec leurs paramètres. Une application de supervision sert à définir des critères spécifiques pour un ensemble de serveurs. En effet, nous pouvons spécifier une salle de supervision. La supervision des applications du SI de France Télécom se fait depuis plusieurs sites. Nous pouvons aussi définir un degré de priorité de l'application qui permet, en plus du niveau de service, d'optimiser le traitement des alertes sur les serveurs. Il est aussi possible d'associer une application à un groupe d'affichage pour la gestion des cartographies BEM.

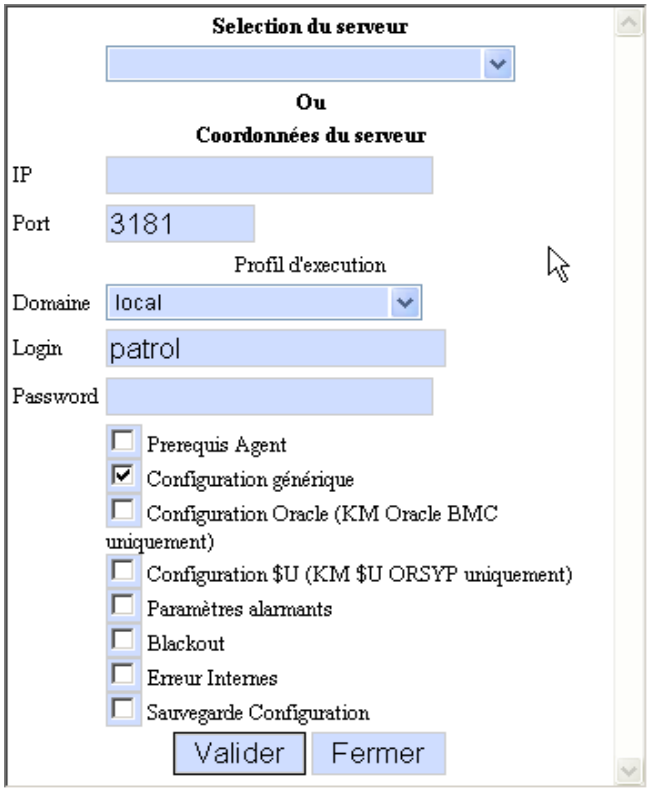
Voici l'écran de création d'une application de supervision.



Audit via le Web/Perl

Spatrol permet aussi de lancer des audits sur les serveurs. Ces audits étaient dans un premier temps lancés de manière synchrone, mais leur nombre ayant fortement augmenté, il a été décidé de les déclencher de manière asynchrone car leur durée d'exécution ne permet pas de les gérer de manière interactive. Les audits sont donc ajoutés à la liste d'attente et un démon va régulièrement lancer les demandes en suspens. Ces audits peuvent être exécutés sur les serveurs déjà connus de Spatrol (production, backup, ...) ou sur les serveurs d'intégration. Seuls les agents de type Patrol (qui représentent 49 % du parc) peuvent être actuellement audités.

Voici l'interface de soumission d'un audit.



Suite à ces audits, la personne qui a fait la demande recevra le compte rendu par mail. Dans ce compte rendu, il trouvera une analyse détaillée.

S'il demande un audit des pré requis, il reçoit toutes les erreurs de droits qui empêcheraient un KM

de fonctionner normalement sur le serveur.

S'il demande un audit de la configuration générique, il reçoit un état des lieux de tout ce qui est supervisé via des KM génériques (Système et Sentry).

S'il demande un audit de la configuration Oracle, il reçoit un état des lieux de ce qui est supervisé via le KM Oracle de BMC.

S'il demande un audit de la configuration DollarU, il reçoit un état des lieux de ce qui est supervisé via le KM DollarU de BMC.

S'il demande un audit des paramètres alarmants, il reçoit une liste de tous les paramètres Patrol sur lesquels des seuils ont été positionnés. Avec cet audit, l'utilisateur peux vérifier qu'il supervise ce qu'il souhaite.

S'il demande un audit de la configuration Blackout, il reçoit un bilan des différentes désactivations de supervision. Cet état permet de vérifier que les désactivations ne se superposent pas et que certains paramétrages problématiques ne sont pas utilisés.

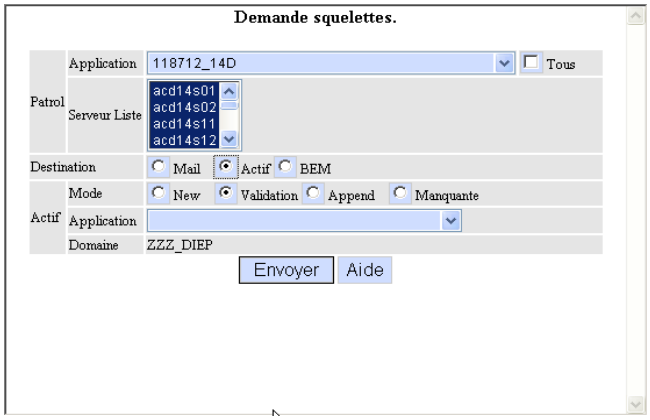
S'il demande un audit des erreurs internes, il reçoit une extraction de toutes les erreurs de l'agent Patrol. Cet audit permet aux experts Patrol d'améliorer et de simplifier leur analyse en cas de problème sur des Agents Patrol.

S'il demande une sauvegarde, il provoque une sauvegarde manuelle de la configuration de l'agent Patrol.

Un compte rendu d'audit est présenté en annexe (Page 57).

Demande de Squelette via le Web/Perl

Les demandes de squelettes permettent de connaître l'ensemble des triplets KM, instances, paramètres susceptibles de générer des alarmes. Un KM est un type d'objet (par exemple les filesystems). Une instance est un objet (par exemple C:). Enfin, un paramètre est un critère (par exemple le taux de remplissage ou le nombre de fichiers). Cet audit n'est pour l'instant compatible qu'avec les agents Patrol.



Sauvegarde des configurations

Ce module permet de générer une sauvegarde du paramétrage de chaque agent Patrol. Régulièrement, les agents sont audités et dans le cadre de cet audit, la configuration est susceptible d'être sauvegardée. A chaque audit, la configuration est épurée des données non pérennes et comparée à la précédente. Si elle diffère, elle est archivée. Cinq fichiers de configuration différents sont conservés. Pour restaurer un paramétrage, il suffit de se positionner sur le partage bureautique des configurations pour retrouver le fichier correct. Les sauvegardes sont nommées avec le nom du serveur, le port de l'agent et un numéro de version de sauvegarde.

Améliorations envisagées

Depuis que j'ai quitté ce projet, des améliorations ont été mises en place. C'est le cas de la remontée des triplets KM/Instance/Paramètre pour l'architecture Patrol7. Ces données servent à redéfinir le cas échéant les criticités des alertes reçues. En effet, les agents émettent des évènements sur trois niveaux de criticité alors que la BEM en gère cinq.

Certaines autres sont en cours d'élaboration ou d'étude. Une solution permettra rapidement de remonter les triplets KM/Instance/Paramètre pour l'architecture Portal et ce, de manière automatique. Ces triplets sont importants car ils permettent de définir des critères de traitement différents pour des problèmes sur un même serveur.

Enfin, des améliorations devront être apportées en 2009. Il va falloir automatiser les audits de squelettes pour ne plus avoir besoin de les déclencher à la demande. Il faudra aussi améliorer les audits de paramétrage en fonction des dysfonctionnements rencontrés.

L'amélioration qui est actuellement à l'étude devrait augmenter la fiabilité de l'application en l'installant sur un serveur virtualisé.

Enfin, la majorité des améliorations passées comme futures sont liées à l'évolution des logiciels en rapport avec la supervision. Spatrol étant extrêmement lié avec eux, une nouvelle version d'un des outils provoque immanquablement une mise à jour de Spatrol.

Bilan

DPSE étant une entité exploitante, les normes de supervision sont définies par la maîtrise d'œuvre supervision SAQS. Sur ce projet, nous avons été à la fois maîtrise d'œuvre et maîtrise d'ouvrage. Les décisions ont parfois dû être prises sans avoir la vision des nouvelles versions d'outils de supervision qui sont en cours d'études, ce qui a engendré parfois des modifications à postériori de certains modules. Mais néanmoins, cette situation a simplifié les prises de décision. Nous avons pu avancer dans la concrétisation du projet beaucoup plus rapidement.

Il reste à ce jour des améliorations à apporter, mais celles-ci sont majoritairement liées à l'évolution des produits de supervision. En outre, la version actuelle est en fonctionnement depuis maintenant 3 ans (cf annexe page 29) et n'a jamais été indisponible. Cela prouve que le choix d'architecture et les solutions techniques retenues ont été pertinents.

La réalisation de cette application m'a permis de mettre en pratique les connaissances techniques acquises durant les formations et mes deux premières années d'expérience au sein de DTO. J'ai aussi durant ces 3 années rencontré de nombreux experts et appréhendé de nouvelles technologies. Cette solution innovante est l'une des réalisations à base des logiciels de BMC les plus abouties à l'heure actuelle en France (cf annexe page 59).

Au moment d’achever ce rapport, je souhaite exprimer la grande fierté d’avoir été leader sur ce projet. Il a suscité chez moi un intérêt important et m’a permis d’appréhender de nombreuses problématiques techniques. Afin d'initier ce projet, j'ai du faire preuve de persuasion auprès de mes responsables afin de dégager le temps nécessaire me permettant de le mener à bien. Enfin, cette réalisation m'a permis d'améliorer mes méthodes de gestion du risque et de gestion de projet.

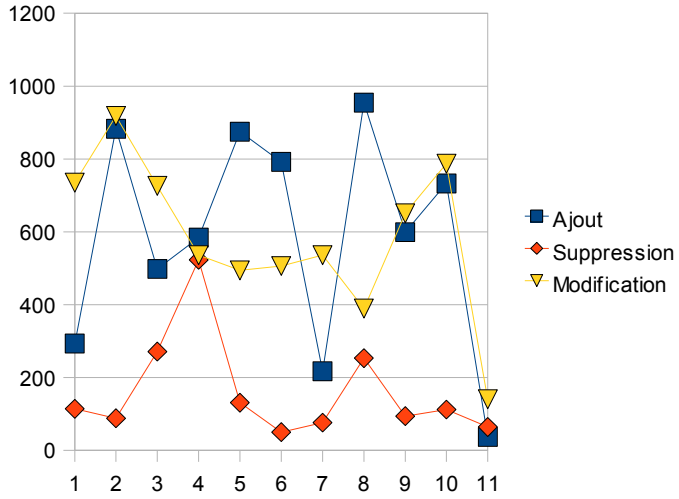
Annexes

Spatrol en chiffres

A l'initialisation du projet (13/12/2005), Spatrol référençait 5492 agents et 8329 comptes et mots de passe d'accès aux serveurs. Aujourd'hui, c'est 9981 agents dont 4927 supervisés via Patrol, 2100 via WMI. Ces supervisions sont réparties sur 1175 applications dont 259 sont considérées comme or et donc comme hypercritiques.

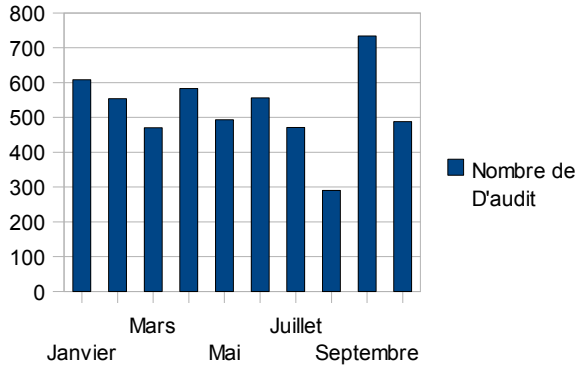
A ce jour, 1368 différentes personnes se sont connectées sur Spatrol ou à une des consoles. Il y a eu 101664 ouvertures de cartographies depuis le 1er janvier 2008.

Les personnes qui gèrent le référentiel (PEOS) ont effectué 18 507 ajouts, 6491 suppressions et 17039 mises à jour de serveurs, applications. Voici le nombre d'actions par type (modification, suppression, ajout) sur l'année 2008.



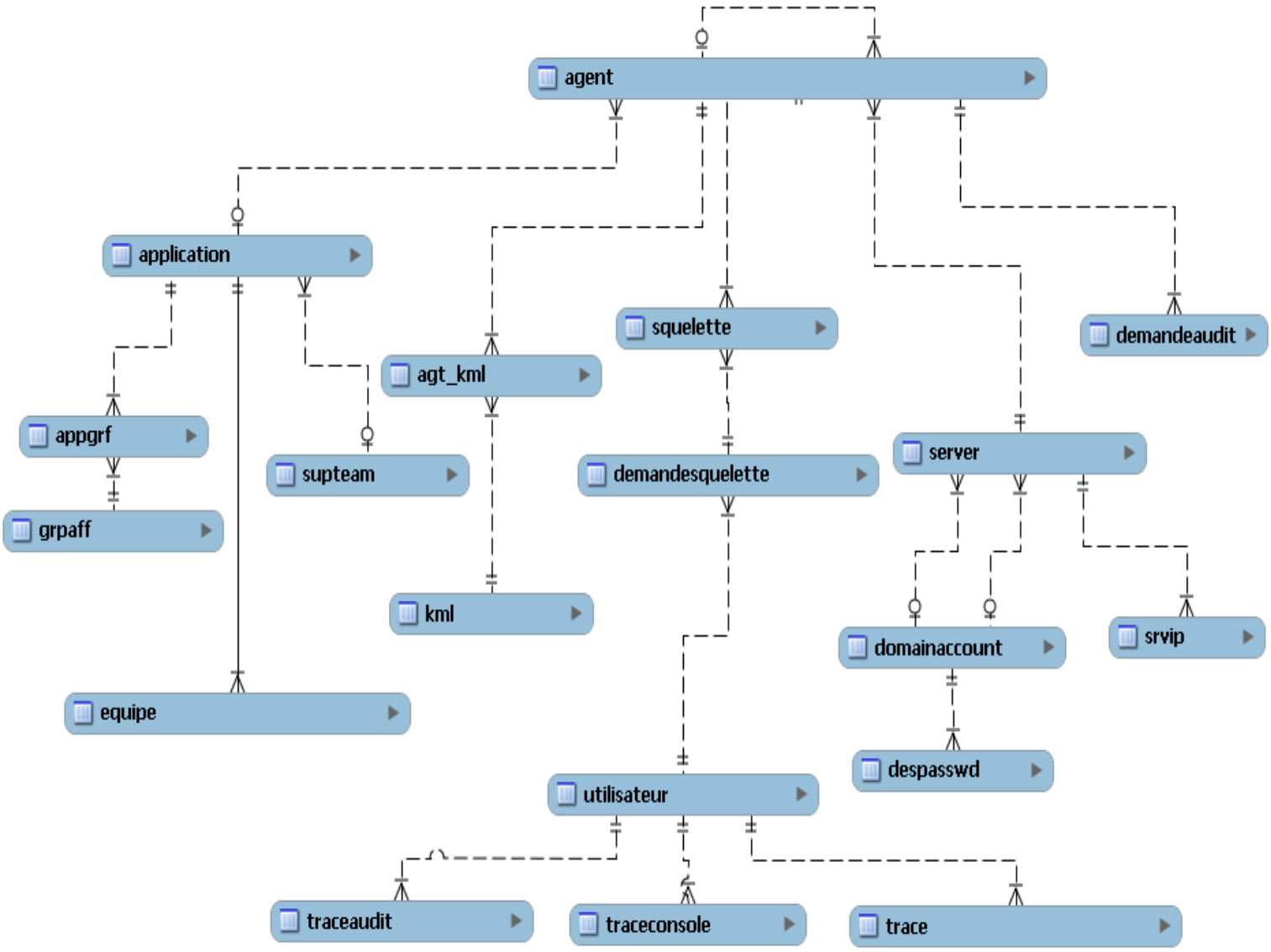
Spatrol sert aussi aux audits d'agents. Depuis le 17/07/2007, il y a eu 14773 audits effectués sur le parc. Il y a environ 200 demandes de squelettes par semaine.

Voici un historique du nombre d'audits par mois.

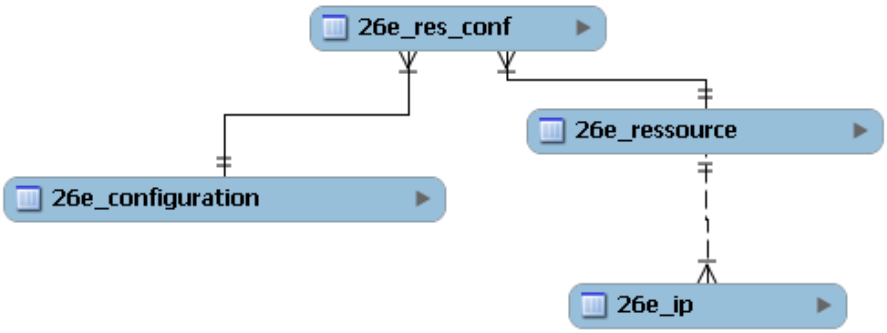


Modèle conceptuel de donnée

Données interne à SPatrol



Données collectées depuis 26E



Script de génération du schéma de base

```
--
-- Structure de la table `supteam`
--


CREATE TABLE `supteam` (
      `sup_id` int(1) unsigned NOT NULL auto_increment,
      `sup_name` varchar(255) BINARY NOT NULL COMMENT 'Nom',
      PRIMARY KEY (`sup_id`),
      UNIQUE KEY sup_name (`sup_name`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COMMENT='Liste des equipes de
supervision.';


-- --------------------------------------------------------


--
-- Structure de la table `grpaff`
--


CREATE TABLE `grpaff` (
      `grpaff_id` int(1) unsigned NOT NULL auto_increment,
      `grpaff_name` varchar(255) BINARY NOT NULL COMMENT 'Nom',
      PRIMARY KEY (`grpaff_id`),
      UNIQUE KEY grpaff_name (`grpaff_name`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COMMENT='Liste des groupes
d\'affichage.';


-- --------------------------------------------------------


--
-- Structure de la table `appgrf`
--


CREATE TABLE `appgrf` (
      `appgrf_grpaff_id` int(1) unsigned NOT NULL,
      `appgrf_app_id` int(1) unsigned NOT NULL,
      FOREIGN KEY (`appgrf_grpaff_id`) REFERENCES `grpaff` (`grpaff_id`),
      FOREIGN KEY (`appgrf_app_id`) REFERENCES `application` (`app_id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COMMENT='Liste des liaisons
applications/groupes d\'affichage.';


-- --------------------------------------------------------
```

```
--
-- Structure de la table `equipe`
--


CREATE TABLE `equipe` (
      `eqp_nom` varchar(255) BINARY NOT NULL COMMENT 'Nom',
      `eqp_detail` varchar(255) COMMENT 'Detail',
      `eqp_tool` ENUM('Genesis', 'Oceane', 'Telephone') NOT NULL default
'Genesis' COMMENT 'Outil associe',
      `eqp_horaire` varchar(255) default NULL COMMENT 'Plage Horaire de
service',
      `eqp_telephone` varchar(255) default NULL COMMENT 'Numero de telephone',
      PRIMARY KEY `eqp_nom` (`eqp_nom`, `eqp_tool`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COMMENT='Liste des equipes.';


-- --------------------------------------------------------
--
-- Structure de la table `timeperiod`
--


CREATE TABLE `timeperiod` (
      `tp_id` int(1) unsigned NOT NULL auto_increment,
      `tp_hostgroup` VARCHAR( 255 ) NOT NULL COMMENT 'Nom du host ou de
l\'application',
      `tp_alias` varchar(50) COMMENT 'Nom d\'usage',
      `tp_sunday` VARCHAR(255) NULL default '00:00-24:00' COMMENT 'Plage du
sunday',
      `tp_monday` VARCHAR(255) NULL default '00:00-24:00' COMMENT 'Plage du
monday',
      `tp_tuesday` VARCHAR(255) NULL default '00:00-24:00' COMMENT 'Plage du
tuesday',
      `tp_wednesday` VARCHAR(255) NULL default '00:00-24:00' COMMENT 'Plage du
wednesday',
      `tp_thursday` VARCHAR(255) NULL default '00:00-24:00' COMMENT 'Plage du
thursday',
      `tp_friday` VARCHAR(255) NULL default '00:00-24:00' COMMENT 'Plage du
friday',
      `tp_saturday` VARCHAR(255) NULL default '00:00-24:00' COMMENT 'Plage du
saturday',
      PRIMARY KEY (`tp_id`),
      UNIQUE KEY `tp_hostgroup` (`tp_hostgroup`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COMMENT='Liste des timeperiod.';


-- --------------------------------------------------------
--
-- Structure de la table `service`
--
```

```
CREATE TABLE IF NOT EXISTS `service` (
      `svc_id` int(1) unsigned NOT NULL auto_increment,
      `svc_type` ENUM('hostgroup', 'host') NOT NULL default 'hostgroup' COMMENT
'Type d\'association',
      `svc_obj_name` VARCHAR( 255 ) NOT NULL COMMENT 'Nom du host ou de
l\'application',
      `svc_command` VARCHAR( 255 ) NOT NULL COMMENT 'Commande Nagios',
      `svc_option` VARCHAR( 255 ) DEFAULT '0' NOT NULL COMMENT 'Options',
      `svc_max_check_attempts` INT(1) unsigned DEFAULT '1' NOT NULL COMMENT
'max_check_attempts',
      `svc_normal_check_interval` INT(1) unsigned DEFAULT '1' NOT NULL COMMENT
'normal_check_interval',
      `svc_retry_check_interval` INT(1) unsigned DEFAULT '1' NOT NULL COMMENT
'retry_check_interval',
      PRIMARY KEY (`svc_id`),
      UNIQUE KEY `svc` (`svc_type`, `svc_obj_name`, `svc_command`),
      INDEX ( `svc_type` ),
      INDEX ( `svc_obj_name` ),
      INDEX ( `svc_command` )
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COMMENT='Liste des services' ;

-- --------------------------------------------------------

--
-- Structure de la table `km`
--

CREATE TABLE `km` (
      `km_id` int(1) unsigned NOT NULL auto_increment,
      `km_nom` varchar(50) NOT NULL COMMENT 'Nom',
      PRIMARY KEY (`km_id`),
      KEY `km_nom` (`km_nom`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COMMENT='Liste des KMs.';

-- --------------------------------------------------------

--
-- Structure de la table `param`
--

CREATE TABLE `param` (
      `par_id` int(1) unsigned NOT NULL auto_increment,
      `par_km_id` int(1) unsigned NOT NULL COMMENT 'Km',
      `par_nom` varchar(50) NOT NULL COMMENT 'Nom',
      PRIMARY KEY (`par_id`),
```

```
      FOREIGN KEY (`par_km_id`) REFERENCES `km` (`km_id`),
      INDEX (`par_nom`),
      KEY `par_nom_km_id` (`par_km_id`, `par_nom`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COMMENT='Liste des parametres.';

-- --------------------------------------------------------
--
-- Structure de la table `banni`
--

CREATE TABLE `banni` (
      `ban_id` int(1) unsigned NOT NULL auto_increment,
      `ban_hostname` varchar(20) NOT NULL COMMENT 'Hostname',
      `ban_uid` char(8) NOT NULL COMMENT 'Ident FT',
      `ban_date` timestamp NOT NULL default CURRENT_TIMESTAMP on update
CURRENT_TIMESTAMP COMMENT 'Date d exclusion',
      `ban_comment` varchar(255) NOT NULL COMMENT 'Commentaire',
      PRIMARY KEY (`ban_id`),
      KEY `bckl_hostname` (`ban_hostname`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COMMENT='Liste des serveurs bannis.';

-- --------------------------------------------------------

--
-- Structure de la table `traceconsole`
--

CREATE TABLE `traceconsole` (
      `trc_cons_id` int(1) unsigned NOT NULL auto_increment,
      `trc_cons_server` varchar(20) NOT NULL COMMENT 'Loggin',
      `trc_cons_user` varchar(20) NOT NULL COMMENT 'Loggin',
      `trc_cons_uid` char(8) NOT NULL COMMENT 'Ident FT',
      `trc_cons_version` char(3) NOT NULL COMMENT 'Version Agent',
      `trc_cons_vue` varchar(20) NOT NULL COMMENT 'Carto Patrol',
      `trc_cons_date` timestamp NOT NULL default CURRENT_TIMESTAMP on update
CURRENT_TIMESTAMP COMMENT 'Date execusion',
      `trc_cons_display` varchar(21) NOT NULL COMMENT 'Display',
      `trc_cons_tdimg` int(1) NOT NULL default 0 COMMENT 'via tdimg',
      `trc_cons_pid` MEDIUMINT(1) unsigned NOT NULL COMMENT 'Pid',
      `trc_cons_action` varchar(10) NOT NULL COMMENT 'Action',
      PRIMARY KEY (`trc_cons_id`),
      KEY `trc_cons_tdimg` (`trc_cons_tdimg`),
      KEY `trc_cons_uid` (`trc_cons_uid`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COMMENT='Trace des acces consoles.';
```

```
-- --------------------------------------------------------

--
-- Structure de la table `utilisateur`
--

CREATE TABLE `utilisateur` (
      `uti_uid` varchar(8) NOT NULL COMMENT 'identFT',
      `uti_nom` varchar(255) NOT NULL COMMENT 'Nom',
      `uti_prenom` varchar(255) NOT NULL COMMENT 'Prenom',
      `uti_email` varchar(255) NOT NULL COMMENT 'Email',
      `uti_hostname` varchar(255) default NULL,
      `uti_nompresentation` varchar(255) NOT NULL COMMENT 'Nom d usage',
      `uti_bureau` varchar(20) default NULL COMMENT 'Bureau',
      `uti_telephone` varchar(25) default NULL COMMENT 'Telephone',
      `uti_fax` varchar(25) default NULL COMMENT 'Fax',
      `uti_portable` varchar(25) default NULL COMMENT 'Portable',
      `uti_entite` varchar(255) NOT NULL default '' COMMENT 'Entite',
      `uti_adresse` varchar(255) default NULL COMMENT 'Adresse',
      `uti_ville` varchar(255) default NULL COMMENT 'Ville',
      `uti_codepostal` varchar(6) default NULL COMMENT 'Code Postal',
      `uti_last_connection` timestamp NOT NULL default CURRENT_TIMESTAMP COMMENT
'Date de derniere connection',
      PRIMARY KEY (`uti_uid`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COMMENT='Propriete des utilisateurs';

-- --------------------------------------------------------

--
-- Structure de la table `demandesquelette`
--

CREATE TABLE IF NOT EXISTS `demandesquelette` (
      `dsq_id` int(1) unsigned NOT NULL auto_increment,
      `dsq_app_id` int(1) unsigned NOT NULL COMMENT 'Application',
      `dsq_agt_id_list` text default NULL COMMENT 'Liste de serveur',
      `dsq_uid` varchar( 8 ) NOT NULL COMMENT 'identFT',
      `dsq_date_demande` TIMESTAMP default CURRENT_TIMESTAMP COMMENT 'Date de la
demande',
      `dsq_date_dispo` TIMESTAMP NULL default NULL COMMENT 'Date de mise a
disposition',
      `dsq_erreur` CHAR( 1 ) DEFAULT '0' NOT NULL COMMENT 'Erreur de
generation',
      `dsq_traite` CHAR( 1 ) DEFAULT '0' NOT NULL COMMENT 'Traitement effectue',
      `dsq_destination` CHAR( 1 ) DEFAULT '0' NOT NULL COMMENT 'Destination',
```

```
	`dsq_mode` CHAR( 1 ) DEFAULT '0' COMMENT 'Mode',
	`dsq_dom_id` int(1) unsigned default NULL COMMENT 'Domaine Actif',
	`dsq_application_dest_nom` varchar(50) default NULL COMMENT 'Application
Destination Nom',
	`dsq_application_dest_id` int(1) unsigned default NULL COMMENT
'Application Destination Id',
	`dsq_audit_info` TEXT DEFAULT NULL COMMENT 'Message d\'audit',
	`dsq_default_gpco` varchar(255) BINARY NOT NULL COMMENT 'GPCO par defaut',
	PRIMARY KEY (`dsq_id`),
	INDEX (`dsq_app_id`),
	INDEX (`dsq_traite`),
	INDEX (`dsq_erreur`),
	INDEX (`dsq_destination`),
	INDEX (`dsq_mode`),
	FOREIGN KEY (`dsq_uid`) REFERENCES `utilisateur` (`uti_uid`)
) ENGINE=innodb DEFAULT CHARSET=latin1 COMMENT='Liste des demandes de
squelette';


-- dsq_traite = 0 -> en attente squelette
-- dsq_traite = 1 -> en cours d audit
-- dsq_traite = 2 -> audit termine
-- dsq_traite = 3 -> en cours de generation actif
-- dsq_traite = 4 -> termine actif
-- dsq_traite = 5 -> a purger
-- dsq_destination = 0 -> Par BEM
-- dsq_destination = 1 -> Par mail
-- dsq_destination = 2 -> Par Actif
-- dsq_mode = 0 -> New
-- dsq_mode = 1 -> Validation
-- dsq_mode = 2 -> Append

-- --------------------------------------------------------


--
-- Structure de la table `squelette`
--


CREATE TABLE IF NOT EXISTS `squelette` (
	`squ_id` int(1) unsigned NOT NULL auto_increment,
	`squ_dsq_id` int(1) unsigned NOT NULL COMMENT 'Demande de squelette',
	`squ_agt_id` int(1) unsigned NOT NULL COMMENT 'Agent',
	`squ_km` varchar( 255 ) NOT NULL COMMENT 'KM',
	`squ_instance` varchar( 255 ) NOT NULL COMMENT 'Instance',
	`squ_param` varchar( 255 ) NOT NULL COMMENT 'Parametre',
	`squ_aide` varchar( 255 ) DEFAULT '' NOT NULL COMMENT 'Aide',
```

```
      `squ_warn` CHAR( 1 ) DEFAULT '0' NOT NULL COMMENT 'Suceptible warning',
      `squ_alarm` CHAR( 1 ) DEFAULT '0' NOT NULL COMMENT 'Suceptible alarm',
      PRIMARY KEY (`squ_id`),
      FOREIGN KEY (`squ_dsq_id`) REFERENCES `demandesquelette` (`dsq_id`),
      INDEX (`squ_agt_id`)
) ENGINE=innodb DEFAULT CHARSET=latin1 COMMENT='Liste des squelettes';

-- --------------------------------------------------------

--
-- Structure de la table `traceaudit`
--

CREATE TABLE IF NOT EXISTS `traceaudit` (
      `trc_aud_id` int(1) unsigned NOT NULL auto_increment,
      `trc_aud_uid` varchar( 8 ) NOT NULL COMMENT 'identFT',
      `trc_aud_date` timestamp NOT NULL default CURRENT_TIMESTAMP on update
CURRENT_TIMESTAMP COMMENT 'Date execusion',
      `trc_aud_hostname` varchar(50) NOT NULL COMMENT 'hostname',
      `trc_aud_ip` int(1) unsigned NOT NULL COMMENT 'IP',
      `trc_aud_port` smallint(1) unsigned NOT NULL default '0' COMMENT 'Port',
      `trc_aud_user` varchar(50) default NULL COMMENT 'user',
      `trc_aud_pwd` varchar(50) default NULL COMMENT 'passwd',
      `trc_aud_status` char(1) default NULL COMMENT 'Status',
      `trc_aud_xml` longtext default NULL COMMENT 'xml',
      `trc_aud_prerequisagent` Binary(1) NOT NULL default '0' COMMENT 'Audit
prerequisagent',
      `trc_aud_confsysapp` Binary(1) NOT NULL default '0' COMMENT 'Audit
confsysapp',
      `trc_aud_confora` Binary(1) NOT NULL default '0' COMMENT 'Audit confora',
      `trc_aud_confdollu` Binary(1) NOT NULL default '0' COMMENT 'Audit
confdollu',
      `trc_aud_squelette` Binary(1) NOT NULL default '0' COMMENT 'Audit
squelette',
      `trc_aud_blackoutlist` Binary(1) NOT NULL default '0' COMMENT 'Audit
blackoutlist',
      `trc_aud_saveconf` Binary(1) NOT NULL default '0' COMMENT 'Sauvegarde de
conf',
      PRIMARY KEY (`trc_aud_id`),
      FOREIGN KEY (`trc_aud_uid`) REFERENCES `utilisateur` (`uti_uid`)
) ENGINE=innodb DEFAULT CHARSET=latin1 COMMENT='Trace des audits';

-- --------------------------------------------------------

--
-- Structure de la table `trace`
```

```
--


CREATE TABLE IF NOT EXISTS `trace` (
      `trc_id` int(1) unsigned NOT NULL auto_increment,
      `trc_uid` varchar( 8 ) NOT NULL COMMENT 'identFT',
      `trc_date` timestamp NOT NULL default CURRENT_TIMESTAMP on update
CURRENT_TIMESTAMP COMMENT 'Date execusion',
      `trc_mode` ENUM('INSERT', 'DELETE', 'UPDATE') NOT NULL COMMENT 'Mode',
      `trc_table` varchar(20) NOT NULL COMMENT 'Table',
      `trc_clef` int(1) NOT NULL COMMENT 'Clef',
      `trc_champs` text DEFAULT NULL COMMENT 'Champs',
      `trc_request` text NOT NULL COMMENT 'Request',
      PRIMARY KEY (`trc_id`),
      INDEX (`trc_table`),
      INDEX (`trc_clef`),
      FOREIGN KEY (`trc_uid`) REFERENCES `utilisateur` (`uti_uid`)
) ENGINE=innodb DEFAULT CHARSET=latin1 COMMENT='Trace des changements';


-- --------------------------------------------------------
--
-- Structure de la table `domainaccount`
--


CREATE TABLE IF NOT EXISTS `domainaccount` (
      `cpt_dom_id` int(1) unsigned NOT NULL auto_increment,
      `cpt_dom_name` varchar(50) NOT NULL COMMENT 'Domaine',
      `cpt_dom_user` varchar(50) NOT NULL COMMENT 'User',
      `cpt_dom_pwd` varchar(50) character set latin1 collate latin1_bin NOT NULL
COMMENT 'Password',
      PRIMARY KEY (`cpt_dom_id`),
      UNIQUE KEY `agt-cpt_dom_name-user` (`cpt_dom_name`,`cpt_dom_user`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COMMENT='Liste des comptes de
domaines' ;


-- --------------------------------------------------------


--
-- Structure de la table `entite`
--


CREATE TABLE IF NOT EXISTS `entite` (
      `ent_id` int(1) unsigned NOT NULL auto_increment,
      `ent_name` varchar(50) NOT NULL COMMENT 'Nom',
      `ent_ent_id` int(1) unsigned,
      PRIMARY KEY (`ent_id`),
```

```
      UNIQUE KEY `ent_name` (`ent_name`),
      KEY `ent_ent_id` (`ent_ent_id`),
      FOREIGN KEY (`ent_ent_id`) REFERENCES `entite` (`ent_id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COMMENT='Liste des entites' ;

-- --------------------------------------------------------

--
-- Structure de la table `application`
--

CREATE TABLE IF NOT EXISTS `application` (
      `app_id` int(1) unsigned NOT NULL auto_increment,
      `app_name` varchar(50) NOT NULL,
      `app_sup_id` int(1) unsigned,
      `app_priority` ENUM('1', '2', '3', '4', '5') NOT NULL default '1' COMMENT
'Proirite',
      PRIMARY KEY (`app_id`),
      UNIQUE KEY `app_name` (`app_name`),
      FOREIGN KEY (`app_sup_id`) REFERENCES `supteam` (`sup_id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COMMENT='Liste des applications' ;

-- --------------------------------------------------------

--
-- Structure de la table `server`
--

CREATE TABLE IF NOT EXISTS `server` (
      `srv_id` int(1) unsigned NOT NULL auto_increment,
      `srv_ip_valid` CHAR( 1 ) DEFAULT '1' NOT NULL COMMENT 'Validite de l ip',
      `srv_ip` int(1) unsigned NOT NULL COMMENT 'IP',
      `srv_hostname` varchar(50) NOT NULL COMMENT 'Hostname',
      `srv_reel_hostname` varchar(50) default NULL COMMENT 'Virtuel Hostname',
      `srv_exec_valid` CHAR( 1 ) DEFAULT '1' NOT NULL COMMENT 'Validite du
compte execusion',
      `srv_exec_cpt_dom_id` int(1) unsigned default NULL,
      `srv_exec_user` varchar(30) default NULL COMMENT 'User execusion',
      `srv_exec_pwd` varchar(50) character set latin1 collate latin1_bin default
NULL COMMENT 'Password execusion',
      `srv_con_valid` CHAR( 1 ) DEFAULT '1' NOT NULL COMMENT 'Validite du compte
connexion',
      `srv_con_cpt_dom_id` int(1) unsigned default NULL,
      `srv_con_user` varchar(30) default NULL COMMENT 'User connexion',
      `srv_con_pwd` varchar(50) character set latin1 collate latin1_bin default
NULL COMMENT 'Password connexion',
```

```
	`srv_app_type` varchar(15) default NULL COMMENT 'Application Type',
	`srv_platforme` varchar(255) NOT NULL COMMENT 'plate-forme',
	`srv_os` varchar(50) NOT NULL COMMENT 'OS',
	`srv_os_version` varchar(65) NOT NULL COMMENT 'Version OS',
	`srv_nb_cpu` tinyint(1) NOT NULL default '0' COMMENT 'Nombre Cpu',
	`srv_26eres_id` char(11) DEFAULT NULL COMMENT 'Numero ressource 26E',
	PRIMARY KEY (`srv_id`),
	UNIQUE KEY `srv` (`srv_ip`),
	UNIQUE KEY `hostname` (`srv_hostname`),
	INDEX ( `srv_ip_valid` ),
	INDEX ( `srv_exec_valid` ),
	INDEX ( `srv_con_valid` ),
	INDEX ( `srv_26eres_id` ),
	FOREIGN KEY (`srv_con_cpt_dom_id`) REFERENCES `domainaccount`
(`cpt_dom_id`),
	FOREIGN KEY (`srv_exec_cpt_dom_id`) REFERENCES `domainaccount`
(`cpt_dom_id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COMMENT='Liste des servers' ;


-- --------------------------------------------------------


--
-- Structure de la table `agent`
--


CREATE TABLE IF NOT EXISTS `agent` (
	`agt_id` int(1) unsigned NOT NULL auto_increment,
	`agt_srv_id` int(1) unsigned NOT NULL default '0',
	`agt_type`
ENUM('Patrol','Patrol7','Pex','Mom','Nagios','Snmp','Wmi','Manuel') NOT NULL
default 'Patrol' COMMENT 'Type d\'agent',
	`agt_port_valid` CHAR( 1 ) DEFAULT '1' NOT NULL COMMENT 'Validite du
port',
	`agt_port` smallint(1) unsigned default NULL COMMENT 'Port',
	`agt_is_proxy` CHAR( 1 ) DEFAULT '0' NOT NULL COMMENT 'Est proxy',
	`agt_proxy_id` int(1) unsigned DEFAULT NULL COMMENT 'Proxy',
	`agt_name` varchar(50) default NULL COMMENT 'Hostname Agent',
	`agt_date_inventory` TIMESTAMP NULL DEFAULT NULL COMMENT 'Date
Inventaire',
	`agt_date_last_failed` TIMESTAMP NULL DEFAULT NULL COMMENT 'Date dernier
echec Inventaire',
	`agt_version` varchar(20) NOT NULL COMMENT 'Version Agent',
	`agt_home` varchar(255) NOT NULL COMMENT 'Patrol Home',
	`agt_license` DATE NULL DEFAULT NULL COMMENT 'Date de License',
	`agt_defaultaccount_valid` CHAR( 1 ) DEFAULT '1' NOT NULL COMMENT
'Validite du defaultaccount',
	`agt_starttime` TIMESTAMP NULL DEFAULT NULL COMMENT 'Date de demarrage
```

```
Agent',
      `agt_defaultaccount` VARCHAR(45) DEFAULT NULL COMMENT 'DefaultAccount',
      `agt_app_id` int(1) unsigned DEFAULT NULL COMMENT 'Application PEO',
      `agt_26econf_id` int(1) unsigned DEFAULT NULL COMMENT 'Application 26E',
      PRIMARY KEY (`agt_id`),
      UNIQUE KEY `agt` (`agt_srv_id`, `agt_type`, `agt_port`),
      UNIQUE KEY `agt_name` (`agt_name`),
      INDEX ( `agt_port_valid` ),
      INDEX ( `agt_srv_id` ),
      INDEX ( `agt_type` ),
      INDEX ( `agt_is_proxy` ),
      FOREIGN KEY (`agt_srv_id`) REFERENCES `server` (`srv_id`),
      FOREIGN KEY (`agt_proxy_id`) REFERENCES `agent` (`agt_id`),
      FOREIGN KEY (`agt_app_id`) REFERENCES `application` (`app_id`),
      INDEX (`agt_26econf_id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COMMENT='Liste des agents' ;


--
-- Structure de la table `kml`
--

CREATE TABLE IF NOT EXISTS `kml` (
      `kml_id` int(1) unsigned NOT NULL auto_increment,
      `kml_name` varchar(50) NOT NULL COMMENT 'Nom',
      `kml_version` varchar(20) NOT NULL default '' COMMENT 'Version',
      PRIMARY KEY (`kml_id`),
      UNIQUE KEY `kml_name` (`kml_name`,`kml_version`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COMMENT='Liste des KMs' ;

-- --------------------------------------------------------

--
-- Structure de la table `agt_kml`
--

CREATE TABLE IF NOT EXISTS `agt_kml` (
      `agtkml_id` int(1) unsigned NOT NULL auto_increment,
      `agtkml_agt_id` int(1) unsigned NOT NULL,
      `agtkml_kml_id` int(1) unsigned NOT NULL,
      `agtkml_instance` text NOT NULL COMMENT 'Instances',
      `agtkml_preloaded` binary(1) NOT NULL COMMENT 'Preloaded 0=non, 1=oui',
      PRIMARY KEY (`agtkml_id`),
      UNIQUE KEY `agtkml_kml_id-agt_id` (`agtkml_kml_id`,`agtkml_agt_id`),
```

```
      FOREIGN KEY (`agtkml_kml_id`) REFERENCES `kml` (`kml_id`),
      FOREIGN KEY (`agtkml_agt_id`) REFERENCES `agent` (`agt_id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COMMENT='Liste des KMs par agents' ;

-- --------------------------------------------------------

--
-- Structure de la table `despasswd`
--

CREATE TABLE IF NOT EXISTS `despasswd` (
      `des_id` int(1) unsigned NOT NULL auto_increment,
      `des_pwd` varchar(255) character set latin1 collate latin1_bin NOT NULL,
      `des_crypt` varchar(255) default NULL,
      PRIMARY KEY (`des_id`),
      UNIQUE KEY `des_pwd` (`des_pwd`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 ;

--
-- Structure de la table `srvip`
--

CREATE TABLE IF NOT EXISTS `srvip` (
      `srvip_id` int(1) unsigned NOT NULL auto_increment,
      `srvip_srv_id` int(1) unsigned NOT NULL default '0',
      `srvip_ip` int(1) unsigned NOT NULL COMMENT 'IP',
      PRIMARY KEY (`srvip_id`),
      INDEX (`srvip_ip`),
      FOREIGN KEY (`srvip_srv_id`) REFERENCES `server` (`srv_id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COMMENT='Liste des IPs' ;

-- --------------------------------------------------------

--
-- Structure de la table `demandeaudit`
--

CREATE TABLE `demandeaudit` (
      `dau_id` int(1) unsigned NOT NULL auto_increment,
      `dau_agt_id` int(1) unsigned NOT NULL,
      `dau_priority` TINYINT(1) unsigned NOT NULL default 0 COMMENT 'Priority',
      `dau_etat` int(0) NOT NULL default '0' COMMENT 'Etat',
      `dau_date` timestamp NOT NULL default CURRENT_TIMESTAMP COMMENT 'Date de
demande',
```

```
      PRIMARY KEY (`dau_id`),
      INDEX (`dau_priority`),
      INDEX (`dau_etat`),
      FOREIGN KEY (`dau_agt_id`) REFERENCES `agent` (`agt_id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COMMENT='Liste des audit en attente';

-- dau_priority 0 Ajout
-- dau_priority 1 Demande Interface
-- dau_priority 2 Modification Agent
-- dau_priority 3 Modification Serveur
-- dau_priority 4 Automatique mais jamais audite
-- dau_priority 5 Automatique
-- --------------------------------------------------------

--
-- Structure de la table `26e`
--


CREATE TABLE IF NOT EXISTS `26e_ressource` (
      `26eres_id` char(11) NOT NULL,
      `26eres_hostname` varchar(30) NOT NULL COMMENT 'Hostname',
      `26eres_ip` int(1) unsigned COMMENT 'Ip',
      `26eres_type` varchar(30) NOT NULL COMMENT 'Type',
      `26eres_etat` varchar(30) NOT NULL COMMENT 'Etat',
      `26eres_usage` varchar(30) NOT NULL COMMENT 'Usage',
      `26eres_site` varchar(60) NOT NULL COMMENT 'Site',
      `26eres_batiment` varchar(30) NOT NULL COMMENT 'Batiment',
      `26eres_etage` varchar(30) NOT NULL COMMENT 'Etage',
      `26eres_piece` varchar(30) NOT NULL COMMENT 'Piece',
      `26eres_emplacement` varchar(30) NOT NULL COMMENT 'Emplacement',
      `26eres_trave` varchar(30) NOT NULL COMMENT 'Trave',
      `26eres_os` VARCHAR(30) NOT NULL DEFAULT '' COMMENT 'OS',
      PRIMARY KEY (`26eres_id`),
      INDEX (`26eres_hostname`),
      INDEX (`26eres_type`),
      INDEX (`26eres_etat`),
      INDEX (`26eres_usage`),
      INDEX (`26eres_ip`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COMMENT='Liste des serveurs 26e' ;

CREATE TABLE IF NOT EXISTS `26e_configuration` (
      `26econf_id` int(1) unsigned NOT NULL auto_increment,
      `26econf_name` varchar(80) NOT NULL COMMENT 'Nom d\'application',
```

```
	`26econf_dpsi` varchar(80) NOT NULL COMMENT 'DPSI',
	`26econf_dpse` varchar(80) NOT NULL COMMENT 'DPSE',
	`26econf_sox` char(1) NOT NULL COMMENT 'Top SOX',
	`26econf_qs` char(1) NOT NULL COMMENT 'Top QS',
	`26econf_basicat` char(3) COMMENT 'Code Basicat',
	`26econf_eds` varchar(6) NOT NULL COMMENT 'Usage',
	`26econf_ftcarto` varchar(10) NOT NULL COMMENT 'Code FT Carto',
	`26econf_applocal` VARCHAR(10) NOT NULL DEFAULT '' COMMENT 'Code Appli
Local',
	`26econf_niveau` ENUM('Or', 'Argent', 'Bronze') NOT NULL default 'Bronze'
COMMENT 'Niveau de service',
	PRIMARY KEY (`26econf_id`),
	INDEX (`26econf_name`),
	INDEX (`26econf_sox`),
	INDEX (`26econf_niveau`),
	INDEX (`26econf_ftcarto`),
	INDEX (`26econf_applocal`),
	INDEX (`26econf_qs`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COMMENT='Liste des applications de
26e' ;


CREATE TABLE IF NOT EXISTS `26e_ip` (
	`26eip_id` int(1) unsigned NOT NULL auto_increment,
	`26eip_26eres_id` char(11) NOT NULL,
	`26eip_ip` int(1) unsigned NOT NULL COMMENT 'IP',
	PRIMARY KEY (`26eip_id`),
	INDEX (`26eip_ip`),
	FOREIGN KEY (`26eip_26eres_id`) REFERENCES `26e_ressource` (`26eres_id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COMMENT='Liste des IPs' ;


CREATE TABLE IF NOT EXISTS `26e_res_conf` (
	`26eresconf_res_id` char(11) NOT NULL,
	`26eresconf_conf_id` int(1) unsigned NOT NULL,
	PRIMARY KEY (`26eresconf_res_id`,`26eresconf_conf_id`),
	FOREIGN KEY (`26eresconf_res_id`) REFERENCES `26e_ressource`
(`26eres_id`),
	FOREIGN KEY (`26eresconf_conf_id`) REFERENCES `26e_configuration`
(`26econf_id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COMMENT='Liste des liaisons
ressources/applications 26e' ;


-- --------------------------------------------------------


--
-- Trigger pour les tables exportes
--
```

```
DELIMITER //
--
-- Trigger pour la table `agent`
--

-- Avant suppresion agent :
--    Suppression agt-km
--    Suppression demandeaudit
CREATE TRIGGER `before-del-agent` BEFORE DELETE ON `agent`
FOR EACH ROW
BEGIN
      DELETE FROM `agt_kml` WHERE `agtkml_agt_id` = OLD.`agt_id`;
      DELETE FROM `demandeaudit` WHERE `dau_agt_id` = OLD.`agt_id`;
END;//

-- Après suppresion agent :
--    Suppression server s'il n'est plus rattaché
CREATE TRIGGER `after-del-agent` AFTER DELETE ON `agent`
FOR EACH ROW
BEGIN
      DECLARE nb INT;
      SELECT count(*) INTO nb FROM `agent` WHERE `agt_srv_id` = OLD.`agt_srv_id`
AND `agt_type` IN ('Patrol', 'Patrol7');
      IF ( nb = 0 ) THEN
            UPDATE `server` SET srv_exec_valid = 0, srv_con_valid = 0,
srv_exec_cpt_dom_id = NULL, srv_con_cpt_dom_id = NULL, srv_exec_user = NULL,
srv_con_user = NULL, srv_exec_pwd = NULL, srv_con_pwd = NULL WHERE `srv_id` =
OLD.`agt_srv_id` ;
      END IF;
      SELECT count(*) INTO nb FROM `agent` WHERE `agt_srv_id` =
OLD.`agt_srv_id`;
      IF ( nb = 0 ) THEN
            DELETE FROM `server` WHERE `srv_id` = OLD.`agt_srv_id`;
      END IF;
END;//

-- Apres ajout agent :
--    Ajout dans demandeaudit
CREATE TRIGGER `after-add-agent` AFTER INSERT ON `agent`
FOR EACH ROW
BEGIN
      INSERT INTO `demandeaudit` (`dau_agt_id`, `dau_priority`) VALUES
( NEW.`agt_id`, 0);
END;//
```

```
-- Apres modification agent :
--    Ajout dans demandeaudit
CREATE TRIGGER `after-mod-agent` AFTER UPDATE ON `agent`
FOR EACH ROW
BEGIN
      DECLARE nb INT;
      SELECT count(*) INTO nb FROM `demandeaudit` WHERE `dau_agt_id` =
NEW.`agt_id` ;
      IF ( nb = 0 ) THEN
            INSERT INTO `demandeaudit` (`dau_agt_id`, `dau_priority`) VALUES
( NEW.`agt_id`, 2);
      END IF;
END;//


--
-- Trigger pour la table `server`
--


-- Après ajout server :
--    Ajout du mot de passe dans despasswd si non existant
CREATE TRIGGER `after-add-server` AFTER INSERT ON `server`
FOR EACH ROW
BEGIN
      DECLARE nb INT;
      IF ( NEW.`srv_exec_pwd` IS NOT NULL ) THEN
            SELECT count(*) INTO nb FROM `despasswd` WHERE `des_pwd` =
NEW.`srv_exec_pwd` ;
            IF ( nb = 0 ) THEN
                  INSERT INTO `despasswd` ( `des_pwd` ) VALUES
( NEW.`srv_exec_pwd` ) ;
            END IF;
      END IF;
      IF ( NEW.`srv_con_pwd` IS NOT NULL ) THEN
            SELECT count(*) INTO nb FROM `despasswd` WHERE `des_pwd` =
NEW.`srv_con_pwd` ;
            IF ( nb = 0 ) THEN
                  INSERT INTO `despasswd` ( `des_pwd` ) VALUES
( NEW.`srv_con_pwd` ) ;
            END IF;
      END IF;
      IF ( NEW.`srv_ip` IS NOT NULL ) THEN
            INSERT INTO `srvip` ( `srvip_ip`, `srvip_srv_id` ) VALUES
(NEW.srv_ip, NEW.srv_id ) ;
      END IF;
END;//
```

```
-- Après modification server :
--    Suppression du mot de passe dans despasswd si non utile
--    Ajout du mot de passe dans despasswd si non existant
CREATE TRIGGER `after-mod-server` AFTER UPDATE ON `server`
FOR EACH ROW
BEGIN
      DECLARE res INT;
      DECLARE nb INT;
      DECLARE nb1 INT;
      DECLARE nb2 INT;
      DECLARE fini TINYINT DEFAULT 0;
      DECLARE cur CURSOR FOR SELECT `agt_id` FROM `agent` WHERE `agt_srv_id` =
NEW.srv_id ;
      DECLARE CONTINUE HANDLER FOR NOT FOUND SET fini = 0;

      IF ( NOT NEW.`srv_ip` <=> OLD.`srv_ip` ) THEN
            IF ( OLD.`srv_ip` IS NOT NULL ) THEN
                  DELETE FROM `srvip` WHERE `srvip_srv_id` = OLD.`srv_id` AND
srvip_ip = OLD.srv_ip;
            END IF;
            IF ( NEW.`srv_ip` IS NOT NULL ) THEN
                  INSERT INTO `srvip` ( `srvip_ip`, `srvip_srv_id` ) VALUES
(NEW.srv_ip, NEW.srv_id ) ;
            END IF;
      END IF;
      IF ( NOT NEW.`srv_exec_pwd` <=> OLD.`srv_exec_pwd` ) THEN
            IF ( OLD.`srv_exec_pwd` IS NOT NULL ) THEN
                  SELECT count(*) INTO nb2 FROM `domainaccount` WHERE
`cpt_dom_pwd` = OLD.`srv_exec_pwd` ;
                  SELECT count(*) INTO nb1 FROM `server` WHERE `srv_exec_pwd` =
OLD.`srv_exec_pwd` OR `srv_con_pwd` = OLD.`srv_exec_pwd` ;
                  IF ( nb1 = 0 && nb2 = 0 ) THEN
                        DELETE FROM `despasswd` WHERE `des_pwd` =
OLD.`srv_exec_pwd`;
                  END IF;
            END IF;
            IF ( NEW.`srv_exec_pwd` IS NOT NULL ) THEN
                  SELECT count(*) INTO nb FROM `despasswd` WHERE `des_pwd` =
NEW.`srv_exec_pwd` ;
                  IF ( nb = 0 ) THEN
                        INSERT INTO `despasswd` ( `des_pwd` ) VALUES
( NEW.`srv_exec_pwd` ) ;
                  END IF;
            END IF;
      END IF;
      IF ( NOT NEW.`srv_con_pwd` <=> OLD.`srv_con_pwd` ) THEN
            IF ( OLD.`srv_con_pwd` IS NOT NULL ) THEN
```

```
                  SELECT count(*) INTO nb2 FROM `domainaccount` WHERE
`cpt_dom_pwd` = OLD.`srv_con_pwd` ;
                  SELECT count(*) INTO nb1 FROM `server` WHERE `srv_exec_pwd` =
OLD.`srv_con_pwd` OR `srv_con_pwd` = OLD.`srv_con_pwd` ;
                  IF ( nb1 = 0 && nb2 = 0 ) THEN
                        DELETE FROM `despasswd` WHERE `des_pwd` =
OLD.`srv_con_pwd`;
                  END IF;
            END IF;
            IF ( NEW.`srv_con_pwd` IS NOT NULL ) THEN
                  SELECT count(*) INTO nb FROM `despasswd` WHERE `des_pwd` =
NEW.`srv_con_pwd` ;
                  IF ( nb = 0 ) THEN
                        INSERT INTO `despasswd` ( `des_pwd` ) VALUES
( NEW.`srv_con_pwd` ) ;
                  END IF;
            END IF;
      END IF;

      SET fini = 1;
      OPEN cur;
      FETCH cur into res;
      WHILE fini DO
            SELECT count(*) INTO nb FROM `demandeaudit` WHERE `dau_agt_id` = res
;
            IF ( nb = 0 ) THEN
                  INSERT INTO `demandeaudit` (`dau_agt_id`, `dau_priority`)
VALUES ( res, 2);
            END IF;
            FETCH cur into res;
      END WHILE;
      CLOSE cur;
END;//

-- Après suppression server :
--    Suppression du mot de passe dans despasswd si non utile
CREATE TRIGGER `after-del-server` AFTER DELETE ON `server`
FOR EACH ROW
BEGIN
      DECLARE nb1 INT;
      DECLARE nb2 INT;
      IF ( OLD.`srv_exec_pwd` IS NOT NULL ) THEN
            SELECT count(*) INTO nb2 FROM `domainaccount` WHERE `cpt_dom_pwd` =
OLD.`srv_exec_pwd` ;
            SELECT count(*) INTO nb1 FROM `server` WHERE `srv_exec_pwd` =
OLD.`srv_exec_pwd` OR `srv_con_pwd` = OLD.`srv_exec_pwd` ;
            IF ( nb1 = 0 && nb2 = 0 ) THEN
```

```
                  DELETE FROM `despasswd` WHERE `des_pwd` = OLD.`srv_exec_pwd`;
            END IF;
      END IF;
      IF ( OLD.`srv_con_pwd` IS NOT NULL ) THEN
            SELECT count(*) INTO nb2 FROM `domainaccount` WHERE `cpt_dom_pwd` =
OLD.`srv_con_pwd` ;
            SELECT count(*) INTO nb1 FROM `server` WHERE `srv_exec_pwd` =
OLD.`srv_con_pwd` OR `srv_con_pwd` = OLD.`srv_con_pwd` ;
            IF ( nb1 = 0 && nb2 = 0 ) THEN
            DELETE FROM `despasswd` WHERE `des_pwd` = OLD.`srv_con_pwd`;
            END IF;
      END IF;
END;//

-- Avant suppression server :
--    Suppression de toutes les adresses IPs associees
CREATE TRIGGER `before-del-server` BEFORE DELETE ON `server`
FOR EACH ROW
BEGIN
      DELETE FROM `srvip` WHERE `srvip_srv_id` = OLD.`srv_id`;
END;//

--
-- Trigger pour la table `domainaccount`
--

-- Après ajout compte de domaine :
--    Ajout du mot de passe dans despasswd si non existant
CREATE TRIGGER `after-add-domainaccount` AFTER INSERT ON `domainaccount`
FOR EACH ROW
BEGIN
      DECLARE nb INT;
      IF ( NEW.`cpt_dom_pwd` <> '' ) THEN
            SELECT count(*) INTO nb FROM `despasswd` WHERE `des_pwd` =
NEW.`cpt_dom_pwd` ;
            IF ( nb = 0 ) THEN
                  INSERT INTO `despasswd` ( `des_pwd` ) VALUES
( NEW.`cpt_dom_pwd` ) ;
            END IF;
      END IF;
END;//

-- Après modification compte de domaine :
--    Suppression du mot de passe dans despasswd si non utile
--    Ajout du mot de passe dans despasswd si non existant
```

```
CREATE TRIGGER `after-mod-domainaccount` AFTER UPDATE ON `domainaccount`
FOR EACH ROW
BEGIN
      DECLARE nb INT;
      DECLARE nb1 INT;
      DECLARE nb2 INT;
      IF ( NEW.`cpt_dom_pwd` <> OLD.`cpt_dom_pwd` ) THEN
            IF ( NEW.`cpt_dom_pwd` <> '' ) THEN
                  SELECT count(*) INTO nb FROM `despasswd` WHERE `des_pwd` =
NEW.`cpt_dom_pwd` ;
                  IF ( nb = 0 ) THEN
                        INSERT INTO `despasswd` ( `des_pwd` ) VALUES
( NEW.`cpt_dom_pwd` ) ;
                  END IF;
            END IF;
            IF ( OLD.`cpt_dom_pwd` <> '' ) THEN
                  SELECT count(*) INTO nb1 FROM `server` WHERE `srv_exec_pwd` =
OLD.`cpt_dom_pwd` OR `srv_con_pwd` = OLD.`cpt_dom_pwd` ;
                  SELECT count(*) INTO nb2 FROM `domainaccount` WHERE
`cpt_dom_pwd` = OLD.`cpt_dom_pwd` ;
                  IF ( nb1 = 0 && nb2 = 0 ) THEN
                        DELETE FROM `despasswd` WHERE `des_pwd` =
OLD.`cpt_dom_pwd`;
                  END IF;
            END IF;
      END IF;
END;//

-- Après suppression compte de domaine :
--    Suppression du mot de passe dans despasswd si non utile
CREATE TRIGGER `after-del-domainaccount` AFTER DELETE ON `domainaccount`
FOR EACH ROW
BEGIN
      DECLARE nb1 INT;
      DECLARE nb2 INT;
      IF ( OLD.`cpt_dom_pwd` <> '' ) THEN
            SELECT count(*) INTO nb1 FROM `server` WHERE `srv_exec_pwd` =
OLD.`cpt_dom_pwd` OR `srv_con_pwd` = OLD.`cpt_dom_pwd` ;
       SELECT count(*) INTO nb2 FROM `domainaccount` WHERE `cpt_dom_pwd` =
OLD.`cpt_dom_pwd` ;
       IF ( nb1 = 0 && nb2 = 0 ) THEN
        DELETE FROM `despasswd` WHERE `des_pwd` = OLD.`cpt_dom_pwd`;
       END IF;
 END IF;
END;//
```

```
--
-- Trigger pour la table `agt_kml`
--

-- Après suppression d'une liaison km-agent :
--    Suppression du km si non utile
CREATE TRIGGER `after-del-agt-kml` AFTER DELETE ON `agt_kml`
FOR EACH ROW
BEGIN
      DECLARE nb INT;
      SELECT count(*) INTO nb FROM `agt_kml` WHERE `agtkml_kml_id` =
OLD.`agtkml_kml_id`;
      IF ( nb = 0 ) THEN
            DELETE FROM `kml` WHERE `kml_id` = OLD.`agtkml_kml_id`;
      END IF;
END;//

--
-- Trigger pour la table `demandesquelette`
--

-- Avant suppression demandesquelette :
--    Suppression de touts les squelettes associees
CREATE TRIGGER `before-del-demandesquelette` BEFORE DELETE ON
`demandesquelette`
FOR EACH ROW
BEGIN
      DELETE FROM `squelette` WHERE `squ_dsq_id` = OLD.`dsq_id`;
END;//

--
-- Vue mises a disposition de wab
--
CREATE VIEW VIEW_SP_REFERENTIEL (objectclass, parameter) AS SELECT km_nom,
par_nom FROM km, param WHERE km_id = par_km_id ;

CREATE VIEW VIEW_SP_EQUIPMENT (
AP_Code,
AP_Name,
EQ_Name,
EQAP_Port,
EQAP_Collector,
AL_Name,
EQ_Address,
TY_Name,
```

```
OS_Name,
DA_Name,
EQ_LO_Room,
EQ_LO_Trav,
EQ_LO_Bay,
CP_Name,
SU_Name,
AP_Qualif,
AP_Exploit,
AP_Support,
GR_Name,
AP_Priority)
AS SELECT `app_id`,
app_name,
`srv_hostname`,
`agt_port`,
`agt_type`,
IFNULL(`agt_name`, `srv_hostname`) AS srv_hostname,
IntToIP(`srv_ip`),
26eres_type,
`srv_os`,
`26eres_site`,
`26eres_piece`,
`26eres_trave`,
'',
26econf_eds,
sup_name,
CONCAT_WS(', ',UPPER(CAST(26econf_niveau AS CHAR)),if(`26econf_sox`=1,
'SOX',NULL), if(`26econf_qs`=1, 'QS',NULL)),
26econf_dpse,
26econf_dpsi,
GROUP_CONCAT(grpaff_name SEPARATOR ', '),
CONCAT('PRIORITY_', app_priority)
FROM `agent`
LEFT JOIN `26e_configuration` ON `26econf_id` = `agt_26econf_id`,
`server` LEFT JOIN `26e_ressource` ON `26eres_id` = `srv_26eres_id`,
`application`
LEFT JOIN supteam ON sup_id = app_sup_id
LEFT JOIN appgrf ON appgrf_app_id = app_id
LEFT JOIN grpaff ON appgrf_grpaff_id = grpaff_id
WHERE agt_srv_id = srv_id AND `app_id` = `agt_app_id`
GROUP BY agt_id;

CREATE VIEW VIEW_SP_ALARM (application, host, port, objectclass, object,
```

```
parameter, warning, alarm) AS SELECT `app_name`, IFNULL(`agt_name`,
`srv_hostname`), `agt_port`, `squ_km`, `squ_instance`, `squ_param`, `squ_warn`,
`squ_alarm` FROM `squelette`, `application`, agent, serveur WHERE `agt_id` =
`squ_agt_id` AND agt_srv_id = srv_id AND agt_app_id = app_id AND ( `squ_warn`
OR `squ_alarm` );


CREATE VIEW VIEW_SP_EDS (nom, description, horaires, telephone) AS SELECT
eqp_nom, eqp_detail, eqp_horaire, eqp_telephone FROM equipe WHERE eqp_tool =
'Oceane';


-- Initialisation des besoins


INSERT INTO `entite` ( `ent_id`, `ent_name`, `ent_ent_id` ) VALUES ('', '',
NULL );
```

Liste des modules Perl utilisés

IPC-Run 0.80 et IO-Tty 1.07

Ces modules ont servi pour les interrogations des agents Patrol et la communication avec le binaire PatrolCli[6]. Ils permettent les interactions bidirectionnelles avec les outils fournis par BMC.

Perl-ldap 0.33

Ce module permet de valider les accès aux consoles par rapport au domaine AD.

URI 1.35

Ce module permet de simuler des requêtes web sans navigateur. Il sert pour les actions massives sur Spatrol.

XML-Parser 2.34 et XML-Simple 2.15

Ces modules permettent de lire les fichiers XML de façon beaucoup plus native.

Net-SMTP-Multipart 1.5.4 et MIME-Types 1.17

Ce module est utilisé pour l'émission de mail entre autre lors des audits.

File-Temp 0.17

Ce module permet de nativement gérer des fichiers temporaires. Il n'est plus nécessaire de supprimer les fichiers en fin de traitement.

Time-HiRes 1.94

Ce module permet de définir des actions en fonction du temps. Il a fortement été utilisé dans la communication avec les agents.

DBD-Sybase 1.07

Ce module permet d'accéder à une base de données Sybase depuis Perl. Il a servi pour les mises à jour et les interactions avec PEM.

DateManip 5.44 et Time-Format 1.02

Ces modules permettent un traitement simplifié et des conversions de dates automatiques.

Config-INI-Simple 0.01

Ce module permet de lire les fichiers INI de façon beaucoup plus native. Les fichiers INI ont été utilisés pour tout le paramétrage de Spatrol.

[6] Binaire fourni par BMC permettant d'interagir avec les agents Patrol.

Liste des traitements programmés pour l'application dans la crontab

```
# Vérification de la cohérence entre les données de Spatrol et celle de 26E
30      7       *       *       2-6     (cd /usr/local/GestionParc;./CheckIntegrite.pl)
# Ordonnancement des audits soumis en fonction de critères pré-établi de priorité
0,10,20,30,40,50 *      *       *       *       (cd /usr/local/GestionParc;./OrdonnanceAudit.pl)
# Déclenchement des audits en attente.
*       *       *       *       *       (cd /usr/local/GestionParc;./AuditAuto.pl)
# Génération des cartographies Patrol en cas de modification
*       6-20    *       *       *       (cd /usr/local/GestionParc;./CryptPasswd.pl;./GenCartoAuto.pl)
# Génération des fichiers de configuration des adapters Biip pour BEM
*       6-20    *       *       *       (cd /usr/local/GestionParc;./GenBiipAuto.pl)
# Génération des fichiers de configuration des FilterPath
*       8-19    *       *       *       (cd /usr/local/GestionParc;./GenFPAuto.pl)
# Génération du fichier hosts qui est déployé sur les différents serveurs
*       6-20    *       *       *       (cd /usr/local/GestionParc;./GenHosts.pl)
# Génération du paramétrage Nagios tel qu'il est défini
2,7,12,17,22,27,32,37,42,47,52,57  6-20    *       *       *       (cd /usr/local/GestionParc;./GenNagiosAuto.pl)
# Récupération automatique des équipements connectés à l'architecture Patrol7
15      5       *       *       *       (cd /usr/local/GestionParc;./ImportPatrol7.pl)
# Récupération automatique des équipements connectés à l'architecture Mom
0       5       *       *       *       (cd /usr/local/GestionParc;./ImportMom.pl)
# Suppression du fichier de tag des cartographies pour génération complète.
0       4       *       *       *       rm -rf /usr/local/GestionParc/TimerCarto.Ref
# Génération complète des cartographies Patrol
1       4       *       *       *       (cd /usr/local/GestionParc;./GenCartoAuto.pl)
# Génération complète des fichiers d'agent pour la mise dans les cartographies
0       3       *       *       *       (cd /usr/local/GestionParc;./GenMachinesAuto.pl)
# Export de la base afin de la sauvegarder
0       0,7-20  *       *       *       (cd /usr/local/GestionParc;./Sauvegarde.ksh)
# Récupération automatique des données issues de 26E
0       7       *       *       *       (cd /usr/local/GestionParc;./Recup26E.ksh)
# Gestion de la console
0       *       *       *       *       (cd /usr/local/GestionParc;./AnalyseTraceConsole.pl)
# Génération des squelettes demandes
*       *       *       *       *       (cd /usr/local/GestionParc;./ExtractSquelettes.pl)
# Vérifie la non modification de la console de backup par rapport à celle de production
1       0       *       *       *       (cd /usr/local/GestionParc;./CompareConsole.ksh)
# Import Chaîne de soutien de Marine
30      6       *       *       *       (cd /usr/local/GestionParc;./ExtractSoutien.pl)
```

Compte rendu d'audit

Il s'agit d'un compte rendu d'audit pour lequel un bilan des pré-requis agent et de la configuration système et applicative a été demandé.

Audit effectué sur prepem 10.168.145.161 3181.
Audit effectué le 2008-11-10 11:24:28.
Audit effectué sur le serveur prepem

Prerequis :

Les mises à jour de version d'agent (sur les serveurs en pré-J2) peuvent nécessiter une nouvelle intégration de la configuration de supervision, une modification des fiches consignes.

- **Analyse Agent :**
- Information système : SunOS 5.9.
- Version d'agent correcte (V3.7.20i).
- La timezone est positionnée à MET.
- Le compte utilisé pour l'audit appartient aux groupes : patrol, adm, patop, patpop, patwatch, patadm, patscadm.
- Les droits du .profile sont mal positionnés.
  Détail des droits actuels : patrol:patrol rw-r-----
  Il faudrait : patrol:patrol rw-r--r--.
- Le .kshrc n'existe pas.
- Droits sur netstat non trouvés
- Le DCM est bien démarré.
- Le bgscollect est bien démarré.
- KM système trouvé : PLATON_UNIX.kml
- **Analyse Oracle :**

  Le produit Oracle semble installé.
  - Aucun process ora_smon démarré.
  - Aucun process tnslsnr démarré.
  - Le fichier oratab de oracle (/var/opt/oracle/oratab) est accessible en lecture pour patrol.
    Les droits du fichier sont les suivants : rw-r--r--
  - Le fichier listener.ora n'a pas été trouvé.

  Le KM Oracle ne semble pas être installé.
- **Analyse Sentry :**
- La version du KM Sentry est correcte (8.501).
- **Analyse Sentry BlackOut :**
- Le KM Sentry Blackout n'est pas installé.
  Voir avec la DT pour faire installer la version 1.101.
- **Analyse Sentry CFT :**
- La version du KM Sentry CFT est correcte (1.005).

- **Analyse Dollar Universe :**

- La version de DollarU n'a pu être trouvé ou DollarU n'est pas installé.
- Le KM DollarU n'est pas installé.
  Voir avec la DT pour faire installer la version V3.1.
- DollarU semble arrêté (uxord).
- Le répertoire home de DollarU n'a pas pu être identifié.

Analyse de la configuration Système et applicative :

- Liste des processus supervisé(e)s par le KM BMC.
  - (cron) Actuellement 1 occurence(s).
  - (inetd) Actuellement 1 occurence(s).
  - (/usr/sbin/nscd) Actuellement 1 occurence(s).
  - (/usr/lib/saf/sac) Actuellement 1 occurence(s).
  - (syslogd) Actuellement 1 occurence(s).
  - (/usr/lib/saf/ttymon) Actuellement 2 occurence(s).
  - (/usr/lib/utmpd) Actuellement 1 occurence(s).
  - (xntpd) Actuellement 1 occurence(s).
- Aucun(e) processus supervisé(e) par le KM Sentry.
- Aucun(e) processus supervisé(e) par le KM FT.
- Liste des taille(s) de log supervisé(e)s par le KM BMC.
  - Le fichier /system/products/save_sys/logs/save_sys.ksh.log supervisé n'existe pas ou patrol n'a pas le droit d'acces au répertoire.
  - Supervision de la taille du fichier /var/adm/system.log.
    Actuellement les droits sont rw-r--r--, le fichier appartient à root:root, la taille est de 56 Bytes.
- Liste des taille(s) de log supervisé(e)s par le KM Sentry.
  - Supervision de la taille du fichier /tmp/jpa.txt.
    Actuellement les droits sont rwxrwxrwx, le fichier appartient à copatrol:patrol, la taille est de 27 Bytes.
- Aucun(e) taille(s) de log supervisé(e) par le KM FT.
- Aucun(e) droit(s) de log supervisé(e) par le KM FT.
- Liste des log(s) supervisé(e)s par le KM BMC.
  - Le fichier /system/products/save_sys/logs/save_sys.ksh.log supervisé n'existe pas ou patrol n'a pas le droit d'acces au répertoire.
  - Supervision du fichier /var/adm/system.log.
    Actuellement les droits sont rw-r--r--, le fichier appartient à root:root, la taille est de 56 Bytes.
- Liste des log(s) supervisé(e)s par le KM Sentry.
  - Supervision du fichier /tmp/jpa.txt.
    Actuellement les droits sont rwxrwxrwx, le fichier appartient à copatrol:patrol, la taille est de 27 Bytes.
- Aucun(e) log(s) supervisé(e) par le KM FT.

Reconnaissance de Spatrol par BMC

Extrait de mail du mardi 14 octobre 2008 à destination de Monsieur Bonnet,

Merci de votre retour et votre réactivité quand à ma requête.

Comme je l'ai évoqué lors de notre dernier entretien, le niveau de maturité et de maitrîse de la problématique de la supervision atteint par l'entité France Telecom PEOS vous placent dans les clients de référence BMC dans le domaine de la supervision/hypervision et comme la référence BMC France pour des environnements de taille équivalentes (tranches hautes des clients BMC dans le domaine de la supervision avec un parc > à 4000 serveurs monitorés).

Nous avons évalué ce niveau de maturité par :

- La gestion de l'hétérogénéité de l'environnement

- Le niveau de paramétrage de l'hyperviseur rapporté à la charge d'administration de ce dernier (maintenabilité)

- La qualité de l'outillage développé pour adresser les spécificités de l'organisation FT (industrialisation de la solution au travers notamment de l'outil Spatrol)

- La mise en œuvre de boucles d'amélioration de la qualité (reporting diffusé en direction des responsables applicatifs afin de diminuer le nombre d'alarme lié à de faux positifs)

A ce titre, nous aimerions pouvoir vous compter comme site de référence sur la problématique de l'industrialisation de la gestion de supervision et de l'hypervision.

Nous vous remercions de votre retour et restons à votre entière disposition.

Cordialement

Ivan Smets

Senior SW Consultant
Service Assurance

BMC Software

Glossaire

AD	Active-Directory
AJAX	Asynchronous JavaScript and XML
BEM	BMC Event Manager
CMDB	Configuration Management DataBase
CSS	Cascading Style Sheets
DEI	Direction d'Exploitation Informatique
DEPFS	Direction Exploitation des plate-formes de Service
DESI	Direction Exploitation du Service Informatique
DISU	Direction Informatique du Service Utilisateur
DOM	Document Object Model
DOSI	Direction des Opérations du SI
DPS	Direction des plate-formes de Service
DPSE	Direction du Pilotage, de la Supervision et de l'Exploitation
DTF	Direction Technique France
DTO	Direction Technique Opérationnelle
EDS	Element de Structure
FTT	France Télécom Terminaux
HTML	Hypertext Markup Language
IPISO	Ingénierie Production et Infrastrucutres en Systèmes Ouverts
IT	Ingénierie Transverse
ITIL	Information Technology Infrastructure Library
KM	Knoledge Module
LVM	Logical Volume Manager
NFS	Network File System
OCISI	Organisme centrale d'intégration et de Soutien informatiques
OS	Opérating System
PEM	Patrol Event Manager
PEOS	Pole d'expert Outil de Supervision
PEX	Patrol Express
PFS	plate-formes de Service
PSL	Patrol Scripting Langage
QS	Qualité de Service
RIFT	Réseau Interne France Télécom

ROSI	Réseaux, Opérateurs et Système d'Information
SAQS	Supervision d'Administration et de Qualité de Service
SI	Système d'information
SLA	Service Level Agreement
SNMP	Simple Network Management Protocol
SNPI	Service National de Production Informatique
SOX	Sarbanes-Oxley
TDIMG	Télé Diagnostic et télé Intervention en Mode Graphique
USEI	Unité de Supervision et d'exploitation
WAB	Web d'Administration de la BEM
WMI	Windows Management Intrumentation
XML	Extensible_Markup_Language
XSLT	eXtensible Stylesheet Language Translation

Bibliographie et Sources

Documentation de BMC Software	http://www.bmc.com/
Documentation interne sur Patrol écrites par DTO.	
Documentation Perl	http://perldoc.perl.org/perl.html
Documentation Php	http://www.php.net/docs.php
Documentation Apache	http://www.apache.org/
Documentation Mysql	http://www.mysql.fr/
Modules Perl	http://www.cpan.org/

Index des images

Remerciements

Francois Bernard, responsable du PEOS supervision pour sa confiance et les moyens mis à ma disposition pour le développement de cet outil durant ces 4 ans.

Hervé Clerbout et Christian Bethermat, prestataires de service au sein du PEOS pour leur aide, leur apport en connaissances aussi bien liées aux applicatifs BMC Software ainsi que pour leur aide pour résoudre les problèmes liés à l'OS et aux logiciels.

Georges Bonnet, Maîtrise d'ouvrage Outils de supervision pour sa connaissance du SI de Francetelecom et son soutien dans la recherche d'experts sur certaines problématiques.

L'équipe du PEO pour sa disponibilité lors de la mise en place de cet outil, leurs remarques d'amélioration ainsi que pour l'aide lors des réglages de l'outil.