

Extrait du Easter-eggs - Spécialiste GNU/Linux

<http://www.easter-eggs.com>

# Passerelle Request Tracker OpenView Service Desk

- Études de cas - Développement applicatif -

Date de mise en ligne : vendredi 21 mars 2008

## **Description :**

Easter-eggs a intégré Request Tracker (RT) pour une multinationale leader sur le marché de la sécurité électronique. L'outil est spécifiquement adapté au système d'information du client, avec l'implémentation d'une passerelle permettant à RT de communiquer avec l'outil helpdesk de HP OpenView Service Desk (OVSD) via le protocole Case Exchange (CX).

---

**Easter-eggs - Spécialiste GNU/Linux**

---

## Sommaire

- [Introduction](#)
- [Les technologies côté RT](#)
- [Les contraintes](#)
- [Quelques chiffres](#)
- [Surcharge de l'existant](#)
- [Champs personnalisés](#)
- [Modification du schéma de \(...\)](#)
- [Gestion des utilisateurs](#)
- [Gestion des queues](#)
- [Gestion des tickets](#)
- [Fonctionnement du serveur](#)
- [Fonctionnement du client](#)
- [Conclusion](#)

*Il s'agit d'un article technique, décrivant pas à pas, mais de manière succincte, la mise en place de la passerelle.*

## Introduction

Le rôle de la passerelle développée est de permettre la gestion partagée des tickets créés, mis à jour et supprimés aussi bien dans [Request Tracker](#) (RT) que dans [OpenView Service Desk](#) (OVSD). Son fonctionnement est basé sur un serveur RT (auquel OVSD communique les actions à synchroniser), et un client RT (qui communique les actions au serveur OVSD).

La contrainte de départ était que nous devions adapter RT à OVSD, l'inverse n'étant possible que dans une moindre mesure<!-- htmlA --> [1]<!-- htmlB -->. Il a donc fallu s'adapter au protocole d'échange utilisé par celui-ci : Case Exchange. Case Exchange est un protocole texte basé sur XML. La communication entre le client et le serveur se fait via HTTP, en utilisant la méthode POST.

Les modifications effectuées l'ont été à plusieurs niveaux. Tout d'abord, RT a été étendu selon son schéma d'extension habituel, à l'aide de surcharges et ajouts de fichiers spécifiques. De même, profitant de ses possibilités natives, nous avons paramétré un certain nombre de champs personnalisés (« Custom Fields »). Pour des raisons de performances nous avons aussi dû intervenir à un niveau plus bas en modifiant directement la base de données afin d'y ajouter de nouveaux champs SQL et de modifier le type des champs existants.

Ensuite, afin que les changements effectués au niveau de la base de données soient pris en compte dans l'interface Web de RT, il a fallu étendre celle-ci à plusieurs niveaux.

## Les technologies côté RT

- ▶ Système Red Hat Enterprise Linux ES release 4 ;
- ▶ Serveur Web Apache 2.0.52 ;
- ▶

- ▶ Base de données Oracle 9i ;
- ▶ Gestion de tickets RT.

## Les contraintes

- ▶ L'impossibilité d'adapter le code de OVSD ;
- ▶ Un fort trafic ;
- ▶ Une grande quantité de données (utilisateurs, queues, tickets).

## Quelques chiffres

- ▶ Plus de 130 queues ;
- ▶ Plus de 190 groupes ;
- ▶ Plus de 750 utilisateurs « privilégiés »<!-- htmlA --> [2]<!-- htmlB --> ;
- ▶ Plus de 17000 utilisateurs « standard »<!-- htmlA --> [3]<!-- htmlB --> ;
- ▶ Plus de 90000 tickets<!-- htmlA --> [4]<!-- htmlB -->.

## Surcharge de l'existant

RT offre d'emblée la possibilité de l'étendre d'une manière propre et efficace. Toutes les surcharges (qu'il s'agisse de surcharger la définition des types de base, les comportements métier ou bien encore la gestion de son interface Web) se font sous un unique répertoire « local/ ». RT va même jusqu'à proposer plusieurs niveaux de surcharge (surcharge « vendor », surcharge « local » etc.).

Toutes nos modifications ont donc été faites en suivant cette méthode, avec l'ajout de répertoires et de classes spécifiques pour la gestion du client et du serveur de notre passerelle.

## Champs personnalisés

RT propose de base une possibilité d'extension dynamique de sa base de données à l'aide de champs personnalisés. Ces champs peuvent être de plusieurs types et leur comportement peut être contrôlé de manière assez fine.

Nous avons donc privilégié cette solution lorsque nous jugions qu'elle n'aurait qu'un faible impact sur les performances de l'ensemble du système.

Une fois ces champs définis, des scripts d'automatisation de leur création/mise à jour ont été développés en prévision du déploiement. Les interfaces d'affichage, de création et de modification des queues ont également été adaptées en conséquence afin que ces champs soient ajoutés automatiquement en fonction du type de queue sélectionné<!-- htmlA --> [5]<!-- htmlB -->.

## Modification du schéma de la base de données

Certaines modifications effectuées directement dans la base de données auraient pu être faites à l'aide de champs personnalisés RT. Mais les contraintes de performances nous ont rapidement fait oublier cette solution. Des ajouts de champs, ainsi que des modifications de types, ont été faits dans la table des transactions, celles des queues et des tickets, mais également celle des utilisateurs.

Par la suite, un travail méticuleux avec les DBA Oracle a permis l'optimisation des champs ainsi ajoutés ou modifiés. Ce qui aurait été impossible en nous reposant uniquement sur des champs personnalisés.

## Gestion des utilisateurs

Étant donné que la synchronisation doit se faire d'un côté comme de l'autre, il est impératif que les utilisateurs enregistrés sur un système soient également identifiés sur l'autre. RT et OVSD ayant une gestion différentes des utilisateurs, avec un identifiant spécifique, il a fallu ajouter un champ spécial à la table utilisateurs pour y conserver l'identifiant OVSD. Cet identifiant est envoyé lors des synchronisations de tickets.

Afin de pouvoir gérer cet identifiant côté RT, l'interface de gestion des utilisateurs a été modifiée en conséquence.

Nous avons également développé et mis en place un script qui permet de synchroniser la base des utilisateurs OVSD vers la base des utilisateurs RT, à partir d'une extraction LDAP.

## Gestion des queues

Tout d'abord, il a fallu prendre en compte le nouvel attribut « type » associé à chaque queue. Une queue sans type CX est considérée comme une queue « native » RT. Mais dès qu'un type est choisi (« Slave RO », « Slave RW » ou « Both »), la queue est considérée comme faisant partie des échanges CX et comme devant être enrôlée dans le processus de synchronisation. Donc, lorsqu'une queue est considérée comme CX, les champs personnalisés correspondants lui sont automatiquement associés.

Chaque type correspond à un mode de synchronisation bien particulier, qui va déterminer le traitement des transactions associées aux tickets d'une queue CX.

Afin de faciliter l'utilisation de l'interface, ainsi que la création/mise à jour massive des queues, une interface spécifique a été développée pour que celles-ci puissent être gérées directement via un import/export de fichiers CSV. Le format particulier de ces fichiers permet une vérification automatique de la validité des informations de chaque queue. Toutes les queues RT peuvent être traitées ainsi, qu'elles soient CX ou non.

## Gestion des tickets

Bien sûr, la gestion des tickets est le coeur d'un système de gestion des incidents. Les tickets étant identifiés différemment dans RT et dans OVSD, il a fallu ajouter un champ permettant de stocker l'identifiant OVSD. Dans le

cas, par exemple, où le ticket a tout d'abord été créé dans RT, ce sera alors l'accusé de réception de la synchronisation renvoyé par OVSD qui contiendra le nouvel identifiant OVSD. De cette manière, à la réception de cet accusé, l'identifiant OVSD est mis à jour côté RT. C'est cet identifiant qui sera envoyé par la suite lors des synchronisations relatives à ce ticket.

En raison de contraintes propres à OVSD, une gestion spécifique des attachements a été mise en place dans l'interface Web de gestion des tickets. Lorsqu'il s'agit d'un ticket CX, l'utilisateur ne peut ajouter des attachements que lors de la création du ticket, et il est limité à trois attachements maximum pour une taille totale de six-cent kilo-octets.

Étant donné que les tickets ont deux identifiants (RT et OVSD), les interfaces de recherche simple et de recherche avancée ont été modifiées pour pouvoir, en cas de besoin, rechercher un ticket directement à partir de son numéro OVSD.

## Fonctionnement du serveur

OVSD communique avec RT par l'intermédiaire d'un serveur CX, ajouté dans RT. Ce serveur reçoit les POST OVSD au format CX, et les convertit afin d'appliquer les opérations dans la base RT.

Les actions synchronisées sont préalablement validées par un schéma XML afin de garantir leur cohérence. Lors de la détection d'une erreur, les données qui lui sont relatives sont stockées dans la table des transactions RT afin qu'elles soient renvoyées de manière asynchrone par le client RT vers OVSD.

Chaque réception de données par RT donne lieu à l'envoi d'un accusé de réception.

## Fonctionnement du client

RT communique avec OVSD par l'intermédiaire d'un client CX, ajouté dans RT. Ce script client est déclenché à intervalles réguliers par le scheduler « Control-M ». Il est chargé de poster vers OVSD, au format CX, les actions effectuées dans RT sur les tickets.

Chaque réception de données par OVSD donne lieu à l'envoi d'un accusé de réception.

## Conclusion

En conclusion, on peut dire que la solution retenue pour la mise en oeuvre de cette passerelle était la bonne. RT joue pleinement son rôle, et sa souplesse a permis un développement respectueux des contraintes du client et adapté à ses besoins.

L'application est en production depuis fin 2007, et nous sommes amenés à la faire évoluer régulièrement afin de l'adapter aux nouvelles demandes.

<!-- htmlA -->[\[1\]](#) <!-- htmlB --> Les seules interventions possibles côté OVSD se font au niveau de la configuration de l'outil. Cette configuration peut être poussée assez loin, mais aucune modification du logiciel n'était envisageable.

<!-- htmlA -->[\[2\]](#) <!-- htmlB --> Utilisateurs ayant les droits nécessaires pour se connecter et traiter des tickets.

<!-- htmlA -->[\[3\]](#) <!-- htmlB --> Utilisateurs utilisant la gestion d'incidents au quotidien, et soumettant des problèmes.

<!-- htmlA -->[\[4\]](#) <!-- htmlB --> Ce chiffre étant bien sûr amené à augmenter quotidiennement.

<!-- htmlA -->[\[5\]](#) <!-- htmlB --> Comme indiqué plus loin dans cet article, nous avons ajouté un attribut supplémentaire aux queues afin de leur donner un type.