Vigilo NMS Documentation

Version 4.1.0

CSSI

nov. 02, 2017

Table des matières

Table des matières :

CHAPITRE 1

Introduction

CHAPITRE 2

Installation

CHAPITRE 3

Prise en main

CHAPITRE 4

Administration

Table des matières :

4.1 Guide d'administration

4.1.1 Installation

Pré-requis logiciels

Afin de pouvoir faire fonctionner le connecteur de métrologie, l'installation préalable des logiciels suivants est requise :

- python (>= 2.5), sur la machine où le connecteur est installé
- rabbitmq (>= 2.7.1), éventuellement sur une machine distante
- rrdtool (>= 1.3), sur la machine où le connecteur est installé

Création du compte sur le bus

Le connecteur FTP nécessite qu'un compte soit créé sur la machine hébergeant le bus. Les comptes doivent être créés sur la machine qui héberge le serveur rabbitmq, à l'aide de la commande :

```
# rabbitmqctl add_user nom_d_utilisateur mot_de_passe
```

4.1.2 Configuration

Le module |name| est fourni avec un fichier de configuration situé par défaut dans « /etc/vigilo/|name|/settings.ini ».

Ce fichier est composé de différentes sections permettant de paramétrer des aspects divers du module, chacune de ces sections peut contenir un ensemble de valeurs sous la forme `clé = valeur`. Les lignes commençant par « ; » ou « # » sont des commentaires et sont par conséquent ignorées.

Le format de ce fichier peut donc être résumé dans l'extrait suivant :

```
# Ceci est un commentaire
; Ceci est également un commentaire
[section1]
option1=valeur1
option2=valeur2
; ...

[section2]
option1=val1
; ...
```

Les sections utilisées par le connecteur et leur rôle sont détaillées ci-dessous :

bus Contient les options relatives à la configuration de l'accès au bus de messages.

connector Contient les options de configuration génériques d'un connecteur de Vigilo.

publications Contient une liste d'associations entre les types de messages envoyés et les nœuds de publication (exchange) vers lesquels ces messages sont transmis. Un paramètre optionnel permet de définir une durée de vie en secondes pour les messages (si un message passe plus de temps que sa durée de vie dans la file d'attente sans être consommé, il est automatiquement supprimé de la file d'attente).

loggers, handlers, formatters, logger_*, handler_*, formatter_* Contient la configuration du mécanisme de journalisation des événements (voir chapitre logging).

« * » correspond au nom d'un logger/handler/formatter défini dans la section loggers, handlers ou formatters (respectivement).

connector-metro Contient les options spécifiques au connecteur metro.

Connexion au bus de messages

Le connecteur utilise un bus de communication basé sur le protocole AMQP pour communiquer avec les autres connecteurs de Vigilo.

Ce chapitre décrit les différentes options de configuration se rapportant à la connexion à ce bus de communication, situées dans la section `[bus]` du fichier de configuration.

Trace des messages

L'option « log_traffic » est un booléen permettant d'afficher tous les messages échangés avec le bus lorsqu'il est positionné à « True ». Cette option génère un volume d'événements de journalisation très important et n'est donc pas conseillée en production.

Adresse du bus

L'option « host » permet d'indiquer le nom ou l'adresse IP de l'hôte sur lequel le bus fonctionne.

Service de publication

Le connecteur utilise les mécanismes de publication de messages du protocole AMQP pour échanger des informations avec les autres connecteurs de Vigilo.

Ces mécanismes nécessites de spécifier le nom du service de publication utilisé pour l'échange de messages sur le bus, appelé *exchange*. Par défaut, le nom de ce service est le nom du type de message à envoyer.

Identifiant

Chaque connecteur de Vigilo est associé à un compte AMQP différent. L'option « user » permet d'indiquer le nom de ce compte.

Mot de passe

L'option « password » permet de spécifier le mot de passe associé au compte AMQP indiqué dans l'option « user ».

Connexions sécurisées

Les connecteurs ont la possibilité de spécifier la politique de sécurité à appliquer pour les connexions avec le bus. Il est possible de forcer l'utilisation d'une connexion chiffrée entre le connecteur et le bus en positionnant l'option « require_ssl » à « True ». Le port de connexion utilisé par défaut sans SSL est le `5672`, avec SSL il devient le `5671`.

Délai maximum de reconnexion

En cas de déconnexion du bus, les connecteurs se reconnectent automatiquement, selon un délai qui augmente exponentiellement (afin d'éviter d'innonder les journaux système avec des messages annonçant la reconnexion).

L'option « max_reconnect_delay » permet d'indiquer le délai maximum (en secondes) qui peut s'écouler entre 2 tentatives de reconnexion. Par défaut, ce délai est fixé à 60 secondes.

File d'attente

L'option « queue » permet de spécifier le nom de la file d'attente AMQP à laquelle se connecter pour recevoir les messages. Si plusieurs connecteurs spécifient la même file d'attente, il en consommeront les messages au fur et à mesure de leur capacité de traitement (mode « répartition de charge »).

Abonnements

L'option « subscriptions » contient la liste des nœuds de publication auxquels le connecteur est abonné (séparés par des virgules). Plus exactement, il s'agit des nœuds de publication auxquels la file d'attente spécifiée par l'option « queue » est abonnée. La valeur configurée par défaut lors de l'installation du connecteur convient généralement à tous les types d'usage.

Attention, il est possible d'ajouter des abonnements dans cette liste, mais les abonnements existants ne seront pas supprimés automatiquement s'ils sont supprimés de la liste (il s'agit d'une limitation actuelle du protocole). Pour cela, il faut passer soit par l'interface de gestion du serveur, soit par la commande `vigilo-config-bus`.

État du connecteur

Les connecteurs de Vigilo sont capables de s'auto-superviser, c'est-à-dire que des alertes peuvent être émises par Vigilo concernant ses propres connecteurs lorsque le fonctionnement de ceux-ci est perturbé ou en défaut.

Ce mécanisme est rendu possible grâce à un signal de vie émis (un état Nagios) par chaque connecteur à intervalle régulier. Chaque signal de vie correspond à un message de type « nagios ».

L'option « status_service » permet de spécifier le nom du service Nagios par lequel on supervise ce connecteur.

Des données de performance sont également générées afin d'alimenter les graphiques de métrologie des différents connecteurs. Chaque donnée de performance correspond à un message de type « perf ».

Les options « self_monitoring_nagios_exchange » et « self_monitoring_perf_exchange » permettent de choisir les nœuds de publication vers lesquels les messages d'état Nagios et de performance du connecteur sont envoyés, respectivement. Dans le cas où cette option ne serait pas renseignée, les nœuds configurés dans la section `[publication]` sont utilisés pour déterminer la destination des messages. Si aucun nœud n'est configuré pour l'envoi des messages d'état Nagios ou de performance, un message d'erreur est enregistré dans les journaux d'événements.

Destination des messages

Le connecteur envoie des messages au bus contenant des informations sur l'état des éléments du parc, ainsi que des données de métrologie permettant d'évaluer la performance des équipements. Chaque message transmis par le connecteur possède un type.

La section `[publications]` permet d'associer le type des messages à un nœud de publication. Ainsi, chaque fois qu'un message doit être transmis au bus, le connecteur consulte cette liste d'associations afin de connaître le nom du nœud sur lequel il doit publier son message.

Les types de messages supportés par un connecteur sont :

- `perf` : messages de performances
- `state` : messages d'état
- `event` : messages d'événements
- `nagios` : commandes Nagios
- `command` : commandes diverses

Si aucune destination n'est renseignée, le message sera envoyé sur un nœud du même nom que le type du message.

Exemple de configuration possible, correspondant à une installation standard :

```
[publications]
perf   = perf
state  = state
event  = event
nagios = nagios
```

Journaux

Le connecteur est capable de transmettre un certain nombre d'informations au cours de son fonctionnement à un mécanisme de journalisation des événements (par exemple, des journaux systèmes, une trace dans un fichier, un enregistrement des événements en base de données, etc.).

Le document Vigilo - Journaux d'événements décrit spécifiquement la configuration de la journalisation des événements au sein de toutes les applications de Vigilo, y compris les connecteurs.

Configuration spécifique au connecteur de métrologie

Ce chapitre décrit les options de configuration spécifiques au connecteur de métrologie. Ces options sont situées dans la section `[connector-metro]` du fichier de configuration (dans `/etc/vigilo/connector-metro/settings.ini`).

Emplacement du fichier de configuration auto-généré

Le connecteur de métrologie utilise un fichier de configuration auto-généré (par VigiConf) afin de connaître la liste des équipements du parc dont il a la responsabilité pour le stockage des données de métrologie.

L'option « config » permet de spécifier l'emplacement de ce fichier de configuration auto-généré. En règle générale, il s'agira de `/etc/vigilo/connector-metro/connector-metro.conf.py`.

Dossier de stockage des fichiers RRD

L'option « rrd_base_dir » donne le nom du dossier racine sous lequel les données de métrologie seront enregistrées.

Le module connector-metro crée automatiquement un dossier au nom de l'hôte la première fois qu'il reçoit une mesure de métrologie portant sur cet hôte. À l'intérieur de ce dossier, un fichier « .rrd » est créé pour chaque indicateur de métrologie disponible sur cet hôte ou sur l'un de ses services.

Emplacement de l'outil « rrdtool »

L'option « rrd_bin » donne l'emplacement de l'outil « rrdtool » sur le système. Une valeur adéquate est « /usr/bin/rrdtool » car il s'agit de l'emplacement par défaut de cet outil sur la plupart des distributions Linux.

4.1.3 Annexes

Glossaire - Terminologie

Ce chapitre recense les différents termes techniques employés dans ce document et donne une brève définition de chacun de ces termes.

AMQP Advanced Message Queuing Protocol. Protocole ouvert de messagerie applicative. Voir amqp.org.

exchange Nœud de publication dans le dialecte AMQP. Les messages publiés sur un *exchange* sont relayés à une ou plusieurs files d'attente sur le serveur, à partir desquelles les connecteurs vont consommer les messages.

inotify L'outil inotify est un mécanisme du noyau Linux qui fournit des notifications concernant le système de fichiers lorsqu'un événement particulier se produit (par exemple, la fermeture d'un fichier).

JSON JavaScript Object Notation. Méthode de sérialisation textuelle compatible JavaScript.

RRD Round-Robin Database. Base de données circulaire permettant de stocker des données disposant d'une granularité différente.

SGBD Système de Gestion de Base de Données. Logiciel permettant d'héberger une base de données sur la machine.

URL Uniform Resource Locator. Chaîne de caractères permettant d'identifier une ressource sur Internet.

XML eXtensible Markup Language. Langage de balisage extensible.

4.2 Guide d'administration

4.2.1 Installation

Pré-requis logiciels

Afin de pouvoir faire fonctionner le connecteur Nagios, l'installation préalable des logiciels suivants est requise :

— python (>= 2.5), sur la machine où le connecteur est installé
— rabbitmq (>= 2.7.1), éventuellement sur une machine distante
— nagios (>= 3.1.0) sur la machine où le connecteur est installé

Création du compte sur le bus

Le connecteur FTP nécessite qu'un compte soit créé sur la machine hébergeant le bus. Les comptes doivent être créés sur la machine qui héberge le serveur rabbitmq, à l'aide de la commande :

```
# rabbitmqctl add_user nom_d_utilisateur mot_de_passe
```

4.2.2 Configuration

Le module |name| est fourni avec un fichier de configuration situé par défaut dans « /etc/vigilo/|name|/settings.ini ».

Ce fichier est composé de différentes sections permettant de paramétrer des aspects divers du module, chacune de ces sections peut contenir un ensemble de valeurs sous la forme `clé = valeur`. Les lignes commençant par « ; » ou « # » sont des commentaires et sont par conséquent ignorées.

Le format de ce fichier peut donc être résumé dans l'extrait suivant :

```
# Ceci est un commentaire
; Ceci est également un commentaire
[section1]
option1=valeur1
option2=valeur2
; ...

[section2]
option1=val1
; ...
```

Les sections utilisées par le connecteur et leur rôle sont détaillées ci-dessous :

bus Contient les options relatives à la configuration de l'accès au bus de messages.

connector Contient les options de configuration génériques d'un connecteur de Vigilo.

publications Contient une liste d'associations entre les types de messages envoyés et les nœuds de publication (exchange) vers lesquels ces messages sont transmis. Un paramètre optionnel permet de définir une durée de vie en secondes pour les messages (si un message passe plus de temps que sa durée de vie dans la file d'attente sans être consommé, il est automatiquement supprimé de la file d'attente).

loggers, handlers, formatters, logger_*, handler_*, formatter_* Contient la configuration du mécanisme de journalisation des événements (voir chapitre logging).

« * » correspond au nom d'un logger/handler/formatter défini dans la section loggers, handlers ou formatters (respectivement).

connector-nagios Contient les options spécifiques au connecteur Nagios.

Connexion au bus de messages

Le connecteur utilise un bus de communication basé sur le protocole AMQP pour communiquer avec les autres connecteurs de Vigilo.

Ce chapitre décrit les différentes options de configuration se rapportant à la connexion à ce bus de communication, situées dans la section `[bus]` du fichier de configuration.

Trace des messages

L'option « log_traffic » est un booléen permettant d'afficher tous les messages échangés avec le bus lorsqu'il est positionné à « True ». Cette option génère un volume d'événements de journalisation très important et n'est donc pas conseillée en production.

Adresse du bus

L'option « host » permet d'indiquer le nom ou l'adresse IP de l'hôte sur lequel le bus fonctionne.

Service de publication

Le connecteur utilise les mécanismes de publication de messages du protocole AMQP pour échanger des informations avec les autres connecteurs de Vigilo.

Ces mécanismes nécessites de spécifier le nom du service de publication utilisé pour l'échange de messages sur le bus, appelé *exchange*. Par défaut, le nom de ce service est le nom du type de message à envoyer.

Identifiant

Chaque connecteur de Vigilo est associé à un compte AMQP différent. L'option « user » permet d'indiquer le nom de ce compte.

Mot de passe

L'option « password » permet de spécifier le mot de passe associé au compte AMQP indiqué dans l'option « user ».

Connexions sécurisées

Les connecteurs ont la possibilité de spécifier la politique de sécurité à appliquer pour les connexions avec le bus. Il est possible de forcer l'utilisation d'une connexion chiffrée entre le connecteur et le bus en positionnant l'option « require_ssl » à « True ». Le port de connexion utilisé par défaut sans SSL est le `5672`, avec SSL il devient le `5671`.

Délai maximum de reconnexion

En cas de déconnexion du bus, les connecteurs se reconnectent automatiquement, selon un délai qui augmente exponentiellement (afin d'éviter d'innonder les journaux système avec des messages annonçant la reconnexion).

L'option « max_reconnect_delay » permet d'indiquer le délai maximum (en secondes) qui peut s'écouler entre 2 tentatives de reconnexion. Par défaut, ce délai est fixé à 60 secondes.

File d'attente

L'option « queue » permet de spécifier le nom de la file d'attente AMQP à laquelle se connecter pour recevoir les messages. Si plusieurs connecteurs spécifient la même file d'attente, il en consommeront les messages au fur et à mesure de leur capacité de traitement (mode « répartition de charge »).

Abonnements

L'option « subscriptions » contient la liste des nœuds de publication auxquels le connecteur est abonné (séparés par des virgules). Plus exactement, il s'agit des nœuds de publication auxquels la file d'attente spécifiée par l'option « queue » est abonnée. La valeur configurée par défaut lors de l'installation du connecteur convient généralement à tous les types d'usage.

Attention, il est possible d'ajouter des abonnements dans cette liste, mais les abonnements existants ne seront pas supprimés automatiquement s'ils sont supprimés de la liste (il s'agit d'une limitation actuelle du protocole). Pour cela, il faut passer soit par l'interface de gestion du serveur, soit par la commande `vigilo-config-bus`.

État du connecteur

Les connecteurs de Vigilo sont capables de s'auto-superviser, c'est-à-dire que des alertes peuvent être émises par Vigilo concernant ses propres connecteurs lorsque le fonctionnement de ceux-ci est perturbé ou en défaut.

Ce mécanisme est rendu possible grâce à un signal de vie émis (un état Nagios) par chaque connecteur à intervalle régulier. Chaque signal de vie correspond à un message de type « nagios ».

L'option « status_service » permet de spécifier le nom du service Nagios par lequel on supervise ce connecteur.

Des données de performance sont également générées afin d'alimenter les graphiques de métrologie des différents connecteurs. Chaque donnée de performance correspond à un message de type « perf ».

Les options « self_monitoring_nagios_exchange » et « self_monitoring_perf_exchange » permettent de choisir les nœuds de publication vers lesquels les messages d'état Nagios et de performance du connecteur sont envoyés, respectivement. Dans le cas où cette option ne serait pas renseignée, les nœuds configurés dans la section `[publication]` sont utilisés pour déterminer la destination des messages. Si aucun nœud n'est configuré pour l'envoi des messages d'état Nagios ou de performance, un message d'erreur est enregistré dans les journaux d'événements.

Destination des messages

Le connecteur envoie des messages au bus contenant des informations sur l'état des éléments du parc, ainsi que des données de métrologie permettant d'évaluer la performance des équipements. Chaque message transmis par le connecteur possède un type.

La section `[publications]` permet d'associer le type des messages à un nœud de publication. Ainsi, chaque fois qu'un message doit être transmis au bus, le connecteur consulte cette liste d'associations afin de connaître le nom du nœud sur lequel il doit publier son message.

Les types de messages supportés par un connecteur sont :

- `perf` : messages de performances
- `state` : messages d'état
- `event` : messages d'événements
- `nagios` : commandes Nagios
- `command` : commandes diverses

Si aucune destination n'est renseignée, le message sera envoyé sur un nœud du même nom que le type du message.

Exemple de configuration possible, correspondant à une installation standard :

```
[publications]
perf    = perf
state   = state
event   = event
nagios = nagios
```

Journaux

Le connecteur est capable de transmettre un certain nombre d'informations au cours de son fonctionnement à un mécanisme de journalisation des événements (par exemple, des journaux systèmes, une trace dans un fichier, un enregistrement des événements en base de données, etc.).

Le document Vigilo - Journaux d'événements décrit spécifiquement la configuration de la journalisation des événements au sein de toutes les applications de Vigilo, y compris les connecteurs.

Configuration spécifique au connecteur Nagios

Cette section décrit les options de configuration spécifiques au connecteur Nagios. Ces options sont situées dans la section `[connector-nagios]` du fichier de configuration (dans `/etc/vigilo/connector-nagios/settings.ini`).

Commandes Nagios acceptées

Le connecteur Nagios est capable d'envoyer des commandes à destination de Nagios. Afin d'éviter des abus éventuels, l'option « accepted_commands » permet de lister les commandes qui seront acceptées.

À minima, la commande « PROCESS_SERVICE_CHECK_RESULT » doit être acceptée si des services de haut niveau ont été configurés au travers de VigiConf.

Socket de réception des messages

L'option « listen_unix » permet d'indiquer l'emplacement du socket Unix sur lequel le connecteur attendra des messages (généralement émis directement par Nagios).

Pipe de commandes Nagios

L'option « nagios_pipe » permet de spécifier l'emplacement du « pipe » (canal de communication) sur lequel Nagios accepte des commandes. La valeur de cette option doit être la même que l'option portant le même nom dans le fichier de configuration de Nagios (nagios.cfg).

4.2.3 Annexes

Glossaire - Terminologie

Ce chapitre recense les différents termes techniques employés dans ce document et donne une brève définition de chacun de ces termes.

AMQP Advanced Message Queuing Protocol. Protocole ouvert de messagerie applicative. Voir amqp.org.

exchange Nœud de publication dans le dialecte AMQP. Les messages publiés sur un *exchange* sont relayés à une ou plusieurs files d'attente sur le serveur, à partir desquelles les connecteurs vont consommer les messages.

inotify L'outil inotify est un mécanisme du noyau Linux qui fournit des notifications concernant le système de fichiers lorsqu'un événement particulier se produit (par exemple, la fermeture d'un fichier).

JSON JavaScript Object Notation. Méthode de sérialisation textuelle compatible JavaScript.

RRD Round-Robin Database. Base de données circulaire permettant de stocker des données disposant d'une granularité différente.

SGBD Système de Gestion de Base de Données. Logiciel permettant d'héberger une base de données sur la machine.

URL Uniform Resource Locator. Chaîne de caractères permettant d'identifier une ressource sur Internet.

XML eXtensible Markup Language. Langage de balisage extensible.

4.3 Guide d'administration

4.3.1 Installation

Pré-requis logiciels

Afin de pouvoir faire fonctionner le connecteur syncevents, l'installation préalable des logiciels suivants est requise :
- python (>= 2.5), sur la machine où le connecteur est installé
- rabbitmq (>= 2.7.1), éventuellement sur une machine distante
- postgresql-server (>= 8.3), éventuellement sur une machine distante

Création du compte sur le bus

Le connecteur FTP nécessite qu'un compte soit créé sur la machine hébergeant le bus. Les comptes doivent être créés sur la machine qui héberge le serveur rabbitmq, à l'aide de la commande :

```
# rabbitmqctl add_user nom_d_utilisateur mot_de_passe
```

4.3.2 Configuration

Le module |name| est fourni avec un fichier de configuration situé par défaut dans « /etc/vigilo/|name|/settings.ini ».

Ce fichier est composé de différentes sections permettant de paramétrer des aspects divers du module, chacune de ces sections peut contenir un ensemble de valeurs sous la forme `clé = valeur`. Les lignes commençant par « ; » ou « # » sont des commentaires et sont par conséquent ignorées.

Le format de ce fichier peut donc être résumé dans l'extrait suivant :

```
# Ceci est un commentaire
; Ceci est également un commentaire
[section1]
option1=valeur1
option2=valeur2
; ...

[section2]
option1=val1
; ...
```

Les sections utilisées par le connecteur et leur rôle sont détaillées ci-dessous :

bus Contient les options relatives à la configuration de l'accès au bus de messages.

connector Contient les options de configuration génériques d'un connecteur de Vigilo.

publications Contient une liste d'associations entre les types de messages envoyés et les nœuds de publication (exchange) vers lesquels ces messages sont transmis. Un paramètre optionnel permet de définir une durée de vie en secondes pour les messages (si un message passe plus de temps que sa durée de vie dans la file d'attente sans être consommé, il est automatiquement supprimé de la file d'attente).

loggers, handlers, formatters, logger_*, handler_*, formatter_* Contient la configuration du mécanisme de journalisation des événements (voir chapitre logging).

« * » correspond au nom d'un logger/handler/formatter défini dans la section loggers, handlers ou formatters (respectivement).

connector-syncevents Contient les options spécifiques au connecteur syncevents.

database Contient les options relatives à la connexion à la base de données de Vigilo.

Connexion au bus de messages

Le connecteur utilise un bus de communication basé sur le protocole AMQP pour communiquer avec les autres connecteurs de Vigilo.

Ce chapitre décrit les différentes options de configuration se rapportant à la connexion à ce bus de communication, situées dans la section `[bus]` du fichier de configuration.

Trace des messages

L'option « log_traffic » est un booléen permettant d'afficher tous les messages échangés avec le bus lorsqu'il est positionné à « True ». Cette option génère un volume d'événements de journalisation très important et n'est donc pas conseillée en production.

Adresse du bus

L'option « host » permet d'indiquer le nom ou l'adresse IP de l'hôte sur lequel le bus fonctionne.

Service de publication

Le connecteur utilise les mécanismes de publication de messages du protocole AMQP pour échanger des informations avec les autres connecteurs de Vigilo.

Ces mécanismes nécessites de spécifier le nom du service de publication utilisé pour l'échange de messages sur le bus, appelé *exchange*. Par défaut, le nom de ce service est le nom du type de message à envoyer.

Identifiant

Chaque connecteur de Vigilo est associé à un compte AMQP différent. L'option « user » permet d'indiquer le nom de ce compte.

Mot de passe

L'option « password » permet de spécifier le mot de passe associé au compte AMQP indiqué dans l'option « user ».

Connexions sécurisées

Les connecteurs ont la possibilité de spécifier la politique de sécurité à appliquer pour les connexions avec le bus. Il est possible de forcer l'utilisation d'une connexion chiffrée entre le connecteur et le bus en positionnant l'option « require_ssl » à « True ». Le port de connexion utilisé par défaut sans SSL est le `5672`, avec SSL il devient le `5671`.

Délai maximum de reconnexion

En cas de déconnexion du bus, les connecteurs se reconnectent automatiquement, selon un délai qui augmente exponentiellement (afin d'éviter d'innonder les journaux système avec des messages annonçant la reconnexion).

L'option « max_reconnect_delay » permet d'indiquer le délai maximum (en secondes) qui peut s'écouler entre 2 tentatives de reconnexion. Par défaut, ce délai est fixé à 60 secondes.

File d'attente

L'option « queue » permet de spécifier le nom de la file d'attente AMQP à laquelle se connecter pour recevoir les messages. Si plusieurs connecteurs spécifient la même file d'attente, il en consommeront les messages au fur et à mesure de leur capacité de traitement (mode « répartition de charge »).

Abonnements

L'option « subscriptions » contient la liste des nœuds de publication auxquels le connecteur est abonné (séparés par des virgules). Plus exactement, il s'agit des nœuds de publication auxquels la file d'attente spécifiée par l'option « queue » est abonnée. La valeur configurée par défaut lors de l'installation du connecteur convient généralement à tous les types d'usage.

Attention, il est possible d'ajouter des abonnements dans cette liste, mais les abonnements existants ne seront pas supprimés automatiquement s'ils sont supprimés de la liste (il s'agit d'une limitation actuelle du protocole). Pour cela, il faut passer soit par l'interface de gestion du serveur, soit par la commande `vigilo-config-bus`.

État du connecteur

Les connecteurs de Vigilo sont capables de s'auto-superviser, c'est-à-dire que des alertes peuvent être émises par Vigilo concernant ses propres connecteurs lorsque le fonctionnement de ceux-ci est perturbé ou en défaut.

Ce mécanisme est rendu possible grâce à un signal de vie émis (un état Nagios) par chaque connecteur à intervalle régulier. Chaque signal de vie correspond à un message de type « nagios ».

L'option « status_service » permet de spécifier le nom du service Nagios par lequel on supervise ce connecteur.

Des données de performance sont également générées afin d'alimenter les graphiques de métrologie des différents connecteurs. Chaque donnée de performance correspond à un message de type « perf ».

Les options « self_monitoring_nagios_exchange » et « self_monitoring_perf_exchange » permettent de choisir les nœuds de publication vers lesquels les messages d'état Nagios et de performance du connecteur sont envoyés, respectivement. Dans le cas où cette option ne serait pas renseignée, les nœuds configurés dans la section `[publication]` sont utilisés pour déterminer la destination des messages. Si aucun nœud n'est configuré pour l'envoi des messages d'état Nagios ou de performance, un message d'erreur est enregistré dans les journaux d'événements.

Destination des messages

Le connecteur envoie des messages au bus contenant des informations sur l'état des éléments du parc, ainsi que des données de métrologie permettant d'évaluer la performance des équipements. Chaque message transmis par le connecteur possède un type.

La section `[publications]` permet d'associer le type des messages à un nœud de publication. Ainsi, chaque fois qu'un message doit être transmis au bus, le connecteur consulte cette liste d'associations afin de connaître le nom du nœud sur lequel il doit publier son message.

Les types de messages supportés par un connecteur sont :
- `perf` : messages de performances
- `state` : messages d'état
- `event` : messages d'événements
- `nagios` : commandes Nagios
- `command` : commandes diverses

Si aucune destination n'est renseignée, le message sera envoyé sur un nœud du même nom que le type du message.

Exemple de configuration possible, correspondant à une installation standard :

```
[publications]
perf    = perf
state   = state
event   = event
nagios = nagios
```

Journaux

Le connecteur est capable de transmettre un certain nombre d'informations au cours de son fonctionnement à un mécanisme de journalisation des événements (par exemple, des journaux systèmes, une trace dans un fichier, un enregistrement des événements en base de données, etc.).

Le document Vigilo - Journaux d'événements décrit spécifiquement la configuration de la journalisation des événements au sein de toutes les applications de Vigilo, y compris les connecteurs.

Configuration spécifique au connecteur syncevents

Le connecteur syncevents dispose de quelques options de configuration spécifiques, détaillées ci-dessous.

Seuils pour l'envoi des demandes d'état

L'option `minutes_old` détermine l'âge minimum (en minutes) d'une alerte ou d'un état portant sur un hôte ou un service de bas niveau au-dessus duquel on demande une mise à jour à Nagios.

Nagios est configuré pour ré-émettre des notifications toutes les 30 minutes pour les hôtes et les services de bas niveau. Il faut donc régler ici une valeur légèrement supérieure, pour ne cibler que les états désynchronisés.

Une valeur négative désactive cette resynchronisation. La valeur par défaut est 45 minutes.

> **Avertissement :** Cette valeur doit être cohérente avec la fréquence des réémissions de notifications configurées dans Nagios pour les alertes sur les hôtes et les services de bas niveau.

L'option `hls_minutes_old` détermine l'âge minimum d'un état (en minutes) portant sur un service de haut niveau au-dessus duquel on demande une mise à jour à Nagios. Elle sert également à initialiser les services de haut niveau plus rapidement, en contrepartie d'une dégradation des performances générales de Vigilo.

Nagios est configuré pour ré-émettre des notifications toutes les 30 minutes pour les services de haut niveau. Il faut donc régler ici une valeur légèrement supérieure, pour ne cibler que les états désynchronisés.

Une valeur négative désactive cette resynchronisation. La valeur par défaut est -1 (ie. la resynchronisation est désactivée).

Avertissement : Si la resynchronisation des services de haut niveau est souhaitée, cette valeur doit être cohérente avec la fréquence des réémissions de notifications configurées dans Nagios pour les alertes sur les services de haut niveau.

Emplacement du fichier de verrou

Un fichier de verrou est créé par le connector-syncevents afin d'empêcher l'exécution simultanée de plusieurs instances du connecteur.

L'option `lockfile` peut être utilisée pour spécifier l'emplacement du fichier de verrou à créer. L'emplacement par défaut de ce fichier de verrou est `/var/lock/subsys/vigilo-connector-syncevents/lock`.

Limite sur le nombre de demandes d'état

L'option `max_events` permet de limiter le nombre de demandes de réémission d'état qui peuvent être envoyées à Nagios au cours d'une exécution du connecteur syncevents.

Cette option est particulièrement utile afin d'empêcher une inondation du bus de communication de Vigilo sur un parc de grande taille lorsqu'un incident majeur survient sur la plate-forme de supervision (par exemple : collecteur qui ne répond plus).

Si cette option n'est pas renseignée, aucune limite n'est imposée sur le nombre de demandes de réémission d'état qui peuvent être envoyées à Nagios au cours de la même exécution.

4.3.3 Utilisation

Lancement du composant

Le composant connector-syncevents est construit de sorte qu'une fois lancé, il liste les états qui semblent désynchronisés en base de données, et envoie des messages à Nagios pour qu'il ré-expédie les notifications sur ces états. Les données transmises sont précisées au chapitre .

S'il n'y a pas d'état à synchroniser, le connecteur s'arrête sans se connecter au bus. Le lancement se fait en exécutant la commande .

Activation régulière

Il est recommandé de planifier l'exécution de la commande à intervalles réguliers (par exemple, toutes les 10 minutes) afin d'effectuer des re-synchronisations assez fréquemment. Grâce à la limite d'âge configurée, seuls les événements qui semblent dé-synchronisés seront sélectionnés. Il y a donc peu d'impact à réaliser cette vérification assez fréquemment (seule une requête SQL est effectuée). Une bonne pratique consiste à exécuter périodiquement cette commande à l'aide d'un planificateur de tâches comme *cron*.

Le listing suivant donne un exemple de configuration utilisant *cron* :

```
# minutes heures n°jour mois jour commande > journal
# Exécution de la commande de synchronisation.
*/10 * * * * vigilo-syncevents /usr/bin/vigilo-connector-syncevents >/dev/null
```

Cette commande doit être lancée soit en tant que l'utilisateur « vigilo-syncevents », soit en tant que « root », pour avoir accès à son fichier settings.ini. Il est recommandé de ne pas utiliser « root », mais de lui préférer l'utilisateur dédié.

À l'installation, le connecteur installe cette tâche planifiée dans *cron*, mais la laisse désactivée. Pour l'activer, il suffit de dé-commenter l'entrée dans le fichier `/etc/cron.d/vigilo-connector-syncevents`.

Nature des informations transmises

Le connecteur syncevents envoie des messages contenant des commandes qui seront récupérées par le connector-nagios, puis transmises à Nagios après avoir été converties en utilisant la syntaxe adéquate.

Ce messages sont des demandes de ré-émission de notification, définis dans la documentation Nagios aux URL suivantes :

— SEND_CUSTOM_HOST_NOTIFICATION
— SEND_CUSTOM_SVC_NOTIFICATION

Pour circuler sur le bus, ces demandes sont encodées dans un message Vigilo de type « command », dont un exemple suit :

```
{ "type": "nagios",
  "cmdname": "SEND_CUSTOM_SVC_NOTIFICATION",
  "value": "server.example.com;Load 01;0;vigilo;syncevents",
}
```

Après réception par Nagios, ce dernier va ré-expédier une notification de l'état de l'hôte ou du service concerné, qui suivra le chemin classique des notifications dans Vigilo : elle sera réceptionnée par le corrélateur, qui mettra à jour l'état dans la base Vigilo.

4.3.4 Annexes

Glossaire - Terminologie

Ce chapitre recense les différents termes techniques employés dans ce document et donne une brève définition de chacun de ces termes.

AMQP Advanced Message Queuing Protocol. Protocole ouvert de messagerie applicative. Voir amqp.org.

exchange Nœud de publication dans le dialecte AMQP. Les messages publiés sur un *exchange* sont relayés à une ou plusieurs files d'attente sur le serveur, à partir desquelles les connecteurs vont consommer les messages.

inotify L'outil inotify est un mécanisme du noyau Linux qui fournit des notifications concernant le système de fichiers lorsqu'un événement particulier se produit (par exemple, la fermeture d'un fichier).

JSON JavaScript Object Notation. Méthode de sérialisation textuelle compatible JavaScript.

RRD Round-Robin Database. Base de données circulaire permettant de stocker des données disposant d'une granularité différente.

SGBD Système de Gestion de Base de Données. Logiciel permettant d'héberger une base de données sur la machine.

URL Uniform Resource Locator. Chaîne de caractères permettant d'identifier une ressource sur Internet.

XML eXtensible Markup Language. Langage de balisage extensible.

4.4 Guide d'administration

4.4.1 Installation

Pré-requis logiciels

Afin de pouvoir faire fonctionner VigiBoard, l'installation préalable des logiciels suivants est requise :

— python (>= 2.5), sur la machine où VigiBoard est installé
— Apache (>= 2.2.0), sur la machine où VigiBoard est installé
— mod_wsgi (>= 2.3), sur la machine où VigiBoard est installé
— PostgreSQL (>= 8.3), éventuellement sur une machine distante

4.4.2 Démarrage et arrêt

VigiBoard fonctionne comme un site Web standard. À ce titre, il n'est pas nécessaire d'exécuter une commande spécifique pour démarrer VigiBoard, dès lors que le serveur Web qui l'héberge a été lancé, à l'aide de la commande :

```
service httpd start
```

De la même manière, il n'y a pas de commande spécifique pour arrêter VigiBoard. L'application est arrêtée en même temps que le serveur Web, à l'aide de la commande :

```
service httpd stop
```

4.4.3 Configuration

La configuration initialement fournie avec VigiBoard est très rudimentaire. Elle est décomposée en deux fichiers :
— le fichier `settings.ini` d'une part, qui contient la majorité des options de configuration ;
— et le fichier `app_cfg.py` qui contient des options de configuration plus complexes, nécessitant l'utilisation d'un langage plus complet (Python).

Ce chapitre a pour but de présenter les différentes options de configuration disponibles afin de configurer VigiBoard en fonction de vos besoins. Les chapitres *Base de données* à *Configuration de l'auto-supervision* reprennent l'ordre de la configuration utilisé dans le fichier `settings.ini` de l'application. Toutes les options de configuration citées ici se trouvent sous la section `[app:main]` du fichier `settings.ini`.

Le chapitre *Options du fichier app_cfg.py* quant à lui décrit certaines options de configuration fournies par le fichier `app_cfg.py`.

Enfin, le chapitre *Intégration de VigiBoard avec Apache / mod_wsgi* donne des informations quant à la méthode utilisée pour intégrer VigiBoard sur un serveur Web de type Apache, grâce au module mod_wsgi.

La configuration de la journalisation des événements se fait également au travers du fichier `settings.ini`. Néanmoins, comme ce procédé se fait de la même manière dans les différents composants de Vigilo, celui-ci fait l'objet d'une documentation séparée dans le document *Vigilo - Journaux d'événements*.

Base de données

Pour fonctionner, VigiBoard nécessite qu'une base de données soit accessible. Ce chapitre décrit les options de configuration se rapportant à la base de données.

Connexion

La configuration de la connexion à la base de base de données se fait en modifiant la valeur de la clé `sqlalchemy.url` sous la section `[app:main]`.

Cette clé contient une URL qui contient tous les paramètres nécessaires pour pouvoir se connecter à la base de données. Le format de cette URL est le suivant :

```
sgbd://utilisateur:mot_de_passe@serveur:port/base_de_donnees
```

Le champ `:port` est optionnel et peut être omis si vous utilisez le port par défaut d'installation du SGBD choisi.

Par exemple, voici la valeur correspondant à une installation mono-poste par défaut de VigiBoard :

```
postgressql://vigilo:vigilo@localhost/vigilo
```

Avertissement : À l'heure actuelle, seul PostgreSQL a fait l'objet de tests intensifs. D'autres SGBD peuvent également fonctionner, mais aucun support ne sera fourni pour ces SGBD.

Optimisation de la couche d'abstraction

L'option `sqlalchemy.echo` permet de forcer l'affichage des requêtes SQL. En production, cette valeur doit être positionnée à `False`. Elle est redondante avec la configuration des journaux d'événements (voir le document intitulé Vigilo - Journaux d'événements pour plus d'information).

L'option `sqlalchemy.echo_pool` permet d'activer le mode de débogage du gestionnaire de connexions à la base de données. De même que pour l'option `sqlalchemy.echo` ci-dessus, elle doit être positionnée à `False` en production.

L'option `sqlalchemy.pool_recycle` permet de définir la durée après laquelle une connexion est « recyclée » (recréée).

L'option `sqlalchemy.pool_size` permet de configurer le nombre de connexions gérées simultanément par le gestionnaire de connexions à la base de données. La valeur recommandée est 20.

L'option `sqlalchemy.max_overflow` permet de limiter le nombre maximales de connexions simultanées à la base de données. La limite correspond à la somme de `sqlalchemy.pool_size` et `sqlalchemy.max_overflow`. Une valeur de 100 convient généralement.

La documentation d'API de SQLAlchemy (la bibliothèque d'abstraction de la base de données utilisée par Vigilo) donne quelques informations supplémentaires sur le rôle de ces différents paramètres. Cette documentation est accessible sur le site du projet.

Éléments de sécurité

Ce chapitre décrit les options relatives à la gestion des données de sécurité (clés de chiffrements, etc.) utilisées par VigiBoard.

Choix de la méthode de hachage des mots de passe

Lorsque l'authentification de Vigilo se base sur les comptes contenus dans la base de données, les mots de passe des utilisateurs sont stockés sous forme hachée afin de rendre plus difficile le cassage de ces mots de passe.

La méthode de hachage sélectionnée peut être configurée en modifiant la valeur de la clé `password_hashing_function` sous la section `[app:main]`. Les méthodes de hachage disponibles sont variées. Les fonctions de hachage suivantes sont notamment disponibles : md5, sha1, sha224, sha256, sha384 et sha512. D'autres fonctions peuvent être disponibles en fonction de votre installation de Python.

Avertissement : En cas d'absence d'une valeur pour cette option ou si la fonction de hachage indiquée n'existe pas, les mots de passe sont stockés en clair. Vérifiez donc la valeur indiquée.

Avertissement : Cette option ne doit être modifiée qu'au moment de l'installation. Si vous modifiez la méthode utilisée ultérieurement, les comptes précédemment enregistrés ne seront plus utilisables. En particulier, le compte d'administration créé par défaut.

Clé de chiffrement / déchiffrement des sessions

Afin de ne pas dévoiler certains paramètres associés à un utilisateur, le fichier de session qui contient ces paramètres est chiffré à l'aide d'une clé symétrique, utilisée à la fois pour le chiffrement et le déchiffrement des sessions de tous les utilisateurs de VigiBoard.

L'option `beaker.session.secret` permet de choisir la clé utilisée pour chiffrer et déchiffrer le contenu des sessions. Cette clé peut être la même que celle configurée pour le chiffrement / déchiffrement des cookies (voir le chapitre suivant), mais ceci est déconseillé afin d'éviter que la compromission de l'une des deux clés n'entraine la compromission de l'autre.

De la même manière, vous pouvez configurer les autres interfaces graphiques de Vigilo pour utiliser les mêmes clés, ou opter de préférence pour des clés différentes (là encore, pour éviter la propagation d'une compromission).

Clé de chiffrement / déchiffrement des cookies

L'association entre un utilisateur et sa session se fait à l'aide d'un cookie de session enregistré sur le navigateur de l'utilisateur. De la même manière que les sessions sont chiffrés afin de garantir la confidentialité de leur contenu, le cookie de session est également chiffré afin de protéger son contenu.

L'option `sa_auth.cookie_secret` permet de choisir la clé utilisée pour chiffrer et déchiffrer le contenu du cookie. Cette clé peut être la même que celle configurée pour le chiffrement / déchiffrement des sessions (voir le chapitre), mais ceci est déconseillé afin d'éviter que la compromission de l'une des deux clés n'entraine la compromission de l'autre.

De la même manière, vous pouvez configurer les autres interfaces graphiques de Vigilo pour utiliser les mêmes clés, ou opter de préférence pour des clés différentes (là encore, pour éviter la propagation d'une compromission).

Emplacement de la configuration d'authentification/autorisation

La directive `auth.config` de la section `[app:main]` permet d'indiquer l'emplacement du fichier contenant la configuration de la couche d'authentification/autorisation de Vigilo.

Il n'est généralement pas nécessaire de modifier cette valeur. La configuration de cette couche d'abstraction est détaillée dans le document *Vigilo - Authentification et autorisation*.

Configuration de l'interface

Ce chapitre décrit les options qui modifient l'apparence de l'interface graphique de VigiBoard.

Langue par défaut de VigiBoard

Au sein de son interface, VigiBoard tente de s'adapter au navigateur de l'utilisateur pour afficher les pages dans sa langue. Toutefois, si l'utilisateur n'a pas paramétré sa langue ou bien si aucune traduction n'est disponible qui soit en accord avec les paramètres du navigateur de l'utilisateur, une langue par défaut est utilisée (dans l'installation par défaut de VigiBoard, cette langue est le Français `fr`).

Vous pouvez modifier la langue utilisée par défaut en changeant la valeur de la clé `lang` sous la section `[app:main]`. La valeur de cette clé est le code de la langue à utiliser, sur deux caractères et en minuscules (format ISO 3166-1 `alpha 2`). Exemples de codes valides : fr, en, de, ...

La liste complète des codes possibles est disponible sur http://fr.wikipedia.org/wiki/ISO_3166-1. La langue retenue doit être disponible parmi les traductions fournies avec VigiBoard.

Emplacement de la documentation en ligne

Il est possible d'ajouter un lien dans l'interface graphique qui redirige l'utilisateur vers la documentation en ligne de l'application. Ceci se fait en assignant une URL à l'option `help_link`.

Si cette option est renseignée, une icône en forme de bouée de sauvetage  apparaît dans l'interface graphique qui permet à l'utilisateur d'accéder à l'URL indiquée.

Délai de rafraîchissement automatique

Le bac à événements de VigiBoard peut être actualisé automatiquement à intervalle régulier afin de donner une vue à jour de l'état du parc aux veilleurs. L'option `refresh_delay` permet de choisir le délai, en secondes, entre deux rafraîchissements automatiques de la page.

Note : Les veilleurs ont la possibilité de désactiver le rafraîchissement automatique durant leur session. Dans tous les cas, si une boîte de dialogue de VigiBoard est affichée à l'écran, le rafraîchissement automatique est mis en pause afin de ne pas perturber les opérations en cours.

État initial du rafraîchissement automatique

Vous avez la possibilité d'activer par défaut le rafraîchissement automatique du bac à événements pour les veilleurs, en positionnant l'option `refresh_enabled` à `True`.

Note : Les veilleurs ont la possibilité de désactiver le rafraîchissement automatique durant leur session. Leur choix (rafraîchissement automatique actif ou non) est conservé en session durant un certain temps.

Configuration du nombre d'événements affichés par page

Le nombre d'événements affichés par page peut être configuré en changeant la valeur de la clé `vigiboard_items_per_page` sous la section `[app:main]`.

Configuration du lien d'accueil

Vous avez la possibilité de rediriger l'utilisateur vers une page de votre choix lorsque celui-ci clique sur le logo en forme de maison  dans l'interface graphique de VigiBoard. Ceci se fait en modifiant l'URL associée à l'option `home_link`.

Ordre de tri de la priorité des événements

VigiBoard prend en compte la priorité des événements pour les triers dans son interface graphique. Néanmoins, chaque système à sa propre définition de la priorité d'un événement. Généralement, plus la priorité d'un événement est élevée, plus cet événement doit être traité en premier. Cependant il se peut que cet ordre de tri soit inversé sur votre parc (c'est-à-dire qu'un événement très prioritaire est représenté par une priorité dont la valeur est très basse).

L'ordre de tri de la priorité est défini grâce à la clé de configuration `vigiboard_priority_order`, sous la section `[app:main]`. Cette clé accepte deux valeurs : `asc` (nombre peu élevé = priorité importante) ou `desc` (nombre élevé = priorité importante).

Choix du critère de tri prioritaire

En fonction de votre parc informatique, il peut être intéressant de trier les événements reçus dans le bac à événements par état Nagios puis par horodatage, ou bien l'inverse.

L'option `state_first` est un booléen qui permet de choisir si le tri se fait d'abord par l'état (`True`), ou d'abord par l'horodatage (`False`).

Configuration de l'auto-supervision

VigiBoard affiche un message d'alerte à l'utilisateur dès lors qu'un des collecteurs Vigilo n'a pas donné signe de vie depuis plus d'une certaine durée. Cette durée-seuil, exprimée en secondes, est configurable à l'aide de l'option « freshness_threshold » . Une valeur négative ou nulle désactive complètement cette fonctionnalité.

Configuration du serveur mandataire

VigiBoard permet d'accéder à la page d'état Nagios d'un hôte ou d'un service, et ce malgré le fait que ces hôtes/services sont supervisés par des serveurs Nagios différents. Ceci est rendu possible par l'existence d'un serveur mandataire (proxy) qui relaye les requêtes au serveur Nagios concerné.

Le chapitre présente tout d'abord les options communes à tous les types de serveurs mandataires de Vigilo. Puis, le chapitre détaille les options spécifiques au serveur mandataire pour Nagios intégré à VigiBoard.

Options communes à tous les serveurs mandataires de Vigilo

Les options communes à tous les serveurs mandataires de Vigilo concernent l'authentification auprès d'un serveur mandataire intermédiaire. Elles sont au nombre de trois :

- `app_proxy_auth_method` indique la méthode d'authentification à utiliser et peut valoir `basic` ou `digest`
- `app_proxy_auth_username` indique le nom d'utilisateur à utiliser pour se connecter au serveur mandataire intermédiaire
- `app_proxy_auth_password` indique le mot de passe associé à ce nom d'utilisateur.

Ces trois options doivent être renseignées pour que l'authentification auprès du serveur mandataire intermédiaire soit effective.

Options spécifiques au serveur mandataire Nagios

L'option `app_path.nagios` indique l'emplacement de l'installation de Nagios sur le serveur Web distant, à partir de la racine du serveur Web. Généralement, il s'agit de `/nagios/` (emplacement par défaut lors d'une nouvelle installation de l'interface graphique CGI de Nagios).

L'option `app_scheme.nagios` indique le protocole à utiliser pour communiquer avec le serveur Web distant. Pour le moment, seuls les protocoles `http` et `https` sont supportés.

L'option `app_port.nagios` permet d'indiquer le port à utiliser pour se connecter, dans le cas où il ne s'agit pas du port standard. Par défaut, le serveur mandataire Nagios utilise le port standard associé au protocole donné par `app_scheme.nagios` (80 pour HTTP, 443 pour HTTPS).

L'option `app_redirect.nagios` permet de modifier le comportement du serveur mandataire. Lorsque cette option vaut `True`, le serveur mandataire agit comme un simple redirecteur de requêtes. Dans ce mode, les options d'authentification liées au serveur mandataire sont ignorées. Ce mode de fonctionnement est utile afin de tester la configuration mais n'est pas recommandé en production.

Les options `app_auth_method.nagios`, `app_auth_username.nagios` et `app_auth_password.nagios` permettent d'indiquer la méthode d'authentification, le nom d'utilisateur et le mot de passe pour accéder à l'interface CGI de Nagios. Ces options sont similaires à celles décrites au chapitre .

Configuration des sessions

Chaque fois qu'un utilisateur se connecte à VigiBoard, un fichier de session est créé permettant de sauvegarder certaines préférences de cet utilisateur (par exemple, le thème de l'application, la taille de la police de caractères, etc.).

Ce chapitre décrit les options relatives à la gestion des sessions.

Emplacement des fichiers de session

Le dossier dans lequel les fichiers de session seront stockés est indiqué par l'option `cache_dir`.

Nom du cookie de session

Afin d'associer un utilisateur au fichier de session qui lui correspond, un cookie de session est créé sur le navigateur de l'utilisateur. L'option `beaker.session.key` permet de choisir le nom du cookie créé. Le nom doit être composé de caractères alphanumériques (a-zA-Z0-9) et commencer par une lettre (a-zA-Z).

Options du fichier `app_cfg.py`

Le fichier `app_cfg.py` contient des réglages spécifiques à VigiBoard plus complexes à représenter que par l'usage du fichier `settings.ini`. Ce chapitre décrit ces réglages.

La modification de ces réglages nécessite une connaissance rudimentaire du langage de programmation Python.

Choix des colonnes affichées dans VigiBoard

Vous avez la possibilité de configurer les colonnes à afficher dans VigiBoard ainsi que leur ordre. VigiBoard est fourni avec un ensemble de colonnes prédéfinies. La liste complète des colonnes disponibles peut être obtenue à l'aide de la commande suivante :

```
vigilo-plugins vigiboard.columns
```

L'option `base_config['vigiboard_plugins']` du fichier `app_cfg.py` contient un tuple des noms des colonnes à afficher (dans leur ordre d'affichage, de gauche à droite sur un navigateur configuré pour un utilisateur français, et de droite à gauche pour un utilisateur hébreu).

Exemple de configuration possible :

```
base_config['vigiboard_plugins'] = (
    'details',
    'date',
    'priority',
    'occurrences',
    'hostname',
    'servicename',
    'output',
    'hls',
    'status',
)
```

Configuration des liens externes

L'option `base_config['vigiboard_links.eventdetails']` contient la liste des liens externes configurés, c'est-à-dire les liens qui seront affichés dans le dialogue de détail d'un événement (figure).

La configuration des liens externes est donnée sous la forme d'un tuple de tuples, de la forme

```
(libellé du lien, URL cible)
```

L'URL peut être relative ou absolue. Dans le cas d'une URL relative, celle-ci est relative à l'emplacement de la racine de VigiBoard sur le serveur Web.

L'URL peut contenir des paramètres qui seront transmis tel quel. De plus, les variables de substitution suivantes sont disponibles :

- `%(idcorrevent)d` est remplacé par l'identifiant (unique) de l'événement corrélé dans Vigilo,
- `%(host)s` est remplacé par le nom de l'hôte impacté par l'événement corrélé,
- `%(service)s` est remplacé par le nom du service impacté ou `None` si l'événement concernant directement l'hôte,
- `%(message)s` est remplacé par le message de supervision remonté par Nagios.

Exemple de configuration possible :

```
base_config['vigiboard_links.eventdetails'] = (
    (
        u'Détail de l\'hôte dans Nagios',
        '/nagios/%(host)s/cgi-bin/status.cgi?host=%(host)s'
    ), (
        u'Détail de la métrologie',
        'http://vigilo.example.com/vigigraph/rpc/fullHostPage?host=%(host)s'
    ), (
        u'Détail de la sécurité',
        'http://security.example.com/?host=%(host)s'
    ), (
        'Inventaire',
        'http://cmdb.example.com/?host=%(host)s'
    ), (
        'Documentation',
        'http://doc.example.com/?q=%(message)s'
    ),
)
```

Cet exemple correspond à la liste de liens suivante :

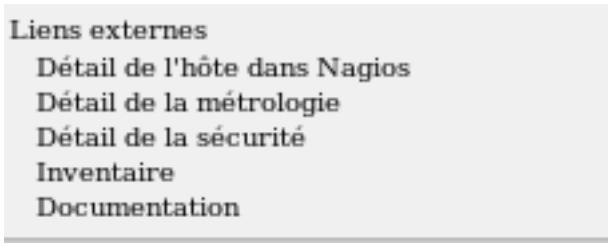


Fig. 4.1 – Liens externes d'un événement

Emplacement du gestionnaire de tickets

Un ticket d'incident peut être associé à un ou plusieurs événements corrélés apparaissant dans VigiBoard. L'adresse du gestionnaire de ticket est paramétrable à l'aide de l'option `base_config['vigiboard_links.tt']`.

Il s'agit d'une URL absolue, dans laquelle les variables de substitution suivantes sont disponibles :

- `%(idcorrevent)d` est remplacé par l'identifiant (unique) de l'événement corrélé dans Vigilo,
- `%(host)s` est remplacé par le nom de l'hôte impacté par l'événement corrélé,
- `%(service)s` est remplacé par le nom du service impacté ou `None` si l'événement concernant directement l'hôte,
- `%(tt)s` est remplacé par la référence du ticket d'incident, telle que saisie par un utilisateur.

Exemple de configuration possible :

```
base_config['vigiboard_links.tt'] = \
    'http://bugs.example.com/?ticket_id=%(tt)s'
```

Intégration de VigiBoard avec Apache / mod_wsgi

VigiBoard a été testé avec le serveur libre Apache. L'application utilise en outre le module Apache `mod_wsgi` pour communiquer avec le serveur. Ce module implémente un modèle de communication basé sur l'interface WSGI. Le reste de ce chapitre décrit la configuration utilisée pour réaliser cette intégration.

Fichier de configuration pour Apache

Le fichier de configuration pour l'intégration de VigiBoard dans Apache se trouve généralement dans `/etc/vigilo/vigiboard/vigiboard.conf` (un lien symbolique vers ce fichier est créé dans le dossier de configuration d'Apache, généralement dans `/etc/httpd/conf.d/vigiboard.conf`).

En général, il n'est pas nécessaire de modifier le contenu de ce fichier. Ce chapitre vise toutefois à fournir quelques informations sur le fonctionnement de ce fichier, afin de permettre d'éventuelles personnalisations de ce comportement.

Ce fichier tente tout d'abord de charger le module `mod_wsgi` (directive LoadModule) puis ajoute les directives de configuration nécessaire à Apache pour faire fonctionner VigiBoard, reprises partiellement ci-dessous :

```
WSGIRestrictStdout off
WSGIPassAuthorization on
WSGIDaemonProcess vigiboard user=apache group=apache threads=2
WSGIScriptAlias /vigilo/vigiboard "/etc/vigilo/vigiboard/vigiboard.wsgi"

KeepAlive Off

<Directory "/etc/vigilo/vigiboard/">
<Files "vigiboard.wsgi">
```

```
WSGIProcessGroup vigiboard
WSGIApplicationGroup %{GLOBAL}

Order deny,allow
Allow from all
</Files>
</Directory>
```

L'option `WSGIRestrictStdout` est positionnée à `off` afin d'éviter qu'Apache ne tue le processus de l'application lorsque des données sont envoyées sur la sortie standard. Ceci permet de récupérer les erreurs critiques pouvant être émises par l'application. Ces erreurs apparaissent alors dans le journal des événements d'Apache (configuré par la directive `error_log`).

L'option `WSGIPassAuthorization` positionnée à `on` indique à Apache et mod_wsgi que les informations d'authentification éventuellement transmises par l'utilisateur doivent être transmises à VigiBoard. En effet, Vigilo utilise son propre mécanisme de gestion de l'authentification et des autorisations (voir la documentation intitulée Vigilo - Authentification et autorisation) et utilise donc ces informations.

L'option `WSGIDaemonProcess` permet de créer un groupe de processus affecté au traitement des requêtes HTTP destinées à VigiBoard. Il permet d'utiliser un nom d'utilisateur et un groupe prédéfini (afin de réduire les privilèges nécessaires), ainsi que le nombre de processus légers à utiliser pour traiter les requêtes (ici, 2).

L'option `WSGIScriptAlias` indique l'emplacement à partir duquel VigiBoard sera accessible (ici, `http://example.com/vigilo/vigiboard` si le serveur Apache est configuré pour le domaine `example.com`) et l'emplacement du script WSGI nécessaire au lancement de l'application (voir le chapitre suivant).

L'option `KeepAlive` positionnée à `off` est nécessaire afin de contourner un problème dans le module `mod_wsgi` d'Apache.

Les autres options permettent d'exécuter le script WSGI de VigiBoard à l'aide du groupe de processus défini précédemment.

La liste complète des directives de configuration supportées par le module `mod_wsgi` d'Apache est disponible dans la documentation officielle.

Script WSGI de VigiBoard

Le script WSGI de VigiBoard est un script Python très simple qui a pour but de démarrer l'exécution de VigiBoard à partir du fichier de configuration associé (`/etc/vigilo/vigiboard/settings.ini`).

Vous n'avez généralement pas besoin de modifier son contenu, sauf éventuellement pour adapter l'emplacement du fichier de configuration en fonction de votre installation.

4.4.4 Annexes

Glossaire - Terminologie

Ce chapitre recense les différents termes techniques employés dans ce document et donne une brève définition de chacun de ces termes.

API (Application Programming Interface) Interface logicielle de programmation, permettant à un développeur d'enrichir la liste des fonctionnalités proposées par un logiciel.

CGI (Common Gateway Interface) Interface standard de communication entre un serveur web et un programme capable de générer une réponse HTTP valide. Il s'agit par exemple de l'interface retenue par Nagios (< 3.3) pour la génération de ses pages web.

CSS (Cascading Style Sheets) Feuilles de styles permettent de modifier la représentation graphique des éléments d'une page web. La version généralement supportée par les navigateurs est la version 2, définie par le document disponible sur http://www.w3.org/TR/CSS2/.

CSV (Comma-Separated Values) À l'origine, désigne un format textuel de transfert de données dans lequel les entrées sont séparées par des retours chariot et les champs par des virgules (comma). De nos jours, désigne plus généralement un format tabulé pour l'échange de données en vue de leur traitement dans un logiciel de type tableur ou par un traitement automatisé (scripts).

DN (Distinguished Name) Identifiant unique dans le cadre d'un annuaire LDAP.

Événement brut Alerte envoyée par Nagios au corrélateur de Vigilo pour analyse.

Événement corrélé Incident détecté par Vigilo suite à la corrélation des alertes Nagios (évènements bruts). Un événement corrélé est causé par un unique événement brut (exemple : la panne d'un routeur), mais de nombreux autres évènements bruts peuvent lui être rattachés (exemple : les alertes indiquant que les serveurs situés derrière le routeur en panne sont indisponibles). Ces événements secondaires rattachés à l'événement corrélé sont alors appelés « événements bruts masqués ».

KDC (Key Distribution Center) Serveur permettant un transfert sécurisé des clés de chiffrement utilisées pour les communications entre divers services. Ce serveur est notamment utilisé lors des échanges initiaux du protocole Kerberos.

LDAP (Lightweight Directory Access Protocol) Protocole pour l'interrogation d'un annuaire, servant généralement à recenser les utilisateurs autorisés d'un système et les différentes propriétés associées à ces utilisateurs.

OS (Operating System) Système d'exploitation.

Nagios Composant libre de supervision système et réseau.

RRD (Round Robin Database) Base de données de taille fixe utilisant des fichiers circulaires, dont les données sont progressivement compressées (avec perte) au fur et à mesure de leur vieillissement.

RRDtool Composant libre de gestion de bases RRD (stockage, restitution, génération de graphiques).

SGBD[R] (Serveur de Gestion de Bases de Données [Relationnelles]) Logiciel permettant d'héberger des bases de données [relationnelles] sur une machine.

SQL (Structured Query Language) Langage de requêtes structuré pour l'interrogation d'une base de données relationnelle.

URL (Uniform Resource Locator) Chaîne de caractères permettant d'identifier une ressource, par exemple une page web sur Internet. Exemple : http://www.vigilo-nms.com/

WSGI (Web Server Gateway Interface) Une interface pour la communication entre une application et un serveur web, similaire à CGI. Il s'agit de l'interface utilisée par Vigilo.

4.5 Guide d'administration

4.5.1 Installation

Prérequis logiciels

Afin de pouvoir faire fonctionner VigiGraph, l'installation préalable des logiciels suivants est requise :

— python (>= 2.5), sur la machine où VigiGraph est installé
— Apache (>= 2.2.0), sur la machine où VigiGraph est installé
— apache-mod_wsgi (>= 2.3), sur la machine où VigiGraph est installé
— postgresql-server (version 8.3), éventuellement sur une machine distante

Reportez-vous aux manuels de ces différents logiciels pour savoir comment procéder à leur installation sur votre machine. VigiGraph requiert également la présence de plusieurs dépendances Python. Ces dépendances seront automatiquement installées en même temps que le paquet de VigiGraph.

Installation du paquet RPM

L'installation de l'application se fait en installant simplement le paquet RPM « vigilo-vigigraph ». La procédure exacte d'installation dépend du gestionnaire de paquets utilisé. Les instructions suivantes décrivent la procédure pour les gestionnaires de paquets RPM les plus fréquemment rencontrés.

Installation à l'aide de urpmi :

```
urpmi vigilo-vigigraph
```

Installation à l'aide de yum :

```
yum install vigilo-vigigraph
```

4.5.2 Démarrage et arrêt de VigiGraph

VigiGraph fonctionne comme un site web standard. À ce titre, il n'est pas nécessaire d'exécuter une commande spécifique pour démarrer VigiGraph, dès lors que le serveur web qui l'héberge a été lancé, à l'aide de la commande :

```
service httpd start
```

De la même manière, il n'y a pas de commande spécifique pour arrêter VigiGraph. L'application est arrêtée en même temps que le serveur web, à l'aide de la commande : service httpd stop

4.5.3 Configuration de VigiGraph

La configuration initialement fournie avec VigiGraph est très rudimentaire. Elle est décomposée en deux fichiers :

- le fichier « settings.ini » d'une part, qui contient la majorité des options de configuration ;
- et le fichier « app_cfg.py » qui contient des options de configuration plus complexes, nécessitant l'utilisation d'un langage plus complet (Python).

Ce chapitre a pour but de présenter les différentes options de configuration disponibles afin de configurer VigiGraph en fonction de vos besoins. Les chapitres à reprennent l'ordre de la configuration utilisé dans le fichier « settings.ini » de l'application. Toutes les options de configuration citées ici se trouvent sous la section [app :main] du fichier « settings.ini ». Enfin, le chapitre donne des informations quant à la méthode utilisée pour intégrer VigiGraph sur un serveur web de type Apache, grâce au module mod_wsgi. La configuration de la journalisation des événements se fait également au travers du fichier « settings.ini ». Néanmoins, comme ce procédé est commun aux différents composants de Vigilo, celui-ci fait l'objet d'une documentation séparée dans le document *Vigilo – Journaux d'événements*.

Configuration de la base de données

Pour fonctionner, VigiGraph nécessite qu'une base de données soit accessible. Ce chapitre décrit les options de configuration se rapportant à la base de données.

Connexion à la base de données

La configuration de la connexion à la base de base de données se fait en modifiant la valeur de la clé « sqlalchemy.url » sous la section [app :main]. Cette clé contient une URL qui contient tous les paramètres nécessaires pour pouvoir se connecter à la base de données. Le format de cette URL est le suivant :

```
sgbd://nom_utilisateur:mot_de_passe@adresse_serveur:port_serveur/nom_base_de_donnees
```

Le champ « :port_serveur » est optionnel et peut être omis si vous utilisez le port par défaut d'installation du SGBD choisi.

Par exemple, voici la valeur correspondant à une installation mono-poste par défaut de VigiGraph :

postgresql ://vigilo :vigilo@localhost/vigilo

Avertissement : À l'heure actuelle, seul PostgreSQL a fait l'objet de tests intensifs. D'autres SGBD peuvent également fonctionner, mais aucun support ne sera fourni pour ces SGBD.

Optimisation de la couche d'abstraction

L'option « sqlalchemy.echo » permet de forcer l'affichage des requêtes SQL. En production, cette valeur doit être positionnée à « False ». Elle est redondante avec la configuration des journaux d'événements (voir le document intitulé Vigilo - Journaux d'événements pour plus d'information).

L'option « sqlalchemy.echo_pool » permet d'activer le mode de débogage du gestionnaire de connexions à la base de données. De même que pour l'option « sqlalchemy.echo » ci-dessus, elle doit être positionnée à « False » en production.

L'option « sqlalchemy.pool_recycle » permet de définir la durée après laquelle une connexion est « recyclée » (recréée).

L'option « sqlalchemy.pool_size » permet de configurer le nombre de connexions gérées simultanément par le gestionnaire de connexions à la base de données. La valeur recommandée est 20.

L'option « sqlalchemy.max_overflow » permet de limiter le nombre maximal de connexions simultanées à la base de données. La limite correspond à la somme de « sqlalchemy.pool_size » et « sqlalchemy.max_overflow ». Une valeur de 100 convient généralement.

La documentation d'API de SQLAlchemy (la bibliothèque d'abstraction de la base de données utilisée par Vigilo) donne quelques informations supplémentaires sur le rôle de ces différents paramètres. Cette documentation est accessible à l'adresse http://www.sqlalchemy.org/docs/05/reference/sqlalchemy/pooling.html.

Configuration des éléments de sécurité

Ce chapitre décrit les options relatives à la gestion des données de sécurité (clés de chiffrements, etc.) utilisées par VigiGraph.

Choix de la méthode de hachage des mots de passe

Lorsque l'authentification de Vigilo se base sur les comptes contenus dans la base de données, les mots de passe des utilisateurs sont stockés sous forme hachée afin de rendre plus difficile le cassage de ces mots de passe.

La méthode de hachage sélectionnée peut être configurée en modifiant la valeur de la clé « password_hashing_function » sous la section [app :main]. Les méthodes de hachage disponibles sont variées.

Les fonctions de hachage suivantes sont notamment disponibles :

- md5
- sha1
- sha224
- sha256
- sha384
- sha512

Des fonctions supplémentaires peuvent être disponibles en fonction de votre installation de Python.

Avertissement : En cas d'absence d'une valeur pour cette option ou si la fonction de hachage indiquée n'existe pas, les mots de passe sont stockés en clair. Vérifiez donc la valeur indiquée.

Avertissement : Cette option ne doit être modifiée qu'au moment de l'installation. Si vous modifiez la méthode utilisée ultérieurement, les comptes précédemment enregistrés ne seront plus utilisables. En particulier, le compte d'administration créé par défaut.

Clé de chiffrement / déchiffrement des sessions

Afin de ne pas dévoiler certains paramètres associés à un utilisateur, le fichier de session qui contient ces paramètres est chiffré à l'aide d'une clé symétrique, utilisée à la fois pour le chiffrement et le déchiffrement des sessions de tous les utilisateurs de VigiGraph.

L'option « beaker.session.secret » permet de choisir la clé utilisée pour chiffrer et déchiffrer le contenu des sessions. Cette clé peut être la même que celle configurée pour le chiffrement / déchiffrement des cookies (voir le chapitre), mais ceci est déconseillé afin d'éviter que la compromission de l'une des deux clés n'entraine la compromission de l'autre.

De la même manière, vous pouvez configurer les autres interfaces graphiques de Vigilo pour utiliser les mêmes clés, ou opter de préférence pour des clés différentes (là encore, pour éviter la propagation d'une compromission).

Clé de chiffrement / déchiffrement des cookies

L'association entre un utilisateur et sa session se fait à l'aide d'un cookie de session enregistré sur le navigateur de l'utilisateur.

De la même manière que les sessions sont chiffrées afin de garantir la confidentialité de leur contenu, le cookie de session est également chiffré afin de protéger son contenu.

L'option « sa_auth.cookie_secret » permet de choisir la clé utilisée pour chiffrer et déchiffrer le contenu du cookie. Cette clé peut être la même que celle configurée pour le chiffrement / déchiffrement des sessions (voir le chapitre), mais ceci est déconseillé afin d'éviter que la compromission de l'une des deux clés n'entraine la compromission de l'autre.

De la même manière, vous pouvez configurer les autres interfaces graphiques de Vigilo pour utiliser les mêmes clés, ou opter de préférence pour des clés différentes (là encore, pour éviter la propagation d'une compromission).

Utilisation d'un mécanisme d'authentification externe

Pour utiliser un mécanisme d'authentification externe (par exemple : Kerberos), définissez la clé « external_auth » à « True » sous la section [app :main]. Dans ce mode, Vigilo ne tente pas d'authentifier l'utilisateur par rapport aux comptes contenus dans sa base de données, mais utilise uniquement le nom d'utilisateur transmis par la source d'authentification externe.

Le nom d'utilisateur peut être transmis de plusieurs manières, par exemple sous la forme d'une variable d'environnement.

Avertissement : N'utilisez ce mode de fonctionnement que si vous comprenez bien les risques associés. En particulier, le fait qu'aucun mot de passe ne sera demandé à l'utilisateur pour confirmer son identité.

Emplacement de la configuration des authentifications / autorisations

La directive « auth.config » de la section [app :main] permet d'indiquer l'emplacement du fichier contenant la configuration de la couche d'authentification / autorisation de Vigilo.

Il n'est généralement pas nécessaire de modifier cette valeur. La configuration de cette couche d'abstraction est détaillée dans le document *Vigilo – Authentification et autorisation*.

Configuration de l'interface

Ce chapitre décrit les options qui modifient l'apparence de l'interface graphique de VigiGraph.

Langue par défaut de VigiGraph

Au sein de son interface, VigiGraph tente de s'adapter au navigateur de l'utilisateur pour afficher les pages dans sa langue.

Toutefois, si l'utilisateur n'a pas paramétré sa langue ou bien si aucune traduction n'est disponible qui soit en accord avec les paramètres du navigateur de l'utilisateur, une langue par défaut est utilisée (dans l'installation par défaut de VigiGraph, cette langue est le Français « fr »).

Vous pouvez modifier la langue utilisée par défaut en changeant la valeur de la clé « lang » sous la section [app :main]. La valeur de cette clé est le code de la langue à utiliser, sur deux caractères et en minuscules (format ISO 3166-1 « alpha 2 »). Exemples de codes valides : fr, en, de, ...

La liste complète des codes possibles est disponible sur http://fr.wikipedia.org/wiki/ISO_3166-1. La langue retenue doit être disponible parmi les traductions fournies avec VigiGraph.

Emplacement de la documentation en ligne

Il est possible d'ajouter un lien dans l'interface graphique qui redirige l'utilisateur vers la documentation en ligne de l'application. Ceci se fait en assignant une URL à l'option « help_link ».

Si cette option est renseignée, une icône en forme de bouée de sauvetage .. image : : img/help_icon.png

apparaît dans l'interface graphique qui permet à l'utilisateur d'accéder à l'URL indiquée.

Délai de rafraîchissement automatique

Les veilleurs ont la possibilité d'activer un rafraîchissement automatique des graphes. L'option « refresh_delay » permet de choisir le délai, en secondes, entre deux rafraîchissements automatiques d'un graphe.

Configuration du lien d'accueil

Vous avez la possibilité de rediriger l'utilisateur vers une page de votre choix lorsque celui-ci clique sur le logo en forme de maison  dans l'interface graphique de VigiGraph. Ceci se fait en modifiant l'URL associée à l'option `home_link`.

Configuration des serveurs mandataires

VigiGraph permet d'accéder à la page d'état Nagios d'un hôte ou d'un service, et ce malgré le fait que ces hôtes/services sont supervisés par des serveurs Nagios différents. De même, il est capable d'afficher des graphes de métrologie en interrogeant le serveur VigiRRD qui gère un hôte donné. Ceci est rendu possible par l'existence d'un serveur mandataire (proxy) au sein de Vigilo qui relaye les requêtes au serveur Nagios ou VigiRRD adéquat. Le chapitre présente tout d'abord les options communes à tous les types de serveurs mandataires de Vigilo. Puis, le chapitre détaille les options spécifiques au serveur mandataire pour Nagios intégré dans VigiGraph. Enfin, le chapitre détaille les options spécifiques au serveur mandataire pour VigiRRD, dont la configuration est très proche de celle du serveur mandataire pour Nagios.

Options communes à tous les serveurs mandataires de Vigilo

Les options communes à tous les serveurs mandataires de Vigilo concernent l'authentification auprès d'un serveur mandataire intermédiaire.

Elles sont au nombre de trois :

- « app_proxy_auth_method » indique la méthode d'authentification à utiliser et peut valoir « basic » ou « digest » ,
- « app_proxy_auth_username » indique le nom d'utilisateur à utiliser pour se connecter au serveur mandataire intermédiaire ,
- « app_proxy_auth_password » indique le mot de passe associé à ce nom d'utilisateur.

Ces trois options doivent être renseignées pour que l'authentification auprès du serveur mandataire intermédiaire soit effective.

Options spécifiques au serveur mandataire Nagios

L'option « app_path.nagios » indique l'emplacement de l'installation de Nagios sur le serveur web distant, à partir de la racine du serveur web. Généralement, il s'agit de « /nagios/ » (emplacement par défaut lors d'une nouvelle installation de l'interface graphique CGI de Nagios).

L'option « app_scheme.nagios » indique le protocole à utiliser pour communiquer avec le serveur web distant. Les protocoles supportés sont « http » et « https ».

L'option « app_port.nagios » permet d'indiquer le port à utiliser pour se connecter, dans le cas où il ne s'agit pas du port standard. Par défaut, le serveur mandataire Nagios utilise le port standard associé au protocole donné par « app_scheme.nagios » (80 pour HTTP, 443 pour HTTPS).

L'option « app_redirect.nagios » permet de modifier le comportement du serveur mandataire. Lorsque cette option vaut « True », le serveur mandataire agit comme un simple redirecteur de requêtes. Dans ce mode, les options d'authentification liées au serveur mandataire sont ignorées. Ce mode de fonctionnement est utile afin de tester la configuration mais n'est pas recommandé en production.

Les options « app_auth_method.nagios », « app_auth_username.nagios » et « app_auth_password.nagios » permettent d'indiquer la méthode d'authentification, le nom d'utilisateur et le mot de passe pour accéder à l'interface CGI de Nagios. Ces options sont similaires à celles décrites au chapitre .

Options spécifiques au serveur mandataire VigiRRD

Les options de configuration pour le serveur mandataire VigiRRD sont exactement les mêmes que pour le serveur mandataire Nagios présenté au chapitre , en remplaçant simplement le suffixe « .nagios » par « .vigirrd ».

Configuration des sessions

Chaque fois qu'un utilisateur se connecte à VigiGraph, un fichier de session est créé permettant de sauvegarder certaines préférences de cet utilisateur (par exemple, le thème de l'application, la taille de la police de caractères, etc.). Ce chapitre décrit les options relatives à la gestion des sessions.

Emplacement des fichiers de session

Le dossier dans lequel les fichiers de session seront stockés est indiqué par l'option « cache_dir ».

Nom du cookie de session

Afin d'associer un utilisateur au fichier de session qui lui correspond, un cookie de session est créé sur le navigateur de l'utilisateur.

L'option « beaker.session.key » permet de choisir le nom du cookie créé. Le nom doit être composé de caractères alphanumériques (a-zA-Z0-9) et commencer par une lettre (a-zA-Z).

4.5.4 Intégration de VigiGraph avec Apache / mod_wsgi

VigiGraph a été testé avec le serveur libre Apache. L'application utilise en outre le module Apache « mod_wsgi » pour communiquer avec le serveur. Ce module implémente un modèle de communication basé sur l'interface WSGI. Le reste de ce chapitre décrit la configuration utilisée pour réaliser cette intégration.

Fichier de configuration pour Apache

Le fichier de configuration pour l'intégration de VigiGraph dans Apache se trouve généralement dans /etc/vigilo/vigigraph/vigigraph.conf (un lien symbolique vers ce fichier est créé dans le dossier de configuration d'Apache, généralement dans /etc/httpd/conf.d/vigigraph.conf).

En général, il n'est pas nécessaire de modifier le contenu de ce fichier. Ce chapitre vise toutefois à fournir quelques informations sur le fonctionnement de ce fichier, afin de permettre d'éventuelles personnalisations de ce comportement.

Ce fichier tente tout d'abord de charger le module « mod_wsgi » (directive LoadModule) puis ajoute les directives de configuration nécessaires à Apache pour faire fonctionner VigiGraph, reprises partiellement ci-dessous :

```
WSGIRestrictStdout off
WSGIPassAuthorization on
WSGIDaemonProcess vigigraph user=apache group=apache threads=2
WSGIScriptAlias /vigilo/vigigraph "/etc/vigilo/vigigraph/vigigraph.wsgi"

KeepAlive Off

<Directory "/etc/vigilo/vigigraph/">
<Files "vigigraph.wsgi">
WSGIProcessGroup vigigraph
WSGIApplicationGroup %{GLOBAL}
```

```
Order deny,allow
Allow from all
</Files>
</Directory>
```

L'option WSGIRestrictStdout est positionnée à « off » afin d'éviter qu'Apache ne tue le processus de l'application lorsque des données sont envoyées sur la sortie standard. Ceci permet de récupérer les erreurs critiques pouvant être émises par l'application. Ces erreurs apparaissent alors dans le journal des événements d'Apache (configuré par la directive error_log).

L'option WSGIPassAuthorization positionnée à « on » indique à Apache et mod_wsgi que les informations d'authentification éventuellement transmises par l'utilisateur doivent être transmises à VigiGraph. En effet, Vigilo utilise son propre mécanisme de gestion de l'authentification et des autorisations (voir la documentation intitulée Vigilo - Authentification et autorisation) et utilise donc ces informations.

L'option WSGIDaemonProcess permet de créer un groupe de processus affecté au traitement des requêtes HTTP destinées à VigiGraph. Il permet d'utiliser un nom d'utilisateur et un groupe prédéfini (afin de réduire les privilèges nécessaires), ainsi que le nombre de processus légers à utiliser pour traiter les requêtes (ici, 2).

L'option WSGIScriptAlias indique l'emplacement à partir duquel VigiGraph sera accessible (ici, http://example.com/vigilo/vigigraph si le serveur Apache est configuré pour le domaine « example.com ») et l'emplacement du script WSGI nécessaire au lancement de l'application (voir le chapitre).

L'option KeepAlive positionnée à « off » est nécessaire afin de contourner un problème dans le module « mod_wsgi » d'Apache.

Les autres options permettent d'exécuter le script WSGI de VigiGraph à l'aide du groupe de processus défini précédemment.

La liste complète des directives de configuration supportées par le module « mod_wsgi » d'Apache est disponible à l'adresse http://code.google.com/p/modwsgi/wiki/ConfigurationDirectives.

Script WSGI de VigiGraph

Le script WSGI de VigiGraph est un script Python très simple qui a pour but de démarrer l'exécution de VigiGraph à partir du fichier de configuration associé (/etc/vigilo/vigigraph/settings.ini).

Vous n'avez généralement pas besoin de modifier son contenu, sauf éventuellement pour adapter l'emplacement du fichier de configuration en fonction de votre installation.

4.5.5 Annexes

Glossaire - Terminologie

Ce chapitre recense les différents termes techniques employés dans ce document et donne une brève définition de chacun de ces termes.

API Interface logicielle de programmation, permettant à un développeur d'enrichir la liste des fonctionnalités proposées par un logiciel.

CGI Interface standard de communication entre un serveur web et un programme capable de générer une réponse HTTP valide. Il s'agit par exemple de l'interface retenue par Nagios (< 3.3) pour la génération de ses pages web.

CSS Feuilles de styles permettent de modifier la représentation graphique des éléments d'une page web. La version généralement supportée par les navigateurs est la version 2, définie par le document disponible sur http://www.w3.org/TR/CSS2/.

CSV À l'origine, désigne un format textuel de transfert de données dans lequel les entrées sont séparées par des retours chariot et les champs par des virgules (comma). De nos jours, désigne plus généralement un format tabulé pour l'échange de données en vue de leur traitement dans un logiciel de type tableur ou par un traitement automatisé (scripts).

DN Identifiant unique dans le cadre d'un annuaire LDAP.

Événement brut Alerte envoyée par Nagios au corrélateur de Vigilo pour analyse.

Événement corrélé Incident détecté par Vigilo suite à la corrélation des alertes Nagios (évènements bruts). Un événement corrélé est causé par un unique événement brut (exemple : la panne d'un routeur), mais de nombreux autres évènements bruts peuvent lui être rattachés (exemple : les alertes indiquant que les serveurs situés derrière le routeur en panne sont indisponibles). Ces événements secondaires rattachés à l'événement corrélé sont alors appelés « événements bruts masqués ».

KDC Serveur permettant un transfert sécurisé des clés de chiffrement utilisées pour les communications entre divers services. Ce serveur est notamment utilisé lors des échanges initiaux du protocole Kerberos.

LDAP Protocole pour l'interrogation d'un annuaire, servant généralement à recenser les utilisateurs autorisés d'un système et les différentes propriétés associées à ces utilisateurs.

OS Système d'exploitation.

Nagios Composant libre de supervision système et réseau.

RRD Base de données de taille fixe utilisant des fichiers circulaires, dont les données sont progressivement compressées (avec perte) au fur et à mesure de leur vieillissement.

RRDtool Composant libre de gestion de bases RRD (stockage, restitution, génération de graphiques).

SGBD[R] Logiciel permettant d'héberger des bases de données [relationnelles] sur une machine.

SQL Langage de requêtes structuré pour l'interrogation d'une base de données relationnelle.

URL Chaîne de caractères permettant d'identifier une ressource, par exemple une page web sur Internet. Exemple : http://www.vigilo-nms.com/

WSGI Une interface pour la communication entre une application et un serveur web, similaire à CGI. Il s'agit de l'interface utilisée par Vigilo.

4.6 Authentification et autorisations

4.6.1 Installation

Pré-requis logiciels

La configuration de l'authentification dans Vigilo nécessite l'installation des logiciels suivants :

— python (>= 2.5), sur la machine à configurer
— `vigilo-turbogears`, installé automatiquement par les interfaces web de Vigilo

Reportez-vous aux manuels de ces différents logiciels pour savoir comment procéder à leur installation sur votre machine.

Le paquet `vigilo-turbogears` requiert également la présence de plusieurs dépendances Python. Ces dépendances seront automatiquement installées en même temps que le paquet `vigilo-turbogears`.

4.6.2 Comptes par défaut

Un compte par défaut est créé lors de l'installation de Vigilo, appelé « manager ». Ce compte correspond à un utilisateur disposant de tous les droits (super-utilisateur). Il n'a pas vocation à être utilisé en cours d'exploitation, mais plutôt à permettre des interventions ponctuelles liées à des tâches d'administration.

Le mot de passe par défaut de ce compte est `iddad`. Ce mot de passe peut être modifié depuis une machine sur laquelle Vigilo est installé, via la ligne de commandes en exécutant la commande vigilo-passwd, à partir du compte de l'utilisateur `root`.

4.6.3 Phases de l'authentification

Le mécanisme d'authentification de Vigilo repose sur le framework `repoze.who`. La gestion des autorisations repose sur le framework `repoze.what`.

Les interfaces graphiques de Vigilo utilisent les prédicats définies dans `repoze.what` afin de restreindre l'accès à certaines fonctionnalités. Ces prédicats peuvent nécessiter que l'utilisateur soit authentifié, appartienne à un groupe d'utilisateurs particulier ou dispose d'une permission spéciale. Ce document ne revient pas sur ce mécanisme de gestion des autorisations mais insiste plutôt sur l'authentification de l'utilisateur, qui permet d'établir les groupes auxquels il appartient et les permissions dont il dispose. Le lecteur pourra consulter la documentation du projet `repoze.what` pour plus d'information sur les mécanismes d'autorisation consulter la documentation de Repoze. L'authentification se déroule en plusieurs phases :

- phase de classification, qui permet de savoir d'où provient la requête en cours de traitement (par exemple, provient-elle d'un navigateur web ou s'agit-il d'une requête interne à Vigilo ?),
- phase d'identification, chargée de reconnaître l'utilisateur lorsqu'il se connecte,
- phase d'authentification, chargée de vérifier l'identité de l'utilisateur,
- phase de « challenge », permettant de demander des informations à l'utilisateur afin de l'authentifier,
- phase d'enrichissement à l'aide de fournisseurs de méta-données, permettant d'associer des méta-données à la session de l'utilisateur authentifié.

4.6.4 Fichier de configuration

La configuration de l'authentification se fait en créant un fichier INI. Ce fichier INI utilise le format suivant :

```
[plugin:foo]
use = bar.baz.foo:FooPlugin
param_foo_1 = valeur1
param_foo_2 = valeur2

[general]
request_classifier = bar.baz:Classifier
challenge_decider = bar.baz:Challenger

[identifiers]
plugins =
    foo

[authenticators]
plugins =
    foo

[challengers]
plugins =
    foo

[mdproviders]
plugins =
    foo
```

La documentation du framework `repoze.who` contient de plus amples informations.

Lien avec les phases de l'authentification

Le contenu du fichier de configuration suit de près les différentes phases de l'authentification. Tout d'abord, des modules sont configurés, à l'aide de sections `plugin:`, puis ceux-ci sont associés aux différentes étapes de l'authentification à l'aide des sections `identifiers` (identification), `authenticators` (authentification), `challengers` (challenge) et `mdproviders` (fournisseurs de méta-données).

La section `general` contient la configuration pour l'étape de classification, ainsi que des paramètres généraux, qui ne sont pas liés aux différentes étapes.

La suite de ce document décrit plus en détail le contenu des différentes sections du fichier de configuration.

Sections `plugin:`

Les sections dont le nom commence par `plugin:` correspondent à la configuration d'un module qui pourra être utilisé dans le cadre de la configuration de l'authentification. Le libellé après les deux points « : » correspond à un alias qui sera affecté au plugin.

La clé `use` permet de spécifier l'emplacement de la fonction ou de la classe Python implémentant le module. L'emplacement du module est donné en utilisant la syntaxe : `module.python:ClassOuFonction`. Les autres clés de la section correspondent aux différents paramètres attendus par le module.

Tous les modules définis par défaut dans *repoze.who* sont utilisables ici [1].

Vigilo fournit également le module d'identification et d'authentification `repoze.who.plugins.vigilo.kerberos:VigiloKerberosAuthenticator` permettant d'utiliser un annuaire LDAP et la méthode Kerberos pour l'authentification des utilisateurs.

Sections `identifiers`, `authenticators`, `challengers` et `mdproviders`

Les sections `identifiers`, `authenticators`, `challengers` et `mdproviders` permettent de définir les modules à utiliser au cours des phases décrites au chapitre *Phases de l'authentification*.

Chacune de ces sections ne contient qu'une seule clé, appelée `plugins`, qui contient la liste des modules à appeler, à l'aide des alias définis pour ces modules lors de leur configuration (voir le chapitre *Fichier de configuration*).

La liste des modules doit être écrites à la ligne, indentée, avec un module par ligne. Il est possible de n'appliquer un module que pour une classification particulière (voir la fonction `request_classifier` décrite dans la *Section general* du chapitre *Fichier de configuration*) en suffixant le nom du module par « ; » suivi de la classification pour laquelle il agit.

Exemple de configuration possible pour les `challengers` :

```
[challengers]
plugins =
    friendlyform;browser
    basicauth;vigilo-api
```

Ici, le module `friendlyform` est appelé lorsque la fonction de classification attribue la classification `browser` à la requête, tandis qu'elle appelle le module `basicauth` lorsque la classification renvoyée est `vigilo-api`.

Section `general`

La section `general` contient deux options :

[1] La liste complète des modules est disponible à l'adresse : http://docs.repoze.org/who/1.0/narr.html#module-repoze.who.plugins.sql

— `request_classifier` permet de classer les requêtes (par exemple, selon leur origine). Il s'agit d'une fonction qui sera appelée à chaque requête et renvoie une chaîne de caractères décrivant la classification de la requête. Cette fonction est indiquée en utilisant la même syntaxe que pour la définition des modules, à savoir : `module.python:ClasseOuFonction`.
Vigilo fournit la fonction `vigilo.turbogears.repoze_plugins:vigilo_api_classifier` qui permet de distinguer les requêtes provenant d'un navigateur web des requêtes provenant de l'API interne de Vigilo.
— `challenge_decider` permet de définir une fonction qui sera appelée pour décider si la requête actuelle nécessite d'obtenir de plus amples informations sur l'utilisateur afin de pouvoir l'authentifier. Si la réponse est positive, alors les plugins définis dans la section `challengers` sont utilisés.
Vigilo utilise le module de challenge standard de `repoze.who` (implémenté par la fonction `repoze.who.classifiers:default_challenge_decider`) pour décider si des informations supplémentaires sont nécessaires au traitement de la demande d'authentification.

4.6.5 Authentification externe

Dans le cas où l'authentification doit se faire en utilisant une source externe (ex : annuaire LDAP), la configuration du fichier de gestion de l'authentification (who.ini) doit être adaptée, ainsi que la configuration des diverses interfaces graphiques de Vigilo.

Dans ce chapitre, nous allons mettre en place une solution d'authentification unique (Single Sign-On) basée sur l'utilisation de la méthode d'authentification Kerberos auprès d'un annuaire LDAP.

On suppose que l'infrastructure nécessaire est déjà en place (un annuaire LDAP, un KDC et éventuellement une PKI). On suppose également que l'annuaire LDAP en place contient des informations sur les autorisations accordées aux différents utilisateurs (par exemple, la liste des groupes auxquels un utilisateur à accès, et donc les éléments du parc ou les applications qu'il est susceptible de consulter).

Le chapitre *Configuration d'Apache avec Kerberos* donne un exemple de configuration du module mod_auth_kerb d'Apache, permettant d'authentifier les utilisateurs grâce à la méthode Kerberos.

Le chapitre *Adaptation du fichier who.ini* décrit les modifications apportées au fichier who.ini afin d'utiliser le ticket Kerberos transmis par Apache au sein des interfaces graphiques de Vigilo. Ce ticket sera notamment utilisé pour interroger un annuaire LDAP et obtenir des informations sur l'utilisateur actuellement connecté (nom complet, adresse email, groupes dont il est membre).

Enfin, le chapitre *Configuration du navigateur web des exploitants* décrit la configuration à apporter au sein du navigateur web afin de permettre l'utilisation de Kerberos comme méthode d'authentification.

Configuration d'Apache avec Kerberos

Afin d'utiliser Kerberos comme méthode d'authentification, le module `mod_auth_kerb` d'Apache doit être configuré afin de pouvoir décoder le ticket Kerberos transmis par le navigateur web des utilisateurs (voir aussi le chapitre *Configuration du navigateur web des exploitants* pour la configuration à apporter dans le navigateur web).

Le listing suivant montre comment charger le module `mod_auth_kerb` dans Apache pour activer le support de Kerberos :

```
1 <IfModule !mod_auth_kerb.c>
2     LoadModule auth_kerb_module extramodules/mod_auth_kerb.so
3 </IfModule>
```

Le chargement du module n'est fait que s'il n'était pas déjà chargé (cette vérification est faite grâce à l'encapsulation dans la directive IfModule à la ligne 1). La directive `LoadModule` à la ligne 2 donne le nom du point d'entrée à charger dans le module (`auth_kerb_module` dans le cas du module mod_auth_kerb), ainsi que l'emplacement du module. Le module peut être installé à un autre endroit sur la machine, en fonction de la distribution Linux utilisée.

Une fois le module chargé, il faut adapter la configuration Apache des différentes applications (fichiers « vigiboard.conf », « vigimap.conf » et « vigigraph.conf » du répertoire /etc/httpd/conf.d/).

Le listing suivant donne un exemple de configuration de VigiBoard dans Apache afin de gérer l'authentification Kerberos. Ce fichier se trouve dans /etc/httpd/conf.d/vigiboard.conf :

```
1  <IfModule mod_wsgi.c>
2
3      WSGISocketPrefix /var/run/wsgi
4      WSGIRestrictStdout off
5      WSGIPassAuthorization on
6      WSGIDaemonProcess vigiboard user=apache group=apache processes=4 threads=1
7      WSGIScriptAlias /vigilo/vigiboard "/etc/vigilo/vigiboard/vigiboard.wsgi"
8
9      KeepAlive Off
10
11     <Directory "/etc/vigilo/vigiboard/">
12         <IfModule mod_headers.c>
13             Header set X-UA-Compatible "IE=edge"
14         </IfModule>
15
16         <Files "vigiboard.wsgi">
17             WSGIProcessGroup vigiboard
18             WSGIApplicationGroup %{GLOBAL}
19             <IfModule mod_authz_core.c>
20                 # Apache 2.4
21                 Require all granted
22             </IfModule>
23             <IfModule !mod_authz_core.c>
24                 # Apache 2.2
25                 Order Deny,Allow
26                 Allow from all
27             </IfModule>
28         </Files>
29     </Directory>
30
31     <Location "/vigilo/vigiboard/login">
32         AuthType kerberos
33         AuthName "Kerberos"
34         KrbServiceName HTTP
35         KrbAuthRealms EXAMPLE.COM
36         Krb5Keytab /etc/httpd/conf/HTTP.vigilo.example.com.keytab
37         KrbMethodNegotiate on
38         KrbMethodK5Passwd off
39         KrbSaveCredentials on
40         KrbVerifyKDC on
41         Require valid-user
42     </Location>
43
44 </IfModule>
```

Avec cette configuration, seule l'URL `http://vigilo.example.com/vigilo/vigiboard/login` est protégée par une authentification Kerberos. Les autres pages redirigent vers celle-ci lorsqu'un utilisateur authentifié est attendu et que l'utilisateur courant ne l'est pas. Cette solution offre le meilleur compromis possible entre la sécurité (il n'est pas possible d'accéder à une ressource protégée sans être authentifié) et les performances (une seule authentification par session).

La ligne 21 indique qu'Apache doit procéder à une authentification de type « kerberos » afin d'autoriser l'accès à l'application (directive `AuthType`).

La ligne 22 permet d'associer un nom à cette méthode d'authentification (directive `AuthName`). Ce nom apparaîtra dans les journaux d'événements du serveur.

La ligne 23 spécifie le nom du service Kerberos qui sera utilisé pour procéder à l'authentification (directive `KrbServiceName`). La valeur par défaut est « HTTP » qui correspond à la valeur recommandée.

La ligne 24 indique le nom du domaine Kerberos dans lequel l'authentification a lieu (directive `KrbAuthRealms`). Par convention, il s'agit du nom de domaine du parc, **en majuscules**.

La ligne 25 spécifie l'emplacement du fichier contenant la clé secrète d'authentification de ce service (directive `Krb5Keytab`). Ce fichier doit être accessible par le serveur web (et uniquement celui-ci).

La directive `KrbMethodNegotiate` à la ligne 26 autorise la négociation de la méthode d'authentification entre le navigateur et le serveur web. Il est recommandé d'autoriser la négociation.

La ligne 27 désactive l'authentification à la volée par identifiant/mot de passe (directive `KrbMethodK5Password`). Cette directive peut être positionnée à « on » pour autoriser les utilisateurs à s'authentifier à la volée auprès du serveur web. Si l'utilisateur tente de se connecter à l'application alors qu'il ne dispose pas d'un ticket Kerberos valide, une boîte de dialogue l'invite à saisir son identifiant et son mot de passe. La suite du processus d'authentification se déroule alors comme si un ticket Kerberos avait été transmis. Dans un environnement configuré pour n'utiliser que Kerberos (et ce dès l'ouverture d'une session au démarrage des postes utilisateurs), il est conseillé de positionner cette directive à « off ». Dans les autres cas, il est recommandé de positionner cette directive à « on » pour permettre aux utilisateurs ne disposant pas des outils nécessaires sur leur machine de pouvoir s'authentifier malgré tout.

La directive `KrbSaveCredentials` à la ligne 28 permet de sauvegarder temporairement le ticket Kerberos de l'utilisateur afin de permettre à l'application d'interroger d'autres services en utilisant la méthode Kerberos. Cette option est nécessaire dans les interfaces graphiques lorsque l'accès à Nagios se fait via une authentification Kerberos, afin de propager le ticket Kerberos reçu et maintenir la traçabilité des accès. Le fichier contenant le ticket Kerberos est supprimé automatiquement à la fin de la requête.

La directive `KrbVerifyKdc` à la ligne 29 désactive la vérification de l'identité du KDC du parc. Pour plus de sécurité, il est recommandé de positionner cette directive à la valeur « on ». L'activation de cette option nécessite cependant une configuration plus avancée de l'infrastructure Kerberos, qui dépasse le cadre de ce document.

La directive `Require` (ligne 32) indique que l'utilisateur doit dispose d'un compte valide dans la base Kerberos pour pouvoir accéder à l'application.

Enfin, la directin `Allow` (ligne 33) indique que n'importe quel machine est autorisée à se connecter à l'application, quelle que soit son adresse IP.

Adaptation du fichier who.ini

La prise en charge de Kerberos comme méthode d'authentification dans Vigilo se fait en paramétrant le fichier « who.ini » des interfaces graphiques (VigiMap, VigiGraph et VigiBoard), selon la méthode présentée dans ce chapitre. Lorsque Kerberos est utilisé comme méthode d'authentification, l'identifiant Kerberos de l'utilisateur est transmis à l'application au travers de la variable CGI `REMOTE_USER`.

Le listing ci-dessous présente un exemple complet de configuration permettant de synchroniser les comptes utilisateurs dans Vigilo avec un annuaire LDAP externe, tout en utilisant l'identité Kerberos obtenue depuis le serveur web :

```
1  [plugin:auth_tkt]
2  use = repoze.who.plugins.auth_tkt:make_plugin
3  secret = vigilo
4  cookie_name = authtkt
5
6  [plugin:basicauth]
7  use = repoze.who.plugins.basicauth:make_plugin
8  realm=Vigilo
9
```

```
10 [plugin:friendlyform]
11 use = repoze.who.plugins.friendlyform:FriendlyFormPlugin
12 login_form_url= /login
13 login_handler_path = /login_handler
14 logout_handler_path = /logout_handler
15 rememberer_name = auth_tkt
16 post_login_url = /post_login
17 post_logout_url = /post_logout
18
19 [plugin:ldapsync]
20 use = vigilo.turbogears.repoze.plugins.mdldapsync:VigiloLdapSync
21 ldap_url = ldap://ldap.example.com
22 ldap_base = ou=people,dc=example,dc.com
23 filterstr= (&(uid=%s)(objectClass=*))
24 ldap_charset = cp1252
25 http_charset = utf-8
26 cache_name = vigilo
27 binddn = mybinduser
28 bindpw = mybindpassword
29 attr_cn = cn
30 attr_mail = mail
31 attr_member_of = memberOf
32 timeout = 3
33
34 [plugin:externalid]
35 use = vigilo.turbogears.repoze.plugins.externalid:ExternalIdentification
36 cache_name = vigilo
37
38 [general]
39 request_classifier = vigilo.turbogears.repoze.classifier:vigilo_classifier
40 challenge_decider = repoze.who.classifiers:default_challenge_decider
41
42 [identifiers]
43 plugins =
44     friendlyform;browser
45     basicauth;vigilo-api
46     auth_tkt
47     externalid;browser
48
49 [authenticators]
50 plugins =
51     vigilo.turbogears.repoze.plugins.sqlauth:plugin
52
53 [challengers]
54 plugins =
55     friendlyform;browser
56     basicauth;vigilo-api
57
58 [mdproviders]
59 plugins =
60     ldapsync
61     vigilo.turbogears.repoze.plugins.mduser:plugin
62     vigilo.turbogears.repoze.plugins.mdgroups:plugin
```

Le module `ldapsync` (classe `vigilo.turbogears.repoze.plugins.mdldapsync:VigiloLdapSync`) défini aux lignes 19 à 31 est responsable de la récupération des informations depuis l'annuaire LDAP à partir de l'identité Kerberos de l'utilisateur.

Les paramètres du module `ldapsync` sont les suivants :

ldap_url Emplacement de l'annuaire LDAP, sous la forme d'une URL. Exemple : `ldap://ldap.example.com`.

ldap_base Base de recherche de l'utilisateur dans l'annuaire LDAP, sous la forme d'un Distinguished Name. Exemple : `ou=People,dc=ldap,dc=example,dc=com`.

filterstr Chaine de filtrage des résultats obtenus par la recherche. Exemple : `sAMAccountName=%`.
Cette chaîne de caractères peut contenir la variable de substitution « %s » qui sera remplacée par l'identifiant Kerberos (principal) de l'utilisateur, privé du nom du domaine (par exemple : « vigilo » si le principal Kerberos est « vigilo@EXAMPLE.COM »).
La variable du substitution ne peut être utilisée qu'une seule fois. Par défaut, le filtre utilisé est `(objectClass=*)`.

ldap_charset Encodage des caractères utilisé par l'annuaire. Cet encodage sera utilisé afin de décoder correctement les valeurs transmises par l'annuaire. Les noms d'encodages valides sont ceux définis par Python [2].
Par défaut, l'encodage utilisé est `utf-8`.

http_charset Ce paramètre est similaire au paramètre « ldap_charset » mais s'applique au serveur web. Il est utilisé afin de décoder correctement le principal Kerberos.
Par défaut, l'encodage utilisé est « utf-8 ».

cache_name Nom d'une clé qui sera définie dans la session de l'utilisateur afin de stocker son identité [3]. Cette clé est ensuite utilisée par la classe `vigilo.turbogears.repoze.plugins.externalid:ExternalIdentification` pour authentifier automatiquement l'utilisateur lors des accès suivants.

binddn DN (optionnel) à utiliser pour se connecter à l'annuaire LDAP (bind). Si ce paramètre n'est pas renseigné, le jeton Kerberos de l'utilisateur est transmis à l'annuaire afin de procéder à un bind par Kerberos (GSSAPI).

bindpw Mot de passe associé au DN indiqué dans le paramètre « binddn ».

attr_cn Nom de l'attribut (mono-valué) dans l'annuaire permettant d'obtenir le nom usuel (Common Name) de l'utilisateur. La valeur par défaut est « cn », ce qui correspond au nom de cet attribut dans un schéma LDAP classique.

attr_mail Nom de l'attribut (mono-valué) dans l'annuaire permettant d'obtenir l'adresse de courrier électronique de l'utilisateur. La valeur par défaut est « mail », ce qui correspond au nom de cet attribut dans un schéma LDAP classique.

attr_member_of Nom de l'attribut (multivalué) dans l'annuaire qui identifie les groupes dont l'utilisateur est membre. La valeur par défaut est « memberOf », ce qui correspond au nom de cet attribut dans un schéma LDAP classique.

Le module `externalid` (classe `vigilo.turbogears.repoze.plugins.externalid:ExternalIdentification`) défini aux lignes 33 à 35 est quant à lui utilisé pour mémoriser le fait que l'utilisateur s'est authentifié à l'aide d'un mécanisme d'authentification externe (ici, Kerberos) afin d'authentifier automatiquement cet utilisateur lorsqu'il tente d'accéder à une page dont l'accès est restreint.

Les paramètres du modules `externalid` sont les suivants :

cache_name Nom d'une clé dans la session contenant l'identité de l'utilisateur. Il doit s'agir de la même valeur que pour l'option *cache_name* du module `ldapsync` (de la classe `vigilo.turbogears.repoze.plugins.mdldapsync:VigiloLdapSync`).

La ligne 46 indique au framework d'authentification d'utiliser le module d'authentification `externalid` défini plus haut, afin d'authentifier automatiquement l'utilisateur s'il s'était identifié au préalable auprès du serveur via Kerberos.

La ligne 59 permet quant à elle d'utiliser le module `ldapsync` afin de synchroniser automatiquement la base de données Vigilo avec les informations issues de l'annuaire LDAP lorsque l'utilisateur s'authentifie via un mécanisme d'authentification externe (ici, Kerberos).

http://docs.python.org/library/codecs.html#standard-encodings

Pour le moment, seule la valeur `vigilo` est fonctionnelle. Les interfaces de Vigilo supposent qu'il s'agit du nom de cette clé et ne fonctionneront pas correctement si une autre valeur est utilisée ici.

Configuration du navigateur web des exploitants

Mozilla Firefox

L'activation de l'authentification par Kerberos dans Firefox se fait en modifiant 2 options dans la configuration. La configuration actuelle de Firefox peut être affichée en ouvrant un nouvel onglet, en tapant `about:config` dans la barre d'adresse et en validant.

Un message de mise en garde s'affiche, comme sur l'illustration intitulée *Avertissement de Mozilla Firefox*.

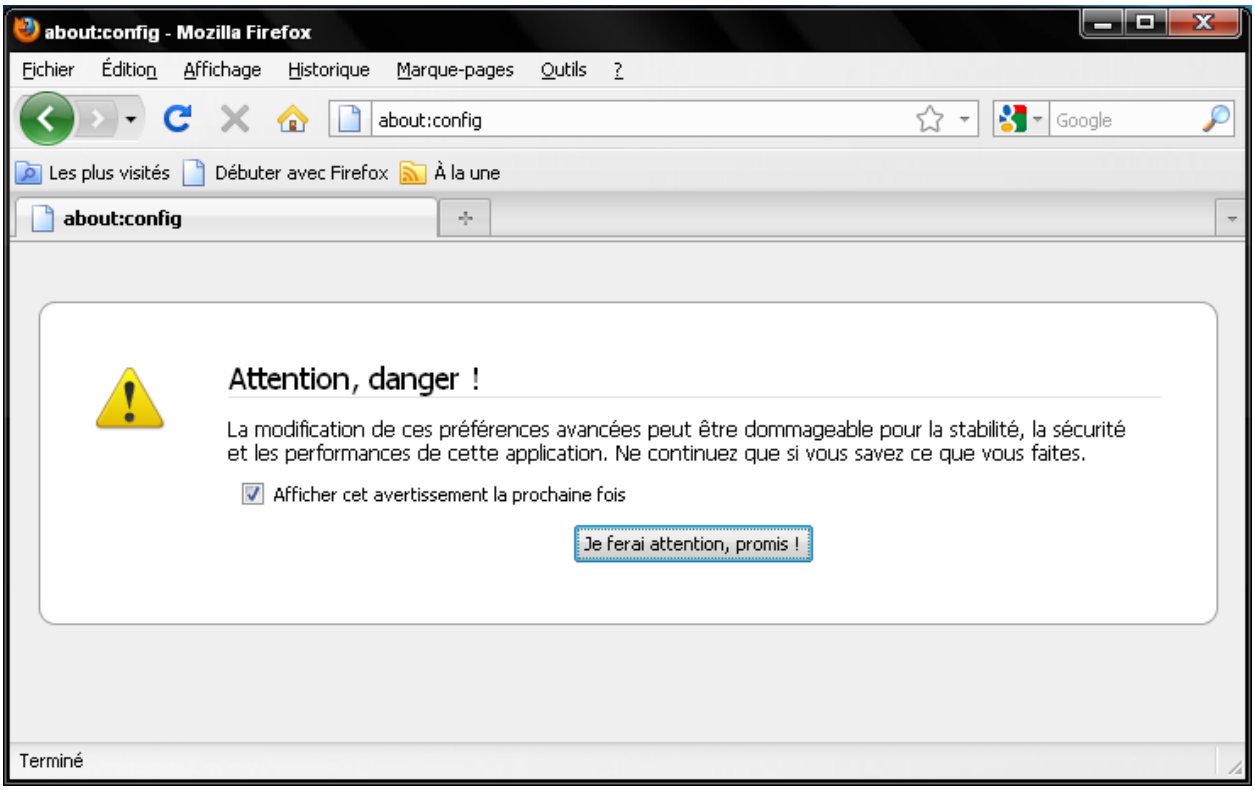


Fig. 4.2 – Avertissement de Mozilla Firefox

Après prise en compte de l'avertissement (« Je ferai attention, promis ! »), la configuration actuelle de Firefox s'affiche, comme sur l'illustration intitulée *Options de configuration de Firefox*.

Dans la barre de filtrage des options (encadrée en rouge sur l'illustration intitulée *Options de configuration de Firefox*), saisir « negotiate-auth ». Les paramètres actuels relatifs à l'authentification Kerberos (via le protocole GSSAPI) s'affichent dans la zone de résultats, comme sur l'illustration intitulée *Options relatives à l'authentification Kerberos*.

Les options à modifier, leur description et la valeur à utiliser sont récapitulées ci-dessous :

network.negotiate-auth.delegation-uris Liste les adresses Internet pour lesquelles la délégation du ticket Kerberos est autorisée. La délégation du ticket doit être autorisée pour utiliser correctement les interfaces graphiques de Vigilo.

Exemple : `https://,vigilo.example.com`. Cette valeur autorise la délégation du ticket pour les sites utilisant une connexion chiffrée (HTTPS) *ou* à destination du serveur `vigilo.example.com`.

network.negotiate-auth.trusted-uris Liste les adresses Internet pour lesquelles un ticket Kerberos doit être transmis.

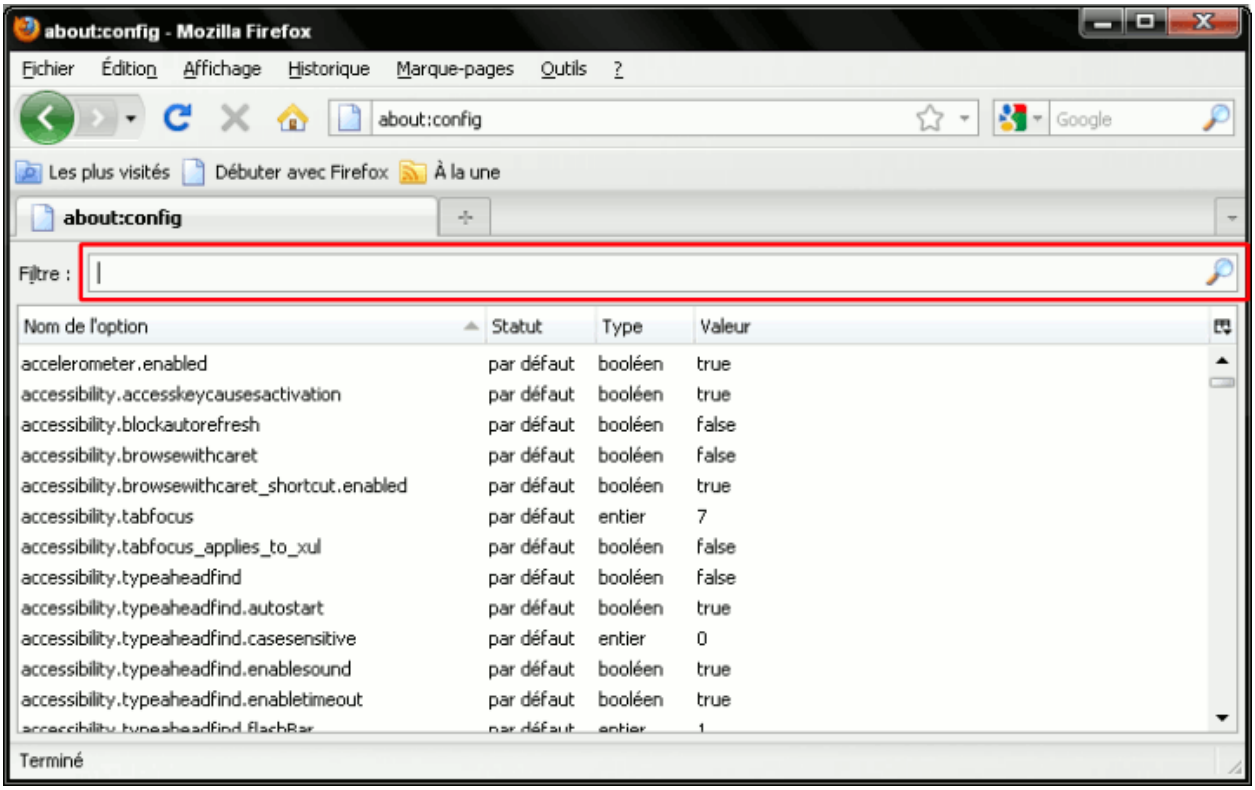


Fig. 4.3 – Options de configuration de Firefox

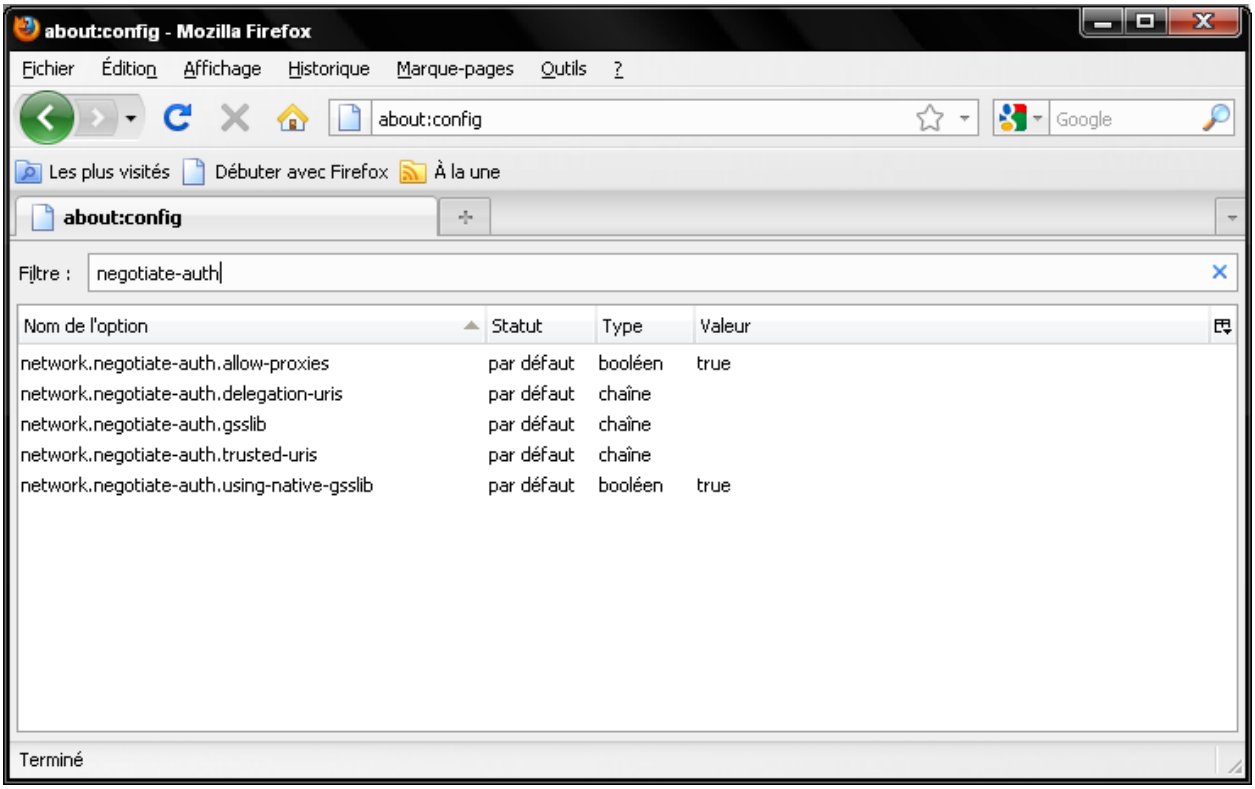


Fig. 4.4 – Options relatives à l'authentification Kerberos

Exemple : `https://,localhost,vigilo.example.com`. Cette valeur autorise la transmission du ticket aux sites utilisant une connexion chiffrée (HTTPS) *ou* à destination du serveur `vigilo.example.com`.

L'illustration suivante montre un exemple de configuration autorisant l'authentification Kerberos pour les sites hébergés par `vigilo.example.com`.

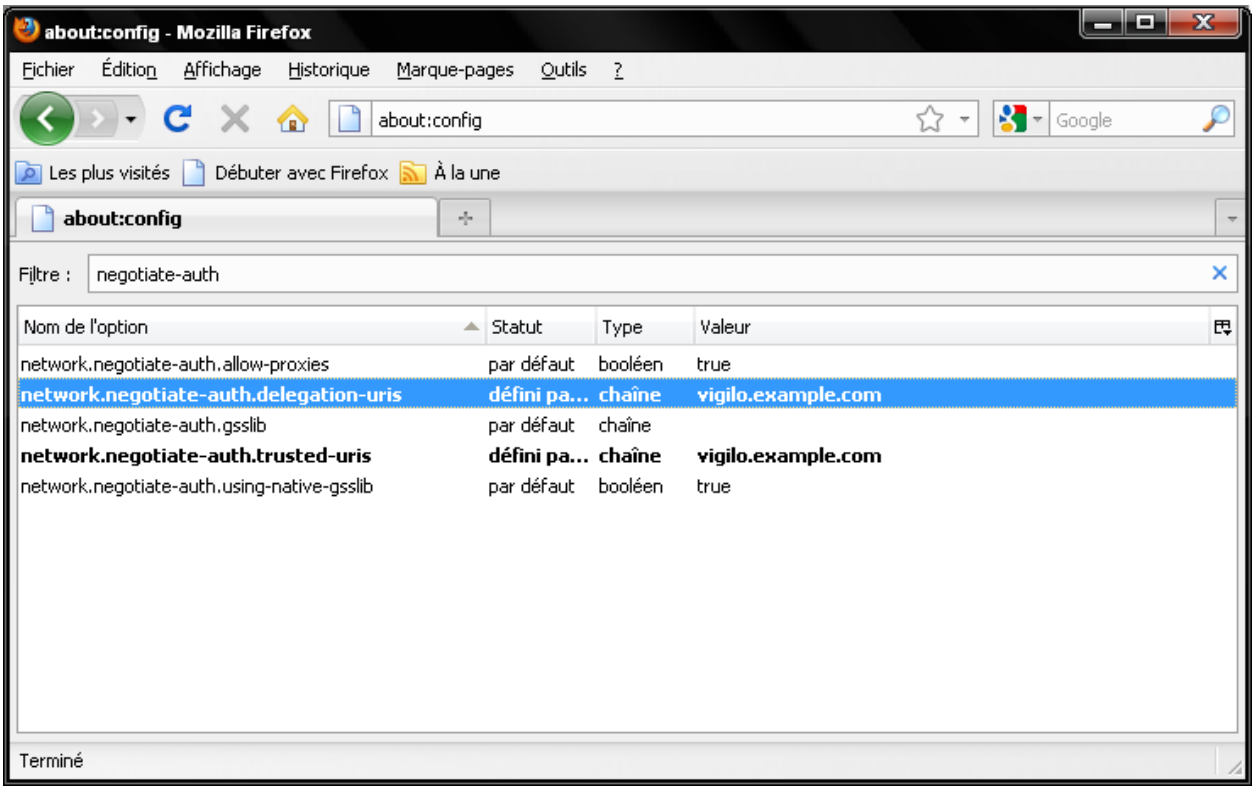


Fig. 4.5 – Configuration autorisant l'authentification Kerberos vers Vigilo

Microsoft Internet Explorer

Depuis Windows XP, la prise en charge de Kerberos dans Internet Explorer nécessite uniquement l'activation du mécanisme d'Authentification Intégrée de Windows.

L'activation se fait en allant dans le menu « Outil » et en sélectionnant « Options Internet ». La boîte de dialogue des options d'Internet Explorer s'ouvre alors (voir illustration intitulée *Options de Microsoft Internet Explorer*).

Cliquer sur l'onglet « Avancées » (en rouge sur l'illustration intitulée *Options de Microsoft Internet Explorer*), puis faire défiler les options jusqu'à trouver la ligne « Activer l'authentification Windows intégrée » (encadrée en rouge sur l'illustration intitulée *Activation de la prise en charge de Kerberos*). L'option doit être cochée pour que l'authentification par Kerberos soit supportée.

Une fois la prise en charge de Kerberos activée, valider la modification en cliquant sur le bouton « OK » et redémarrer Internet Explorer.

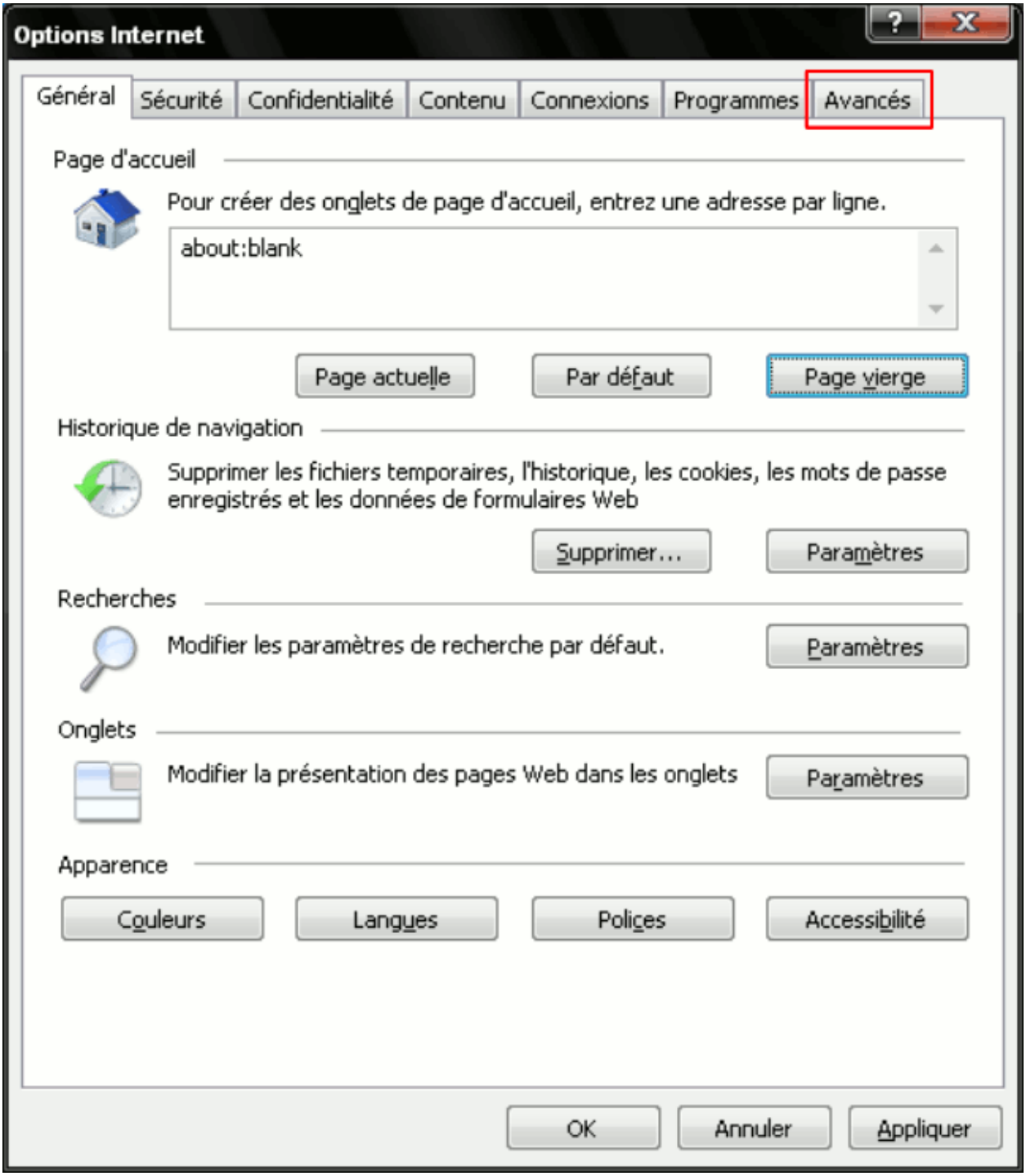


Fig. 4.6 – Options de Microsoft Internet Explorer

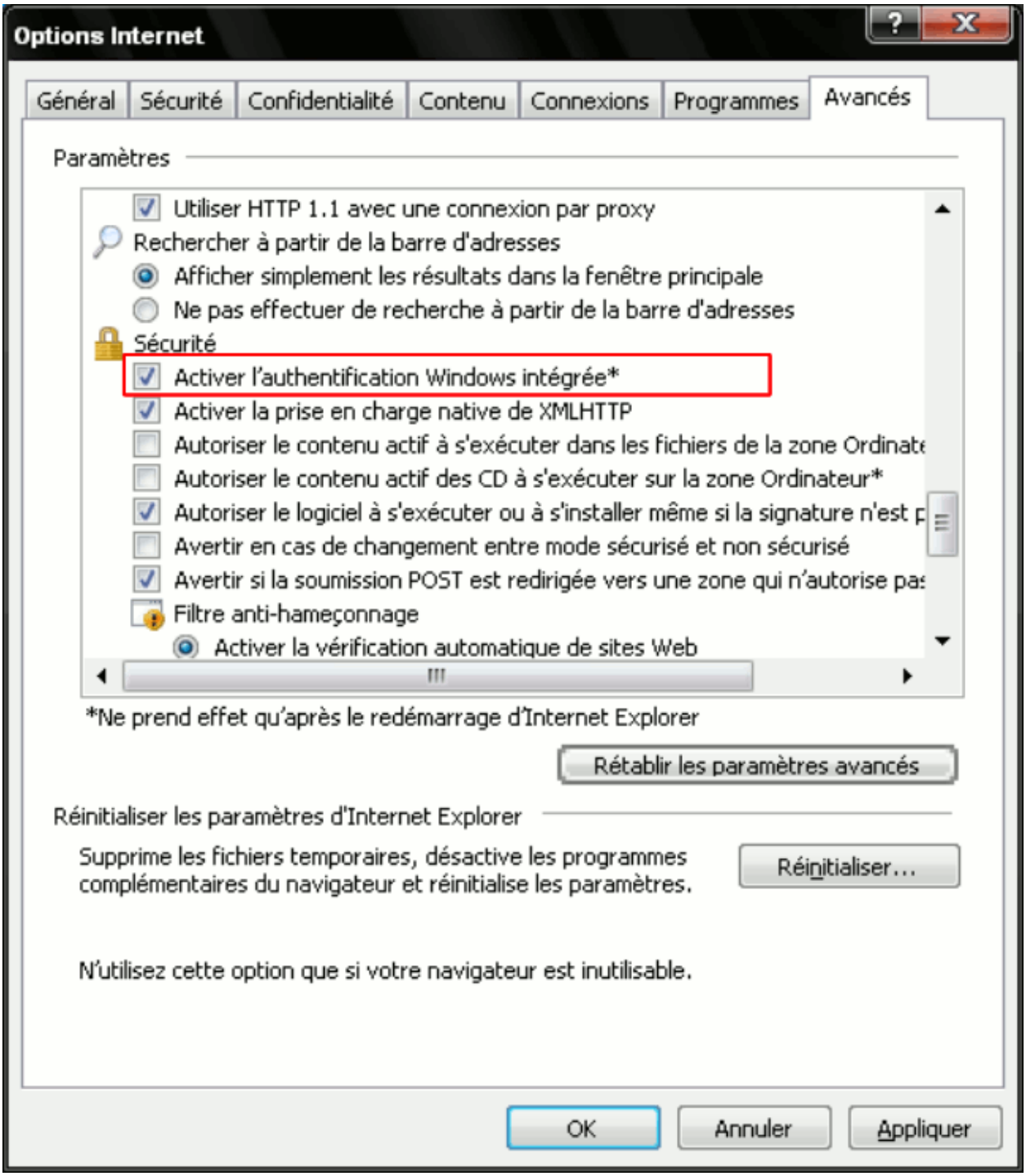


Fig. 4.7 – Activation de la prise en charge de Kerberos

Vérification du bon fonctionnement

La manière la plus simple de vérifier le bon fonctionnement de l'authentification Kerberos consiste simplement à se connecter à l'une des interfaces web de Vigilo.

Si vous ne disposez pas encore d'un ticket Kerberos valide et que la directive `KrbMethodK5Passwd` a été positionnée à « on » sur le serveur (voir le chapitre *Configuration d'Apache avec Kerberos*), une boîte de dialogue vous invite à vous authentifier à l'aide de votre identifiant et de votre mot de passe.

En revanche, si cette directive a été positionnée à « off », l'authentification échoue et une page d'erreur apparaît dans le navigateur. Dans ce cas, vous devez d'abord obtenir un ticket Kerberos pour accéder à l'application.

Sous Linux, vous pouvez obtenir un ticket Kerberos à l'aide de la commande suivante :

```
$ kinit -f <identifiant Kerberos>
```

L'option « -f » indique que le ticket peut être réutilisé par les services auxquels vous vous connectez (délégation). Elle est nécessaire au bon fonctionnement des interfaces de Vigilo.

4.6.6 Annexes

Matrice des permissions associées aux applications

Le tableau suivant liste les permissions associées à chaque application avec leur rôle.

vigiboard-access Autorise l'utilisateur à se connecter à VigiBoard.

vigiboard-update Autorise l'utilisateur à mettre à jour des événements dans VigiBoard.

vigiboard-admin Autorise l'utilisateur à forcer l'état d'un événement du bac à « OK ».

vigiboard-silence Autorise l'utilisateur à consulter et à éditer les règles de mise en silence du bac à événements.

vigigraph-access Autorise l'utilisateur à se connecter à VigiGraph.

vigimap-access Autorise l'utilisateur à se connecter à VigiMap.

vigimap-edit Autorise l'utilisateur à accéder au Mode Édition de VigiMap (pour éditer les cartes).

vigimap-admin Autorise l'utilisateur à administrer les groupes de cartes.

Matrice des permissions sur les groupes de données

Vigilo permet, via l'interface VigiAdmin, d'accorder des permissions à un utilisateur sur un groupe de données. Ces groupes de données peuvent être de trois types :

— Groupes d'éléments supervisés (hôtes ou services),
— Groupes de cartes,
— Groupes de graphes.

Les accès accordés sont soit en lecture seule, soit en lecture/écriture. Lorsqu'un accès est donné sur un groupe, il est également donné implicitement à tous les descendants de ce groupe dans l'arborescence.

La signification de l'accès en lecture/écriture aux données varie selon le type d'objet et l'interface manipulée. Le tableau suivant précise la signification de chaque type d'accès, selon le type d'objet sur lequel il est appliqué et l'interface de Vigilo consultée.

VigiBoard L'accès en lecture seule sur un groupe permet de voir les événements se rapportant aux hôtes ou aux services de ce groupe. L'accès en lecture/écriture permet en plus de modifier le statut d'acquittement ou le ticket associé aux événements concernant des hôtes ou services du groupe [4].

[4] Pour le moment, l'accès en lecture seule est suffisant pour ça...

VigiGraph Groupes d'hôtes ou de services : l'accès en lecture seule sur un groupe permet de voir les graphes se rapportant aux hôtes de ce groupe. L'accès en lecture/écriture confère les mêmes droits.

Groupes de graphes : l'accès en lecture seule permet de consulter les graphes associés à ce groupe. L'accès en lecture/écriture [5] confère exactement les mêmes droits.

VigiMap Groupes d'hôtes ou de services : l'accès en lecture seule permet de voir les hôtes et services contenus dans le groupe et apparaissant sur une carte. Il permet également d'utiliser les hôtes et services contenus dans le groupe lors de la création ou de la modification d'une carte. L'accès en lecture/écriture confère les mêmes droits [6].

Groupes de cartes : l'accès en lecture seule sur un groupe de cartes permet de consulter les cartes contenues dans ce groupe. L'accès en lecture/écriture permet en plus de créer ou de modifier des cartes dans ce groupe [7].

Groupes de graphes : L'accès en lecture seule permet de voir les graphes associés au groupe lorsqu'ils sont utilisés sur une carte au travers d'un lien de type « service ». Il permet également d'utiliser ces graphes lors de la création ou de la modification d'une carte, au sein d'un lien de type « service ». L'accès en lecture/écriture confère les mêmes droits [8].

Glossaire - Terminologie

Ce chapitre recense les différents termes techniques employés dans ce document et donne une brève définition de chacun de ces termes.

Glossaire - Terminologie

Ce chapitre recense les différents termes techniques employés dans ce document et donne une brève définition de chacun de ces termes.

API Interface logicielle de programmation, permettant à un développeur d'enrichir la liste des fonctionnalités proposées par un logiciel.

CGI Interface standard de communication entre un serveur web et un programme capable de générer une réponse HTTP valide. Il s'agit par exemple de l'interface retenue par Nagios (< 3.3) pour la génération de ses pages web.

CSS Feuilles de styles permettent de modifier la représentation graphique des éléments d'une page web. La version généralement supportée par les navigateurs est la version 2, définie par le document disponible sur http://www.w3.org/TR/CSS2/.

CSV À l'origine, désigne un format textuel de transfert de données dans lequel les entrées sont séparées par des retours chariot et les champs par des virgules (comma). De nos jours, désigne plus généralement un format tabulé pour l'échange de données en vue de leur traitement dans un logiciel de type tableur ou par un traitement automatisé (scripts).

DN Identifiant unique dans le cadre d'un annuaire LDAP.

Événement brut Alerte envoyée par Nagios au corrélateur de Vigilo pour analyse.

Événement corrélé Incident détecté par Vigilo suite à la corrélation des alertes Nagios (évènements bruts). Un événement corrélé est causé par un unique événement brut (exemple : la panne d'un routeur), mais de nombreux autres évènements bruts peuvent lui être rattachés (exemple : les alertes indiquant que les serveurs situés derrière le routeur en panne sont indisponibles). Ces événements secondaires rattachés à l'événement corrélé sont alors appelés « événements bruts masqués ».

KDC Serveur permettant un transfert sécurisé des clés de chiffrement utilisées pour les communications entre divers services. Ce serveur est notamment utilisé lors des échanges initiaux du protocole Kerberos.

L'accès en lecture seule ou en lecture/écriture devrait permettre de voir les graphes en question.
Cette fonctionnalité n'est pas encore implémentée.
Pour le moment, l'accès en lecture seule se comporte comme l'accès en lecture/écriture.
Cette fonctionnalité n'est pas encore implémentée.

LDAP Protocole pour l'interrogation d'un annuaire, servant généralement à recenser les utilisateurs autorisés d'un système et les différentes propriétés associées à ces utilisateurs.

OS Système d'exploitation.

Nagios Composant libre de supervision système et réseau.

RRD Base de données de taille fixe utilisant des fichiers circulaires, dont les données sont progressivement compressées (avec perte) au fur et à mesure de leur vieillissement.

RRDtool Composant libre de gestion de bases RRD (stockage, restitution, génération de graphiques).

SGBD[R] Logiciel permettant d'héberger des bases de données [relationnelles] sur une machine.

SQL Langage de requêtes structuré pour l'interrogation d'une base de données relationnelle.

URL Chaîne de caractères permettant d'identifier une ressource, par exemple une page web sur Internet. Exemple : http://www.vigilo-nms.com/

WSGI Une interface pour la communication entre une application et un serveur web, similaire à CGI. Il s'agit de l'interface utilisée par Vigilo.

4.7 Guide d'administration

4.7.1 Installation

Prérequis logiciels

Afin de pouvoir faire fonctionner VigiRRD, l'installation préalable des logiciels suivants est requise :

- python (>= 2.5), sur la machine où VigiRRD est installé
- Apache (>= 2.2.0), sur la machine où VigiRRD est installé
- apache-mod_wsgi (>= 2.3), sur la machine où VigiRRD est installé
- rrdtool (>= 1.3), sur la machine où VigiRRD est installé

Reportez-vous aux manuels de ces différents logiciels pour savoir comment procéder à leur installation sur votre machine. VigiRRD requiert également la présence de plusieurs dépendances Python. Ces dépendances seront automatiquement installées en même temps que le paquet de VigiRRD.

Installation du paquet RPM

L'installation du connecteur se fait en installant simplement le paquet RPM « vigilo-vigirrd ». La procédure exacte d'installation dépend du gestionnaire de paquets utilisé.

Les instructions suivantes décrivent la procédure pour les gestionnaires de paquets RPM les plus fréquemment rencontrés.

Installation à l'aide de urpmi :

```
urpmi vigilo-vigirrd
```

Installation à l'aide de yum :

```
yum install vigilo-vigirrd
```

4.7.2 Démarrage et arrêt de VigiRRD

VigiRRD fonctionne comme un site web standard. À ce titre, il n'est pas nécessaire d'exécuter une commande spécifique pour démarrer VigiRRD, dès lors que le serveur web qui l'héberge a été lancé, à l'aide de la commande :

```
service httpd start
```

De la même manière, il n'y a pas de commande spécifique pour arrêter VigiRRD. L'application est arrêtée en même temps que le serveur web, à l'aide de la commande :

service httpd stop

4.7.3 Configuration de VigiRRD

La configuration initialement fournie avec VigiRRD est très rudimentaire. Elle est composée uniquement du fichier « settings.ini ».

Ce chapitre a pour but de présenter les différentes options de configuration disponibles afin de configurer VigiRRD en fonction de vos besoins.

Les chapitres suivants réutilisent l'ordre des directives de configuration utilisé dans le fichier « settings.ini » de l'application. Toutes les options de configuration citées ici se trouvent sous la section [app :main] du fichier « settings.ini ».

Le chapitre donne des informations quant à la méthode utilisée pour intégrer VigiRRD sur un serveur web de type Apache, grâce au module mod_wsgi.

La configuration de la journalisation des événements se fait également au travers du fichier « settings.ini ». Néanmoins, comme ce procédé se fait de la même manière dans les différents composants de Vigilo, celui-ci fait l'objet d'une documentation séparée dans le document Vigilo – Journaux d'événements.

Configuration des éléments de sécurité

Ce chapitre décrit les options relatives à la gestion des données de sécurité (clés de chiffrements, etc.) utilisées par VigiRRD.

Clé de chiffrement / déchiffrement des sessions

Afin de ne pas dévoiler certains paramètres associés à un utilisateur, le fichier de session qui contient ces paramètres est chiffré à l'aide d'une clé symétrique, utilisée à la fois pour le chiffrement et le déchiffrement des sessions de tous les utilisateurs de VigiRRD.

L'option « beaker.session.secret » permet de choisir la clé utilisée pour chiffrer et déchiffrer le contenu des sessions. Cette clé peut être la même que celle configurée pour le chiffrement / déchiffrement des cookies (voir le chapitre), mais ceci est déconseillé afin d'éviter que la compromission de l'une des deux clés n'entraine la compromission de l'autre.

De la même manière, vous pouvez configurer les autres interfaces graphiques de Vigilo pour utiliser les mêmes clés, ou opter de préférence pour des clés différentes (là encore, pour éviter la propagation d'une compromission).

Clé de chiffrement / déchiffrement des cookies

L'association entre un utilisateur et sa session se fait à l'aide d'un cookie de session enregistré sur le navigateur de l'utilisateur.

De la même manière que les sessions sont chiffrés afin de garantir la confidentialité de leur contenu, le cookie de session est également chiffré afin de protéger son contenu.

L'option « sa_auth.cookie_secret » permet de choisir la clé utilisée pour chiffrer et déchiffrer le contenu du cookie. Cette clé peut être la même que celle configurée pour le chiffrement / déchiffrement des sessions (voir le chapitre),

mais ceci est déconseillé afin d'éviter que la compromission de l'une des deux clés n'entraine la compromission de l'autre.

De la même manière, vous pouvez configurer les autres interfaces graphiques de Vigilo pour utiliser les mêmes clés, ou opter de préférence pour des clés différentes (là encore, pour éviter la propagation d'une compromission).

Configuration de l'interface

Ce chapitre décrit les options qui modifient l'apparence de l'interface graphique de VigiRRD.

Emplacement du cache de graphes

Afin d'améliorer les performances de VigiRRD, lorsque l'image d'un graphe est générée, elle est enregistré dans un dossier qui joue le rôle d'un cache. Si l'utilisateur demande à nouveau le graphe, VigiRRD renverra directement l'image précédemment générée et enregistrée dans le cache au lieu de la générer à nouveau.

L'option « image_cache_dir » permet de définir l'emplacement du dossier temporaire jouant le rôle de cache de graphes. Il est recommandé d'utiliser un dossier temporaire (un sous-dossier du dossier « /tmp » par exemple).

Dossier de stockage des bases RRD

Les graphes générés par VigiRRD le sont à partir de données contenues dans des bases de données de type « round-robin » (RRD).

L'option « rrd_base » permet d'indiquer le dossier racine sous lequel les fichiers RRD sont stockés. Ce dossier doit coïncider avec l'option équivalente du connecteur de métrologie (vigilo-connector-metro).

Dossier contenant les configurations auto-générées de VigiRRD

Chaque serveur VigiRRD ne gère qu'une portion du parc supervisé, ce qui facilite le passage à l'échelle de la supervision. Afin de savoir quels sont les équipements du parc sous sa responsabilité, VigiRRD utilise des fichiers de configuration Python, auto-générés par VigiConf.

L'option « conf_dir » permet de spécifier le dossier dans lequel les fichiers de configuration Python auto-générés sont stockés par VigiConf. Une valeur appropriée pour cette option est « /etc/vigilo/vigirrd », qui correspond à une installation standard de VigiRRD et VigiConf.

Formatage de l'export CSV

VigiRRD est capable d'exporter les données d'un graphe de métrologie au format CSV. Plusieurs options sont disponibles afin de paramétrer l'export CSV.

csv_delimiter_ch ar	Caractère de séparation des champs dans l'export CSV.
csv_quote_char	Caractère de délimitation de la valeur d'un champ.
csv_escape_char	Caractère d'échappement pour les caractères spéciaux (ceux définis dans les options précédentes).
csv_date_format	Format du champ « Timestamp » dans l'export CSV. Par défaut, ce champ contient l'horodatage UNIX à laquelle la valeur a été émise. Vous pouvez spécifier une chaîne de formatage de dates en utilisant les options de formatage décrites sur http://docs.python.org/release/2.5/lib/module-time.html. Un exemple d'une telle chaîne de formatage pourrait être : « %d/%m/%Y %H :%M :%S » pour afficher la date et l'heure au format français (JJ/MM/AAAA hh :mm :ss).
csv_respect_lo cale	Spécifie si le formatage des valeurs (nombres et dates) dans l'export CSV doit tenir compte des conventions applicables à la langue de l'utilisateur (locale) ou non. Par défaut, VigiRRD ne tente pas de respecter les conventions régionales de l'utilisateur et utilise des conventions neutres à la place, en vu de faciliter un traitement ultérieur automatisé des données exportées.

Configuration des sessions

Chaque fois qu'un utilisateur se connecte à VigiRRD, un fichier de session est créé permettant de sauvegarder certaines préférences de cet utilisateur (par exemple, le thème de l'application, la taille de la police de caractères, etc.). Ce chapitre décrit les options relatives à la gestion des sessions.

Emplacement des fichiers de session

Le dossier dans lequel les fichiers de session seront stockés est indiqué par l'option « cache_dir ».

Nom du cookie de session

Afin d'associer un utilisateur au fichier de session qui lui correspond, un cookie de session est créé sur le navigateur de l'utilisateur. L'option « beaker.session.key » permet de choisir le nom du cookie créé. Le nom doit être composé de caractères alphanumériques (a-zA-Z0-9) et commencer par une lettre (a-zA-Z).

4.7.4 Intégration de VigiRRD avec Apache / mod_wsgi

VigiRRD a été testé avec le serveur libre Apache. L'application utilise en outre le module Apache « mod_wsgi » pour communiquer avec le serveur. Ce module implémente un modèle de communication basé sur l'interface WSGI. Le reste de ce chapitre décrit la configuration utilisée pour réaliser cette intégration.

Fichier de configuration pour Apache

Le fichier de configuration pour l'intégration de VigiRRD dans Apache se trouve généralement dans /etc/vigilo/vigirrd/vigirrd.conf (un lien symbolique vers ce fichier est créé dans le dossier de configuration d'Apache, généralement dans /etc/httpd/conf.d/vigirrd.conf).

En général, il n'est pas nécessaire de modifier le contenu de ce fichier. Ce chapitre vise toutefois à fournir quelques informations sur le fonctionnement de ce fichier, afin de permettre d'éventuelles personnalisations de ce comportement.

Ce fichier tente tout d'abord de charger le module « mod_wsgi » (directive LoadModule) puis ajoute les directives de configuration nécessaires à Apache pour faire fonctionner VigiRRD, reprises partiellement ci-dessous :

```
WSGIRestrictStdout off
WSGIPassAuthorization on
WSGIDaemonProcess vigirrd user=apache group=apache threads=2
WSGIScriptAlias /vigilo/vigirrd "/etc/vigilo/vigirrd/vigirrd.wsgi"

KeepAlive Off

<Directory "/etc/vigilo/vigirrd/">
<Files "vigirrd.wsgi">
WSGIProcessGroup vigirrd
WSGIApplicationGroup %{GLOBAL}

Order deny,allow
Allow from all
</Files>
</Directory>
```

L'option WSGIRestrictStdout est positionnée à « off » afin d'éviter qu'Apache ne tue le processus de l'application lorsque des données sont envoyées sur la sortie standard. Ceci permet de récupérer les erreurs critiques pouvant être émises par l'application. Ces erreurs apparaissent alors dans le journal des événements d'Apache (configuré par la directive error_log).

L'option WSGIPassAuthorization positionnée à « on » indique à Apache et mod_wsgi que les informations d'authentification éventuellement transmises par l'utilisateur doivent être transmises à VigiRRD. Bien que VigiRRD n'utilise pas les informations d'authentification d'Apache, cette option est positionnée par homogénéité avec la configuration des autres interfaces web de Vigilo.

L'option WSGIDaemonProcess permet de créer un groupe de processus affecté au traitement des requêtes HTTP destinées à VigiRRD. Il permet d'utiliser un nom d'utilisateur et un groupe prédéfini (afin de réduire les privilèges nécessaires), ainsi que le nombre de processus légers à utiliser pour traiter les requêtes (ici, 2).

L'option WSGIScriptAlias indique l'emplacement à partir duquel VigiRRD sera accessible (ici, http://example.com/vigilo/vigirrd/ si le serveur Apache est configuré pour le domaine « example.com ») et l'emplacement du script WSGI nécessaire au lancement de l'application (voir le chapitre).

L'option KeepAlive positionnée à « off » est nécessaire afin de contourner un problème dans le module « mod_wsgi » d'Apache.

Les autres options permettent d'exécuter le script WSGI de VigiRRD à l'aide du groupe de processus défini précédemment.

La liste complète des directives de configuration supportées par le module « mod_wsgi » d'Apache est disponible à l'adresse http://code.google.com/p/modwsgi/wiki/ConfigurationDirectives.

Script WSGI de VigiRRD

Le script WSGI de VigiRRD est un script Python très simple qui a pour but de démarrer l'exécution de VigiRRD à partir du fichier de configuration associé (/etc/vigilo/vigirrd/settings.ini).

Vous n'avez généralement pas besoin de modifier son contenu, sauf éventuellement pour adapter l'emplacement du fichier de configuration en fonction de votre installation.

4.7.5 Annexes

Glossaire - Terminologie

Ce chapitre recense les différents termes techniques employés dans ce document et donne une brève définition de chacun de ces termes.

API Interface logicielle de programmation, permettant à un développeur d'enrichir la liste des fonctionnalités proposées par un logiciel.

CGI Interface standard de communication entre un serveur web et un programme capable de générer une réponse HTTP valide. Il s'agit par exemple de l'interface retenue par Nagios (< 3.3) pour la génération de ses pages web.

CSS Feuilles de styles permettent de modifier la représentation graphique des éléments d'une page web. La version généralement supportée par les navigateurs est la version 2, définie par le document disponible sur http://www.w3.org/TR/CSS2/.

CSV À l'origine, désigne un format textuel de transfert de données dans lequel les entrées sont séparées par des retours chariot et les champs par des virgules (comma). De nos jours, désigne plus généralement un format tabulé pour l'échange de données en vue de leur traitement dans un logiciel de type tableur ou par un traitement automatisé (scripts).

DN Identifiant unique dans le cadre d'un annuaire LDAP.

Événement brut Alerte envoyée par Nagios au corrélateur de Vigilo pour analyse.

Événement corrélé Incident détecté par Vigilo suite à la corrélation des alertes Nagios (évènements bruts). Un événement corrélé est causé par un unique événement brut (exemple : la panne d'un routeur), mais de nombreux autres évènements bruts peuvent lui être rattachés (exemple : les alertes indiquant que les serveurs situés derrière le routeur en panne sont indisponibles). Ces événements secondaires rattachés à l'événement corrélé sont alors appelés « événements bruts masqués ».

KDC Serveur permettant un transfert sécurisé des clés de chiffrement utilisées pour les communications entre divers services. Ce serveur est notamment utilisé lors des échanges initiaux du protocole Kerberos.

LDAP Protocole pour l'interrogation d'un annuaire, servant généralement à recenser les utilisateurs autorisés d'un système et les différentes propriétés associées à ces utilisateurs.

OS Système d'exploitation.

Nagios Composant libre de supervision système et réseau.

RRD Base de données de taille fixe utilisant des fichiers circulaires, dont les données sont progressivement compressées (avec perte) au fur et à mesure de leur vieillissement.

RRDtool Composant libre de gestion de bases RRD (stockage, restitution, génération de graphiques).

SGBD[R] Logiciel permettant d'héberger des bases de données [relationnelles] sur une machine.

SQL Langage de requêtes structuré pour l'interrogation d'une base de données relationnelle.

URL Chaîne de caractères permettant d'identifier une ressource, par exemple une page web sur Internet. Exemple : http://www.vigilo-nms.com/

WSGI Une interface pour la communication entre une application et un serveur web, similaire à CGI. Il s'agit de l'interface utilisée par Vigilo.

CHAPITRE 5

Utilisation

Table des matières :

5.1 Manuel utilisateur

5.1.1 Installation

Pré-requis logiciels

Afin de pouvoir faire fonctionner VigiConf, l'installation préalable des logiciels suivants est requise :

- python (>= 2.5), sur la machine où VigiConf est installé
- postgresql-server (>= 8.2), éventuellement sur une machine distante

Reportez-vous aux manuels de ces différents logiciels pour savoir comment procéder à leur installation sur votre machine.

VigiConf requiert également la présence de plusieurs dépendances Python. Ces dépendances seront automatiquement installées en même temps que le paquet de VigiConf.

Installation du paquet RPM

L'installation de VigiConf se fait en installant simplement le paquet RPM "`vigilo-vigiconf`". La procédure exacte d'installation dépend du gestionnaire de paquets utilisé. Les instructions suivantes décrivent la procédure pour les gestionnaires de paquets RPM les plus fréquemment rencontrés.

Installation à l'aide de urpmi :

```
urpmi vigilo-vigiconf        # sur la machine d'administration
urpmi vigilo-vigiconf-local  # sur les autres machines de la plate-forme de␣
↪supervision
```

Installation à l'aide de yum :

```
yum install vigilo-vigiconf        # sur la machine d'administration
yum install vigilo-vigiconf-local  # sur les autres machines de la plate-forme de␣
↪supervision
```

5.1.2 Configuration

La configuration de VigiConf comprend deux parties. D'une part, la configuration de l'outil en lui-même. D'autre part, la configuration du parc informatique supervisé par Vigilo (et dont la configuration est gérée par VigiConf). Ce chapitre ne porte que sur la configuration de VigiConf. Pour la documentation sur la configuration du parc informatique supervisé, reportez-vous au chapitre *Configuration du parc à superviser*.

Par défaut, la configuration de VigiConf se trouve dans `/etc/vigilo/vigiconf/settings.ini`. Vous devrez **impérativement** modifier ce fichier de configuration avant d'utiliser VigiConf.

Si vous le souhaitez, vous pouvez également placer les options de configuration communes à tous les modules de Vigilo installés sur la machine dans le fichier `/etc/vigilo/settings.ini`. Ce fichier générique sera chargé en premier au lancement de VigiConf.

Ces fichiers sont composés de différentes sections permettant de paramétrer des aspects divers de VigiConf, chacune de ces sections peut contenir un ensemble de valeurs sous la forme `cle=valeur`. Les lignes commençant par " ;" ou "#" sont des commentaires et sont par conséquent ignorées.

Le format de ces fichiers peut donc être résumé dans l'extrait suivant :

```
# Ceci est un commentaire
; Ceci est également un commentaire

[section1]
option1=valeur1
; ...
optionN=valeurN

; ...

[sectionN]
option1=valeur1
; ...
optionN=valeurN
```

Configuration de la base de données

Connexion

VigiConf enregistre une partie des informations de la configuration du parc supervisé en base de données. La configuration de la connexion à cette base de données se fait en modifiant la valeur de la clé `sqlalchemy_url` sous la section `[database]`.

Cette clé consiste en une *URL* définissant tous les paramètres nécessaires pour pouvoir se connecter à la base de données. Le format de cette URL est le suivant :

```
sgbd://utilisateur:mot_de_passe@serveur:port/base_de_donnees
```

Le champ `:port` est optionnel et peut être omis si vous utilisez le port par défaut d'installation du SGBD choisi.

Par exemple, voici la valeur correspondant à une installation mono-poste par défaut de Vigilo :

```
postgresql://vigilo:vigilo@localhost/vigilo
```

Avertissement : À l'heure actuelle, seul PostgreSQL a fait l'objet de tests intensifs. D'autres SGBD peuvent également fonctionner, mais aucun support ne sera fourni pour ceux-ci.

Configuration du dépôt pour le suivi des évolutions

Les modifications apportées à la configuration du parc sont enregistrées dans un dépôt *SVN*, permettant ainsi d'assurer un suivi des modifications et un éventuel retour arrière.

La configuration de ce dépôt se fait en utilisant les clés suivantes de la section `vigiconf` :

confdir Dossier contenant un « checkout » (version de travail) du dépôt SVN et dont les modifications seront enregistrées dans le dépôt à chaque utilisation de la commande "`vigiconf deploy`".

Cette option doit pointer vers le dossier où la configuration du parc supervisé est sauvegardée.

svnusername Nom d'utilisateur pour accéder au dépôt SVN.

svnpassword Mot de passe pour accéder au dépôt SVN.

svnrepository URL indiquant l'emplacement du dépôt SVN. Il peut s'agir d'une URL pointant vers un dépôt local (`file://`) ou distant (`ssh+svn://`, `http://`, etc.). Référez-vous à la documentation de Subversion pour les différents protocoles supportés.

Autres options de configuration

Les paragraphes qui suivent décrivent les autres options de configuration disponibles dans VigiConf et situées sous la section `[vigiconf]` du fichier `settings.ini`.

En règle générale, les valeurs correspondant à une nouvelle installation de VigiConf sont suffisantes et il n'est pas nécessaire de les modifier.

Répertoire de travail pour la génération

L'option "`libdir`" permet de spécifier l'emplacement du répertoire de travail servant à générer les fichiers de configuration des applications.

La valeur définie dans la configuration initiale est `/var/lib/vigilo/vigiconf`.

Emplacement final de la configuration

La directive "`targetconfdir`" permet d'indiquer le dossier vers lequel les fichiers de configuration finaux seront télé-déployés sur les serveurs de supervision.

La valeur définie dans la configuration initiale est `/etc/vigilo/vigiconf`. Les applications dont dépend Vigilo (ex : Nagios) doivent être configurées pour aller chercher leur fichier de configuration dans le sous-dossier "`prod`" de ce dossier.

Répertoire des plugins de VigiConf

L'option "`pluginsdir`" permet de faciliter l'extension de VigiConf à l'aide de plugins (modules complémentaires). Il s'agit de l'emplacement d'un répertoire qui contiendra des modules Python (eggs) qui seront chargés automatiquement au lancement de VigiConf. Ces modules ont la possibilité d'enregistrer des points d'entrée Python afin d'ajouter de nouvelles fonctionnalités.

La valeur de cette option dans la configuration initiale fournie avec VigiConf est `/etc/vigilo/vigiconf/plugins`.

Emplacement du verrou

Afin d'éviter un conflit lorsque plusieurs administrateurs du même parc effectuent un nouveau déploiement de la configuration simultanément, VigiConf acquiert un verrou au démarrage. Ce verrou est automatiquement libéré lors de l'arrêt de VigiConf.

La directive "`lockfile`" permet de spécifier l'emplacement du fichier qui correspondra au verrou. Dans la configuration fournie par défaut avec VigiConf, le verrou est enregistré dans `/var/lock/subsys/vigilo-vigiconf/vigiconf.token`.

Mode de simulation des opérations

L'option "`simulate`" est un booléen qui permet d'activer un mode spécial de VigiConf dans lequel les opérations sont simulées et ne sont pas validées. Cette option est destinée uniquement au débogage de l'application lors de la phase de développement et doit être positionnée à "`False`" en production.

5.1.3 Configuration du parc à superviser

La configuration du parc à superviser se fait au travers de fichiers XML. Ces fichiers sont stockés dans le répertoire pointé par l'option "`confdir`" de la section "`vigiconf`" dans le fichier de configuration de VigiConf. Des fichiers d'exemple sont installés en même temps que VigiConf.

Ce chapitre présente la structure de la configuration et le contenu des différents fichiers.

Fichiers de configuration XML

Afin d'éviter les erreurs de saisie dans les fichiers de configuration de VigiConf, ceux-ci font systématiquement l'objet d'une validation à l'aide de schémas XML.

Ces schémas sont stockés dans :

`/usr/lib`*`arch`*`/python`*`version`*`/site-packages/vigilo/vigiconf/validation/xsd/`

Par exemple, pour une installation standard de Python 2.5 sur une machine équipée d'une architecture x86 :

`/usr/lib/python2.5/site-packages/vigilo/vigiconf/validation/xsd/`

Dans la suite de ce document, on considère qu'un fichier *`type`*`.xml` de la configuration de VigiConf est valide s'il respecte le schéma défini dans le fichier *`type`*`.xsd` situé dans ce répertoire correspondant au type d'objet manipulé.

Pour le reste des explications de ce chapitre, tous les emplacements de fichiers ou dossiers indiqués sont relatifs au dossier de configuration du parc (par défaut, `/etc/vigilo/vigiconf/conf.d/`)

Configuration des hôtes

Le dossier "`hosts`" contient les fichiers de définition des hôtes supervisés du parc. Tous les fichiers XML de ce dossier sont analysés et doivent contenir la définition d'un ou plusieurs hôtes.

La balise à la racine de ce document se nomme "`hosts`" et peut contenir un ou plusieurs blocs "`host`", correspondant chacun à la définition d'un hôte.

Le fragment de code suivant rappelle la structure générale du fichier :

```
<?xml version="1.0"?>
<hosts>
  <host name="host1" attr1="..." attr2="..." attrN="...">
    ...
  </host>
  <host name="host2" attr1="..." attr2="..." attrN="...">
    ...
  </host>
  ...
</hosts>
```

Définition d'un hôte

Un hôte est défini à l'aide d'une balise *host* ayant la forme suivante :

```
<host name="localhost" address="127.0.0.1" ventilation="P-F">
  ...
</host>
```

Un bloc de données *host* possède les attributs suivants :

name [requis] Nom de l'hôte. Il peut s'agir d'un nom entièrement qualifié (FQDN) ou simplement d'un nom court permettant d'identifier l'équipement au sein du parc.

address [optionnel] Adresse permettant de communiquer avec l'hôte. Il peut s'agir d'une adresse IP (v4 ou v6) ou d'un nom de domaine entièrement qualifié (FQDN). Si cet attribut est omis, la valeur de l'attribut *name* est utilisée.

ventilation [optionnel] Nom du groupe de supervision dans lequel cet hôte doit être placé (ventilé).

En général, il n'est pas nécessaire de spécifier cet attribut. VigiConf tente de le déduire automatiquement à partir des noms des groupes auxquels l'hôte est rattaché. Cet attribut permet essentiellement de résoudre les conflits entre les groupes. Il peut également être utilisé pour affecter l'hôte à un groupe de supervision qui n'a aucune relation avec les groupes métier.

La balise *host* peut contenir les balises suivantes :

- `class` (0 ou plus)
- `template` (0 ou plus)
- `attribute` (0 ou plus)
- `nagios` (0 ou 1)
- `test` (0 ou plus)
- `tag` (0 ou plus)
- `group` (1 ou plus)

Balise "`class`"

Syntaxe :

```
<class>nom de la classe</class>
```

Indique la ou les classes d'équipements auxquelles l'hôte appartient. En fonction de ces classes, des tests spécifiques peuvent être disponibles afin d'obtenir des informations plus précises sur l'état de l'équipement.

Balise "template"

Syntaxe :

```
<template>nom du modèle</template>
```

Précise le nom du modèle d'hôtes duquel cet hôte hérite une partie de ses propriétés. L'utilisation de l'héritage est pratique lorsque votre parc est composé d'éléments (serveurs, routeurs, etc.) homogènes. Vous pouvez alors définir un modèle (template) pour chaque type d'équipement avec tous les tests associés et créer ensuite simplement une définition d'hôte pour chaque équipement.

Cette balise peut être utilisée à plusieurs reprises. Les paramètres du dernier modèle chargé écrasent ceux des modèles précédents. Les valeurs définies au niveau d'un hôte écrase toujours les valeurs définies au niveau d'un modèle hérité (en particulier, les paramètres des tests).

Note : Pour être utilisable via cette balise, le modèle doit avoir été défini à l'aide de la balise *hosttemplate* (voir *Configuration des modèles d'hôtes*).

Avertissement : En cas de conflits liés aux dépendances des modèles, il se peut que les modèles soient réordonnés (un message d'avertissement est alors émis par VigiConf lors du déploiement). Dans ce cas, l'ordre d'application des paramètres peut être légèrement différent de l'ordre d'affectation des modèles dans le fichier de configuration.

Balise "attribute"

Syntaxes :

```
<attribute name="nom de l'attribut">valeur de l'attribut</attribute>

<attribute name="nom de l'attribut">
  <item>valeur 1</item>
  <item>valeur 2</item>
</attribute>
```

Cette balise permet de fixer certains des attributs de l'hôte, comme par exemple le nombre de processeurs présents sur la machine. En général, ces informations sont extraites automatiquement des équipements par une interrogation SNMP (voir à ce sujet le chapitre « *Découverte des services à superviser sur un hôte* »).

Les noms d'attributs utilisables dépendent des tests de supervision installés avec VigiConf. Par défaut, les attributs suivants sont disponibles :

- "collectorTimeout" : délai d'attente utilisé lors de la récupération des données SNMP de l'hôte ;
- "cpulist" : la liste des identifiants des processeurs sur l'équipement ;
- "fans" : la liste des identifiants des ventilateurs sur l'équipement ;
- "snmpCommunity" : la communauté pour l'accès SNMP à l'équipement ;
- "snmpPort" : le port SNMP à utiliser (par défaut, le port 161 est utilisé) ;
- "snmpVersion" : la version SNMP à utiliser (par défaut, la version 2 est utilisée) ;
- "tempsensors" : la liste des noms des sondes de température présentes sur l'équipement.

Balise "test"

Syntaxe :

```
<test name="nom du test">
  <arg name="nom_argument_1">valeur argument 1</arg>
  <arg name="nom_argument_2">valeur argument 2</arg>
  ...
  <arg name="nom_argument_n">valeur argument n</arg>
  <nagios>
    <directive name="nom_directive_1">valeur directive 1</directive>
    ...
    <directive name="nom_directive_n">valeur directive n</directive>
  </nagios>
</test>
```

La balise `test` permet d'ajouter un test de supervision à l'hôte. Elle possède un attribut `name` obligatoire qui désigne le test de supervision à appliquer (par exemple : "`CPU`" pour superviser l'état du processeur d'un équipement).

Un test accepte généralement zéro, un ou plusieurs arguments, qui doivent être passés dans l'ordre lors de la déclaration du test, à l'aide de la balise `arg`. Chaque argument dispose d'un nom (attribut `name`) et d'une valeur (ou liste de valeurs). Pour spécifier une liste de valeurs pour un argument, la sous-balise `item` doit être utilisée, comme dans l'extrait suivant :

```
<test name="ACL">
    <arg name="policy">whitelist</arg>
    <!--
        L'argument "addresses" est une liste d'adresses IP,
        sous-réseaux ou noms de machines.
    -->
    <arg name="addresses">
        <item>127.0.0.1</item>
        <item>localhost</item>
        <item>192.168.0.0/24</item>
    </arg>
</test>
```

Chaque argument possède un type (spécifié dans la documentation du test). Les types reconnus sont :

- les chaînes de caractères ;
- les nombres entiers ;
- les nombres relatifs (flottants) ;
- les booléens (utiliser la valeur `1`, `true`, `on` ou `yes` pour la valeur `True` ou bien `0`, `false`, `off` ou `no` pour la valeur `False`).

Vous pouvez également, de façon optionnelle, définir des paramètres spécifiques pour la supervision à l'aide de la balise `nagios`, qui contiendra une ou plusieurs directives adressées au moteur de supervision Nagios. Voir la section ci-dessous pour plus d'informations.

Note : Si le même argument est défini deux fois, seule la dernière valeur sera utilisée.

Balise "nagios"

Un bloc de données `nagios` contient des blocs `directive` dont l'attribut `name` appartient à la liste des directives "`host`" de Nagios. La documentation sur ces directives est disponible dans la documentation Nagios.

Syntaxe :

```
<nagios>
  <directive name="max_check_attempts">5</directive>
  <directive name="check_interval">10</directive>
  <directive name="retry_interval">1</directive>
</nagios>
```

Toutes les directives proposées par Nagios sont utilisables ici. Néanmoins, un mauvais réglage des directives peut dégrader sérieusement les performances de la supervision, voire entrainer un dysfonctionnement. L'utilisation des directives est donc à laisser à des utilisateurs avertis.

Un bloc `nagios` peut se trouver au sein d'un bloc `host`/`hosttemplate` ou d'un bloc `test`.

Si la même directive est défini deux fois, la dernière valeur est celle qui sera utilisée.

Balise "`tag`"

Syntaxe :

```
<tag service="service" name="étiquette"/>

<tag service="service" name="étiquette">valeur</tag>
```

La balise `tag` permet d'affecter une étiquette à un hôte ou un service. L'attribut `name` permet de préciser le nom de l'étiquette à ajouter. Il doit correspondre au nom d'une image (**privée de son extension**) à afficher dans VigiMap. Cette image doit se trouver dans `/etc/vigilo/vigimap/public/images/tags`.

L'attribut `service` permet, quant à lui, d'indiquer le nom du service auquel cette étiquette sera affectée. Utilisez la valeur "`host`" si l'étiquette doit porter sur l'hôte en lui-même et non pas sur l'un de ses services.

Enfin, la valeur de l'étiquette est facultative et fait office de méta-donnée. Exemple, pour associer à un hôte l'étiquette de MCO (l'image `mco.png` est fournie dans toute installation standard de Vigilo) :

```
<tag service="host" name="mco"/>
```

Balise "`group`"

Syntaxe :

```
<group>/Chemin complet/vers le/Groupe</group>

<group>Nom de groupe</group>
```

La balise `group` permet d'indiquer les groupes métiers auxquels cet équipement appartient. Les groupes sont organisés de manière hiérarchique (sous la forme d'un arbre).

La première forme (chemin absolu) permet de se déplacer dans cette hiérarchie en donnant le chemin complet jusqu'au groupe, de la racine de l'arbre vers les feuilles. Chaque élément du chemin est précédé du symbole "`/`". Si le nom de l'élément contient un "`/`" ou un "`\`", vous devez le faire précéder du caractère d'échappement "`\`". Ainsi, l'élément "`Serveurs Linux/Unix`" sera écrit dans les chemins comme "`Serveurs Linux\/Unix`".

La seconde forme (chemin relatif) permet d'ajouter l'équipement à tous les groupes dont le nom vaut celui indiqué, quelque soit leur position dans l'arbre. Il n'est pas possible de préciser plusieurs éléments (par exemple "`A/B`") lorsque cette forme est utilisée. Les règles d'échappement de la première forme s'appliquent également ici.

Note : Les groupes sont utilisés pour décider de la ventilation des équipements sur les différents groupes de supervision. Une fois tous les groupes exprimés sous la forme d'un chemin absolu, VigiConf suppose que le premier élément du chemin correspond au groupe à utiliser pour la ventilation. En cas de conflit, ou pour placer l'équipement dans un autre groupe de ventilation que celui déterminé automatiquement, vous devez utiliser l'attribut *ventilation* de la balise *host* afin de spécifier manuellement le groupe de ventilation à utiliser.

Remarques

De même que pour les tests, l'hôte peut disposer de directives de configuration spécifiques destinées à Nagios. Celles-ci seront groupées sous une balise `nagios` (voir également la documentation concernant la balise `test` ci-dessus).

Chaque hôte hérite implicitement d'un modèle appelé "`default`". Toutes les directives définies dans le modèle "`default`" sont donc appliquées à la configuration des différents hôtes, et ce même si ces hôtes n'indiquent pas explicitement qu'ils utilisent ce modèle, via la balise `<template>`. Voir le chapitre pour plus d'information.

Configuration des modèles d'hôtes

Le dossier "`hosttemplates`" contient les fichiers de définition des modèles d'hôtes. Un modèle d'hôtes permet de regrouper un ensemble d'éléments de configuration communs à plusieurs hôtes. Les hôtes peuvent ensuite être déclarés comme héritant de ce modèle. Tous les fichiers XML de ce dossier sont analysés et doivent contenir la définition d'un ou plusieurs modèles.

La balise à la racine de ce document se nomme "`hosttemplates`" et peut contenir un ou plusieurs blocs "`hosttemplate`", correspondant chacun à la définition d'un hôte.

Le fragment de code suivant rappelle la structure générale du fichier :

```
<?xml version="1.0"?>
<hosttemplates>
    <hosttemplate attr1="..." attr2="..." attrN="..."> ... </hosttemplate>
    <hosttemplate attr1="..." attr2="..." attrN="..."> ... </hosttemplate>
...
</hosttemplates>
```

Un bloc de données `hosttemplate` possède les attributs suivants :

— `name` : Nom du modèle.

Un bloc de données *hosttemplate* contient les blocs suivants, en respectant l'ordre :

— `parent` (0 ou un)
— `attribute` (0 ou plus)
— `tag` (0 ou plus)
— `group` (0 ou plus)
— `class` (0 ou plus)
— `test` (0 ou plus)
— `nagios` (0 ou un)
— `item` (0 ou plus)

Balise "`parent`"

Un bloc de données `parent` contient une simple chaîne de caractères, le nom du template dont ce template hérite. Il est possible de créer autant de niveaux d'héritage de templates que souhaité et chaque template peut hériter d'un ou plusieurs templates (héritage multiple).

Exemple :

```
<parent>generic</parent>
```

Balise “attribute“

Un bloc de données attribute possède un attribut : name.

Un bloc de données attribute contient une valeur de type chaîne de caractères.

Exemple :

```
<attribute name="snmpOIDsPerPDU">10</attribute>
```

Balise “tag“

Syntaxe :

```
<tag service="service" name="étiquette"/>

<tag service="service" name="étiquette">valeur</tag>
```

La balise tag permet d'affecter une étiquette à un hôte ou un service. L'attribut name permet de préciser le nom de l'étiquette à ajouter. Il doit correspondre au nom d'une image (**privée de son extension**) à afficher dans VigiMap. Cette image doit se trouver dans /etc/vigilo/vigimap/public/images/tags.

L'attribut service permet, quant à lui, d'indiquer le nom du service auquel cette étiquette sera affectée. Utilisez la valeur “host” si l'étiquette doit porter sur l'hôte en lui-même et non pas sur l'un de ses services.

Enfin, la valeur de l'étiquette est facultative et fait office de méta-donnée. Exemple, pour associer à un hôte l'étiquette de MCO (l'image mco.png est fournie dans toute installation standard de Vigilo) :

```
<tag service="host" name="mco"/>
```

Balise “group“

Un bloc de données group contient une chaîne de caractères.

Exemple :

```
<group>AIX servers</group>
```

Balise “class“

Un bloc de données class contient une simple chaîne de caractère.

Exemple :

```
<class>aix</class>
```

Balise "test"

Un bloc de données test possède l'attribut suivant :
— name (1 exactement)
Un bloc de données test contient les blocs suivants, dans l'ordre :
— arg (0 ou plus)
— nagios (0 ou 1)
Reportez-vous à la documentation sur la *configuration des tests d'un hôte* pour plus d'information sur les attributs et blocs acceptés par cette balise.

Exemple :

```
<test name="Proc">
  <arg name="label">aixmibd</arg>
  <arg name="processname">.*aixmibd .*</arg>
  <nagios> ... </nagios>
</test>
```

Balise "nagios"

Voir la définition utilisée pour la balise host : *Balise "nagios"*.

Balise "item"

Un bloc de données item contient une simple chaîne de caractère.

Exemple :

```
<item>item1</item>
```

Cas particulier du modèle "default.xml"

Le modèle nommé "default.xml" est un cas particulier de modèle. Il est appliqué systématiquement à tous les hôtes configurés.

Par défaut, ce modèle contient des attributs destinés à configurer le comportement de Nagios et à renseigner certaines informations relatives à la configuration de SNMP sur le parc. Il contient également le test "UpTime" qui envoie une alerte lorsque la durée de fonctionnement d'une machine est trop faible (c'est-à-dire lorsqu'elle a été redémarrée récemment).

Dossier "groups"

Le dossier "groups" contient les fichiers de définition des groupes d'éléments supervisés. Tous les fichiers XML de ce dossier sont analysés et doivent contenir la définition d'un ou plusieurs groupes.

L'utilisation de groupes facilite la gestion au quotidien des permissions (application en masse d'une permission sur un groupe d'éléments supervisés à un groupe d'utilisateurs).

La balise à la racine de ce document se nomme "groups" et peut contenir un ou plusieurs blocs "group", correspondant chacun à la définition d'un groupe d'éléments supervisés.

Le fragment de code suivant rappelle la structure générale du fichier :

```
<?xml version="1.0"?>
<groups>
  <group attr1="..." attr2="..." attrN="..."> ... </group>
  <group attr1="..." attr2="..." attrN="..."> ... </group>
  ...
</groups>
```

Balise "group"

Un bloc de données `group` possède un attribut : `name`.

Le bloc de données `group` peut possèder un ou plusieurs sous-blocs `group`. Cette imbrication peut être répétée autant de fois que nécessaire et permet de construire une hiérarchie de groupes. Cette hiérarchie est ensuite utilisée dans les différentes interfaces, pour la gestion des permissions, ou encore, pour organiser les informations.

Exemple :

```
<group name="group1" />

<group name="group2" >
  <group name="group21"/>
  ...
</group>
```

Le même nom de groupe ne peut pas être utilisé plusieurs fois au même niveau dans la hiérarchie des groupes. C'est-à-dire que l'exemple suivant est interdit :

```
<!-- Attention, cet exemple ne fonctionne pas ! -->
<group name="group1">
    <group name="group2"/>
    <group name="group2"/>
</group>
```

En revanche, le même nom de groupe peut être utilisé dans des endroits séparés de l'arborescence, comme dans l'exemple ci-dessous :

```
<group name="group1">
    <group name="group1">
        <group name="group1"/>
    </group>
</group>
```

Notez que chacun des "`group1`" correspond à un groupe différent.

Dossier "hlservices"

Cette fonctionnalité n'est disponible que dans la version Enterprise de Vigilo.

5.1.4 Configurations particulières

Ce chapitre recense des cas particuliers de configuration. Pour chaque cas, un exemple de configuration associée est donné.

Utilisation de SNMPv3

L'utilisation de SNMP version 3 nécessite un peu plus de configuration que pour SNMP version 1 ou 2c. En effet, en plus de la définition de la communauté de l'équipement, il est nécessaire de spécifier les éléments de sécurité permettant d'authentifier la connexion à l'équipement et/ou d'assurer la confidentialité des échanges SNMP avec l'équipement.

Aucun test particulier n'est à appliquer pour utiliser SNMPv3 dans Vigilo. Cependant, des attributs supplémentaires doivent être définis au niveau de la configuration de l'équipement fonctionnant avec SNMPv3.

Le listing suivant présente un exemple de configuration d'un hôte devant être interrogé en utilisant SNMPv3 :

```
<attribute name="snmpVersion">3</attribute>
<attribute name="snmpSeclevel">authPriv</attribute>
<attribute name="snmpAuthproto">MD5</attribute>
<attribute name="snmpPrivproto">DES</attribute>
<attribute name="snmpSecname">snmpuser1</attribute>
<attribute name="snmpAuthpass">12345678</attribute>
<attribute name="snmpPrivpass">12345678</attribute>
```

Les noms des attributs suivent la terminologie standard de SNMPv3, excepté pour le préfixe "`snmp`" qui ne sert qu'à empêcher d'éventuels conflits de nommage avec d'autres attributs similaires.

Le rôle de chacun de ces attributs est précisé ci-dessous :

- L'attribut "`snmpVersion`" indique que c'est la version 3 du protocole SNMP qui doit être utilisée pour interroger l'équipement.
- L'attribut "`snmpSeclevel`" indique les contraintes de sécurité à apporter sur les communications avec l'équipement. Les valeurs possibles sont :
  - "`noAuthNoPriv`" (no authentication / no privacy) : aucune authentification n'a lieu et les échanges ne sont pas chiffrés. Cette valeur correspond à ce qui est fourni par SNMPv1 et SNMPv2c et n'offre **aucune sécurité**. Il est donc recommandé de **ne pas utiliser cette valeur** (sauf cas exceptionnels).
  - "`authNoPriv`" (authentication / no privacy) : Vigilo s'authentifiera auprès de l'équipement avec un nom d'utilisateur et un mot de passe dédiés. Les échanges avec l'équipement ne seront pas chiffrés et les communications pourront donc être écoutées par n'importe quelle personne disposant d'un accès physique au réseau.
  - "`authPriv`" (authentication / privacy) : Vigilo s'authentifiera auprès de l'équipement avec un nom d'utilisateur et un mot de passe dédiées. De plus, tous les échanges seront chiffrés. Cette valeur est celle offrant le plus de sécurité et est donc recommandée.
- L'attribut "`snmpAuthproto`" est obligatoire lorsque "`snmpSeclevel`" vaut "`authNoPriv`" ou "`authPriv`" et spécifie l'algorithme utilisé pour masquer le mot de passe lors des transmissions avec l'équipement. Les valeurs possibles sont "`MD5`" et "`SHA1`".
- L'attribut "`snmpPrivproto`" est obligatoire lorsque "`snmpSeclevel`" vaut "`authPriv`" et spécifie l'algorithme de chiffrement utilisé pour les échanges avec l'équipement. Les valeurs possibles sont "`DES`" et "`AES`".
- L'attribut "`snmpSecname`" est obligatoire lorsque "`snmpSeclevel`" vaut "`authNoPriv`" ou "`authPriv`" et indique le nom d'utilisateur avec lequel Vigilo s'authentifiera auprès de l'équipement.
- L'attribut "`snmpAuthpass`" est obligatoire lorsque "`snmpSeclevel`" vaut "`authNoPriv`" ou "`authPriv`" et indique le mot de passe à utiliser pour s'authentifier auprès de l'équipement. Il peut être donné sous forme textuelle (comme dans l'exemple ci-dessus) ou sous forme de chaîne de caractères hexadécimaux en préfixant la valeur par "`0x`".
- L'attribut "`snmpPrivpass`" est obligatoire lorsque "`snmpSeclevel`" vaut "`authPriv`" et spécifie la clé de chiffrement utilisée pour les échanges avec l'équipement. Elle peut être donnée sous forme textuelle (comme dans l'exemple ci-dessus) ou sous forme de chaîne de caractères hexadécimaux en préfixant la valeur par "`0x`".

5.1.5 Configuration des journaux

VigiConf est capable de transmettre un certain nombre d'informations au cours de son fonctionnement à un mécanisme de journalisation des événements (par exemple, des journaux systèmes, une trace dans un fichier, un enregistrement des événements en base de données, etc.).

La syntaxe de la configuration des journaux est la même que pour le reste de la configuration. Les mécanismes utilisés pour gérer les journaux sont ceux mis à disposition par le module `logging` de Python.

Le reste de cette chapitre présente les informations basiques concernant la configuration des journaux. La documentation complète du format de configuration est disponible à l'adresse http://docs.python.org/library/logging.html#configuration-file-format

La configuration de la journalisation se fait en manipulant trois types d'objets :

— les `loggers`, qui permettent de configurer les modules python qui devront être journalisés et de définir les gestionnaires d'événements à appliquer ;
— les `handlers` (gestionnaires d'événements), qui indiquent le traitement qui doit être appliqué aux événements reçus (enregistrement dans un fichier, dans les journaux Syslog du système, stockage en base de données, etc.) ;
— les `formatters`, qui permettent de configurer le formatage appliqué aux événements.

Les sections qui suivent vous donnent des informations sur la configuration de chacun de ces éléments.

Configuration des `loggers`

Les loggers permettent d'indiquer les sources d'événements pour lesquelles les événements de journalisation seront effectivement consignés. Ils permettent également de définir les traitements qui seront appliqués à ces événements (pour définir par exemple la destination des événements).

La configuration des loggers se fait en deux étapes :

— dans un premier temps, les loggers sont déclarés,
— puis, les loggers déclarés sont configurés.

Déclaration d'un `logger`

La déclaration des loggers se fait sous la forme d'une liste des noms des loggers, contenue sous la clé « keys » de la section « loggers ».

Les loggers sont organisés sous une forme hiérarchique, qui suit l'organisation des modules Python sur le disque dur. Si aucun logger n'est défini pour traiter les messages d'un module, le logger défini sur le niveau supérieur est utilisé. L'arborescence du système de fichiers est ainsi parcourue jusqu'à ce qu'un `logger` intercepte le message ou qu'on ne puisse plus remonter.

Si aucun `logger` n'est trouvé après le parcours de la hiérarchie, un message d'avertissement est affiché sur la console afin d'inciter l'utilisateur à configurer correctement les journaux.

Par ailleurs, un `logger` spécial est défini qui apparaît toujours au sommet de la hiérarchie des loggers, nommé « root ».

Exemple de parcours de la hiérarchie des loggers :

Si le module Python `vigilo.common.conf` émet un événement et qu'aucun logger n'est associé à ce module, le gestionnaire de journaux recherche un logger pour le module `vigilo.common`. S'il n'en trouve pas, il recherche un logger pour le module `vigilo`. Si après toutes ces recherches il n'en trouve toujours pas, il recherche le logger `root`.

Configuration d'un `logger`

Chaque logger déclaré doit ensuite être configuré. La configuration d'un logger appelé `<nom_du_logger>` se fait en ajoutant une section appelée `logger_<nom_du_logger>` dans le fichier de configuration.

Pour chaque logger, les clés suivantes sont configurables :

- `level` (obligatoire) : niveau de criticité à partir duquel les messages seront pris en compte. Les niveaux utilisables sont (par ordre de criticité croissante) :
  - NOTSET pour que l'événement soit toujours pris en compte, quelle que soit sa criticité,
  - DEBUG pour ne prendre en compte que les événements au niveau « débogage » ou supérieur,
  - INFO pour ne prendre en compte que les événements au niveau « information » ou supérieur,
  - WARNING pour ne prendre en compte que les événements au niveau « avertissement » ou supérieur,
  - ERROR pour ne prendre en compte que les événements au niveau « erreur » ou supérieur,
  - CRITICAL pour ne prendre en compte que les événements au niveau « critique ».
- `qualname` optionnel le nom du module Python qui sert de source pour les événements enregistrés par ce logger. Si ce champ est omis, c'est que le logger se trouve au sommet de la hiérarchie, et donc, qu'il s'agit du logger « root ».
- `propagate` optionnel un entier qui indique si l'événement doit être propagé (1) aux loggers situés « au-dessus » dans la hiérarchie ou non (0). Par défaut, les messages sont propagés.
- `handlers` obligatoire une liste contenant les noms des handlers gestionnaires d'événements qui traiteront les événements capturés par ce logger.

Exemple de configuration d'un `logger`

L'encadré suivant présente un exemple de configuration d'un `logger` nommé « log ». Cet exemple suppose par ailleurs qu'un `handler` « hand » a été défini dans la configuration.

```
[loggers]
keys=log

[logger_log]
level=WARNING
qualname=vigilo
handlers=hand
```

Ce logger intercepterait tous les événements émis par un composant de Vigilo et les enverrait au handler « hand » pour qu'ils soient traités.

Configurations des `handlers` (gestionnaires d'événements)

Les handlers permettent d'indiquer comment sont traités les événements reçus. Ils permettent par exemple d'indiquer que les messages doivent être stockés dans une base de données, dans les journaux systèmes, envoyés par email, etc.

La configuration des handlers se fait en deux étapes :

- dans un premier temps, les handlers sont déclarés,
- puis, les handlers déclarés sont configurés.

Déclaration d'un `handler`

La déclaration des handlers se fait sous la forme d'une liste des noms des handlers, contenue sous la clé « keys » de la section « handlers ».

Configuration d'un `handler`

Chaque handler déclaré doit ensuite être configuré. La configuration d'un handler appelé <nom_du_handler> se fait en ajoutant une section appelée « handler_<nom_du_handler> » dans le fichier de configuration.

Pour chaque handler, les clés suivantes sont configurables :

- `class` (obligatoire) : la classe Python qui effectuera le stockage des messages. Les plus couramment utilisées sont « handlers.SysLogHandler » (enregistrement vers les journaux systèmes de Linux), « StreamHandler » (enregistrement dans un flux, comme par exemple la sortie standard du programme). La liste complète des handlers utilisables est disponible sur http://docs.python.org/library/logging.html
- `args` (optionnel) : arguments qui seront passés au constructeur de la classe désignée par la clé `class`.
- `level` (obligatoire) : le niveau de criticité à partir duquel les messages seront pris en compte. Les niveaux utilisables sont les mêmes que pour les loggers.
- `formatter` (obligatoire) : le nom du `formatter` à utiliser pour mettre en forme les messages.

Exemple de configuration d'un `handler`

L'encadré suivant présente un exemple de configuration d'un handler nommé « hand ». Cet exemple suppose par ailleurs qu'un `formatter` « fmt » a été défini dans la configuration.

```
[handlers]
keys=hand

[handler_hand]
class=handlers.SysLogHandler
args='/dev/log', 'daemon'
level=NOTSET
formatter=fmt
```

Ce handler enregistre les événements dans le journal du système, quelque soit leur niveau de criticité, en utilisant le `formatter` « fmt ».

Configuration des `formatters`

Les formatters permettent de mettre en forme les messages avant leur stockage. La configuration des formatters se fait en deux étapes :

- dans un premier temps, les formatters sont déclarés,
- puis, les formatters déclarés sont configurés.

Déclaration d'un `formatter`

La déclaration des formatters se fait sous la forme d'une liste des noms des formatters, contenue sous la clé « keys » de la section « formatters ».

Configuration d'un `formatter`

Chaque formatter déclaré doit ensuite être configuré. La configuration d'un formatter appelé <nom_du_formatter> se fait en ajoutant une section appelée «formatter_<nom_du_formatter> » dans le fichier de configuration.

Pour chaque formatter, les clés suivantes sont configurables :

— `format` (obligatoire) : un texte qui décrit la mise en forme du message. Ce texte peut faire références à des informations contextuelles concernant l'événement à enregistrer. Ces informations contextuelles sont insérées en utilisant le mécanisme de formatage des chaines de caractères de Python. Par exemple, le message original de l'événement peut être inséré dans le texte qui sera effectivement enregistré à l'aide de la substitution suivante

```
%(message)s
```

La liste des informations contextuelles les plus fréquemment utilisées est fournie dans la suite de ce document. La liste complète peut être consultée en lisant la documentation de Python concernant le module `logging`.

— `datefmt` (optionnel) : un texte indiquant le format à utiliser pour les dates. La description complète du format est présentée dans la documentation de Python (http://docs.python.org/library/time.html#time.strftime). Par défaut, le format ISO 8601 est utilisé.

Informations contextuelles disponibles pour les `formatters`

Plusieurs informations contextuelles peuvent être insérées automatiquement par les formatters avant l'envoi d'un message dans les journaux.

Les informations contextuelles suivantes sont disponibles :

— La date à laquelle l'événement est survenu peut être insérée, dans un format intelligible, grâce à la substitution suivante

```
%(asctime)s
```

— La criticité du message reçu (DEBUG, WARNING, ERROR, etc.) peut être insérée sous forme textuelle grâce à la substitution suivante

```
%(levelname)s
```

— Le nom du module qui a émis le message peut être inséré grâce à cette substitution

```
%(module)s
```

— Le message original associé à l'événement peut être inséré grâce à cette substitution

```
%(message)s
```

Cette liste ne contient que les informations les plus couramment utilisées. La liste complète est disponible sur http://docs.python.org/library/logging.html#formatter.

Exemple de configuration d'un `formatter`

L'encadré suivant présente un exemple de configuration d'un formatter nommé « fmt ».

```
[formatters]
keys=fmt

[formatter_fmt]
format=[%(asctime)s] %(levelname)s - %(message)s
datefmt=%a, %d %b %Y %H:%M:%S
```

Ce formateur affiche le niveau de criticité (sous forme textuelle) ainsi que le message de l'événement original. Le message est précédé de la date.

Un exemple de message enregistré en utilisant ce formateur serait :

```
[Thu, 28 Jun 2001 14:17:15] ERROR - Could not connect to memcached.
```

Exemple de configuration complète des journaux

L'encadré qui suit donne un exemple simple mais complet d'une configuration des journaux dans le cas du corrélateur de Vigilo. Le résultat de cette configuration est ensuite détaillé.

```
[loggers]
keys = root,rules

[handlers]
keys = syslog

[formatters]
keys = generic

[logger_root]
level = WARNING
handlers = syslog

[logger_rules]
level = INFO
handlers = syslog
qualname = vigilo.correlator.rules

[handler_syslog]
class=handlers.SysLogHandler
level=DEBUG
formatter=nullFormatter
args='/dev/log', 'daemon'

[formatter_generic]
format = %(message)s
datefmt =
```

Dans cette configuration, tous les messages émis par le corrélateur et qui ont un niveau de criticité au moins équivalent à « WARNING » sont enregistrés dans les journaux du système (syslog).

Par ailleurs, les messages issus des règles de corrélation (module Python `vigilo.correlator.rules`) sont consignés dès lors que leur niveau de criticité est supérieur ou égal à « INFO ».

Le message enregistré a un formatage « brut » où seul le message apparaît. En effet, dans le cas d'un handler de type « handlers.SysLogHandler », il n'est pas nécessaire d'ajouter des informations supplémentaires (nom et PID du processus qui a envoyé l'événement, heure où l'événement a été consigné, etc.). Le système ajoute automatiquement ces informations à l'événement.

5.1.6 Utilisation de l'utilitaire "`vigiconf`"

La génération des fichiers de configuration des différentes applications en charge de la supervision du parc se fait en utilisant l'utilitaire "`vigiconf`" fourni lors de l'installation du paquet portant le même nom.

Cet utilitaire permet d'effectuer les opérations suivantes :

- Affichage des informations concernant la configuration actuelle ;
- Gestion des applications ;
- Retour arrière de la configuration ;
- Déploiement d'une nouvelle configuration ;
- Découverte des services présents sur le réseau.

La commande "`vigiconf --help`" permet d'obtenir l'aide de l'utilitaire. Elle affiche la sortie suivante :

```
usage : vigiconf [-h] [--debug] {info,server-status,discover,deploy} ...

Gestionnaire de configuration Vigilo

arguments optionnels:
-h, --help affiche ce message d'aide et quitte
--debug Mode de débogage

Sous-commandes: {info,server-status,discover,deploy}

info           Affiche un résumé de la configuration actuelle.
deploy         Déploie la configuration sur chaque serveur si la
               configuration a changé.
discover       Découvrir les services disponibles sur un serveur
               distant.
server-status  Active ou désactive un serveur Vigilo.
```

Ces différentes opérations sont détaillées dans les sections qui suivent.

Vous pouvez obtenir l'aide sur une commande en tapant "`vigiconf nom_de_la_commande --help`".

Affichage des informations

L'affichage des informations concernant la configuration actuelle se fait en exécutant la commande "`vigiconf info`".

La commande renvoie plusieurs informations, comme dans l'exemple d'exécution suivant :

```
VigiConf a été lancé depuis le compte 'root'. Utilisation du compte 'vigiconf' à la␣
↪place.
Révision actuelle dans le dépôt : 358
Serveur supserver1.example.com:
    déployé: 358
    installé: 358
    précédent: 358
Serveur supserver2.example.com:
    déployé: 358
    installé: 358
    précédent: 358
    DÉSACTIVÉ
VigiConf se termine
```

La révision actuelle correspond à la dernière version de la configuration validée sur la machine qui héberge VigiConf.

Les lignes suivantes affichent, pour chaque serveur de supervision : son nom, éventuellement son état (lorsque le serveur est désactivé), ainsi que les numéros de la révision de la configuration actuellement déployée (donc en production), de la dernière révision installée et de la précédente révision installée.

Vous pouvez préciser les noms des serveurs de supervision dont les informations doivent être affichés en passant simplement leur nom comme argument de la commande.

Note : Toutes les valeurs sont à 0 lors d'une nouvelle installation (car aucun des serveurs n'a encore été configuré).

Déploiement de la configuration

Une fois la configuration des différents éléments du parc effectuées, vous pouvez la déployer sur les divers machines de la supervision à l'aide de la commande "`vigiconf deploy`".

Vous pouvez passer en argument la liste des noms des serveurs de supervision pour lesquels une nouvelle version de la configuration doit être déployée.

Découverte des services à superviser sur un hôte

Afin d'assister l'administrateur dans sa configuration des tests de supervision d'un serveur, VigiConf dispose d'un mode de découverte automatique des services disponibles sur un hôte.

La commande "`vigiconf discover`" prend en arguments les noms d'hôtes (entièrement qualifiés) ou les fichiers de scan à analyser afin de découvrir les services à superviser. Les fichiers de scan doivent avoir été générés à l'aide de la commande "`snmpwalk`" sur l'OID "`.1`" en lui passant les options "`-OnQe`".

Exemple de commande pour créer le fichier de scan :

```
SNMPCONFPATH=/dev/null snmpwalk -OnQe -v2c -c public hostname .1 \
    > hostname-OnQe.walk
```

Note : "`SNMPCONFPATH=/dev/null`" permet d'éviter l'import d'options de configuration depuis des fichiers du système tels que :

- `/etc/snmp/snmp.local.conf`
- `/etc/snmp/snmp.conf`

Par défaut, cette commande affiche sur la sortie standard le fichier de configuration XML correspondant aux équipements analysés. Le listing suivant présente le résultat d'une telle analyse :

```
<?xml version="1.0"?>
<host address="127.0.0.1" name="localhost.localdomain">
  <group>Servers</group>
  <class>ucd</class>
  <test name="Users" />
  <test name="Partition">
    <arg name="partname">/home</arg>
    <arg name="label">/home</arg>
  </test>
  <test name="Partition">
    <arg name="partname">/</arg>
    <arg name="label">/</arg>
  </test>
  <test name="TotalProcesses" />
  <test name="Swap" />
  <test name="CPU" />
  <test name="Load" />
  <test name="RAM" />
  <test name="UpTime" />
  <test name="TCPConn" />
  <test name="InterrCS" />
</host>
```

Le résultat de cette commande peut être enregistré directement dans un fichier de configuration XML afin d'être lu ensuite par VigiConf en utilisant l'option "`-o`" suivie de l'emplacement du fichier dans lequel le résultat doit être sauvegardé.

La découverte peut-être limitée à un test en particulier via l'option "`-t`".

Activation / désactivation d'un serveur de supervision

La commande "`vigiconf server-status`" permet d'activer ou de désactiver un ou plusieurs serveurs de la plate-forme de supervision. Elle doit être appelée avec au moins 2 arguments.

Le 1er argument indique l'action à effectuer et peut valoir :
- "`enable`" pour activer un ou plusieurs serveurs de supervision précédemment désactivés.
- "`disable`" pour désactiver un ou plusieurs serveurs de supervision.

5.1.7 Annexes

Tests de supervision disponibles dans Vigilo

Le tableau suivant recense les tests de supervision disponibles dans Vigilo.

Tableau 5.1 – Liste des tests de supervision disponibles dans Vigilo

Nom du test	Classes disponibles	Détails
Collector	— *Toutes*	The Collector service
CPU	— linux — ucd	Check the CPU usage of a host (service only) Paramètres acceptés par le test : `crit` : CRITICAL threshold (unused, for compatibility) `warn` : WARNING threshold (unused, for compatibility)
DiskIO	— ucd	Monitor the disks Input/Output Paramètres acceptés par le test : `crit` : tuple containing the thresholds for CRITICAL status. Ex : (limit_reads, limit_writes) `diskname` : disk name. Default : hdisk0 `warn` : tuple containing the thresholds for WARNING status. Ex : (limit_reads, limit_writes)
HTTP	— *Toutes*	Check if the HTTP service is up
HTTP_url	— *Toutes*	Check if the HTTP service is up for a given url Paramètres acceptés par le test : `port` : Le port sur lequel le serveur web écoute (80 par défaut). `service` : Le nom du service à utiliser dans Nagios.
HTTPS	— *Toutes*	Check if the SSL certificates on the HTTPS server are OK
Interface	— *Toutes*	Check the status of an interface, and graph its throughput Paramètres acceptés par le test : `counter32` : to use Counter32bits specifically for this interface. `crit` : CRITICAL threshold. See description for the format. `errors` : create a graph for interface errors `ifname` : SNMP name for the interface `label` : Label to display `max` : the maximum bandwidth available through this interface, in bits/s `staticindex` : consider the ifname as the static SNMP index instead of the interface name. It's not recommanded, but it can be necessary as some OS (Windows among others) assign the same name to different interfaces. `teststate` : Used to deactivate the interface state control. (When you only need statistics.)

Messages d'erreurs/d'alerte/d'informations

Ce chapitre recense les messages d'erreurs les plus courants que vous êtes susceptibles de rencontrer, ainsi que la méthode de résolution de ces problèmes.

VigiConf n'a pas été lancé en utilisant le compte 'vigiconf'. Abandon. Afin de garantir la sécurité du système, VigiConf doit être exécuté en tant qu'utilisateur "`vigiconf`". Ce message d'erreur s'affiche lorsque cette condition n'est pas remplie et VigiConf refuse de s'exécuter. Le compte "`root`" peut également être utilisé pour exécuter VigiConf (dans ce cas, l'application commencera par basculer vers le compte "`vigiconf`" avant de lâcher ses privilèges). L'utilisation du compte "`root`" pour démarrer VigiConf en production est déconseillée.

VigiConf a été lancé depuis le compte "`root`" (super-utilisateur). Utilisation du compte 'vigiconf' à la place. Ce message d'avertissement est affiché lorsque VigiConf est exécuté depuis le compte "`root`".

Glossaire - Terminologie

Ce chapitre recense les différents termes techniques employés dans ce document et donne une brève définition de chacun de ces termes.

XML eXtensible Markup Language. Langage de balisage extensible.

URL Uniform Resource Locator. Chaîne de caractères permettant d'identifier une ressource sur Internet.

SGBD Système de Gestion de Base de Données. Logiciel permettant d'héberger une base de données sur la machine.

SVN Subversion, système de contrôle de versions. URL : http://subversion.apache.org

5.2 Guide d'utilisation

5.2.1 Démarrage rapide

VigiBoard fournit une interface graphique accessible depuis un navigateur Web. La configuration par défaut est suffisante pour un démarrage rapide.

Accès à l'interface

L'utilisation de VigiBoard se fait simplement en accédant, via votre navigateur, à l'adresse indiquée par votre administrateur. Par exemple : http://supervision.example.com/vigiboard/.

Authentification

Note : Dans le cas où un mécanisme d'authentification externe a été défini par votre administrateur, il se peut qu'aucune authentification ne vous soit demandée, même lorsqu'il s'agit de votre première connexion. Le reste de ce chapitre décrit le cas où une authentification interne a lieu et ne s'applique donc pas au cas de figure cité ci-dessus. Contactez votre administrateur pour plus d'information sur la configuration utilisée.

Si vous ne vous êtes jamais connecté sur VigiBoard ou si vous n'êtes plus authentifié, le formulaire d'authentification de la figure suivante s'affiche :

Fig. 5.1 – Écran d'authentification.

Selon le compte utilisateur auquel vous vous connecterez, vous disposerez d'un accès à plus ou moins d'hôtes et de services (et donc d'informations). Les données d'authentification demandées ici vous ont normalement été transmises par votre administrateur.

— Saisir les données d'authentification en renseignant les zones de saisie « Identifiant » et « Mot de passe ».
— Valider la saisie en cliquant sur le bouton « Connexion » (entouré en rouge sur la figure suivante).

Fig. 5.2 – Bouton de validation des données d'authentification.

En cas de succès, le bac à événements s'affiche. Sinon, le formulaire d'authentification s'affiche à nouveau, avec un message précisant la nature de l'erreur :

Fig. 5.3 – Formulaire après un échec de l'authentification.

5.2.2 Fonctionnalités

Tableau des évènements

Une fois la page d'authentification passée, le tableau des évènements s'affiche, comme sur la figure qui suit. Il s'agit de l'interface principale de VigiBoard.

Chaque ligne de ce tableau représente l'un des évènements corrélés (incidents) détectés par Vigilo. Les informations suivantes y sont détaillées :

- L'état de l'hôte ou du service sur lequel l'évènement est survenu, représenté par une icône en début de ligne, dont la signification est détaillée dans le tableau ci-après ;
- La date à laquelle il a été détecté ;
- La priorité avec laquelle il devrait être traité (sa gravité) ;
- Le nombre d'alertes remontées par Vigilo le concernant ;
- Le nom de l'hôte (et éventuellement du service) sur lequel il est survenu ;
- Le message d'alerte ;
- Les services de haut niveau impactés ;
- Le numéro de ticket associé ;
- Le statut de sa prise en compte par les opérateurs de supervision.

Tableau 5.2 – Signification des icônes d'état

Icône	État correspondant pour un hôte	État correspondant pour un service
	Down (En panne)	Critical (Critique)
	Unreachable (Injoignable)	Unknown (Inconnu)
	**n/a**	Warning (Avertissement)
	Up (Fonctionnel)	OK (OK)

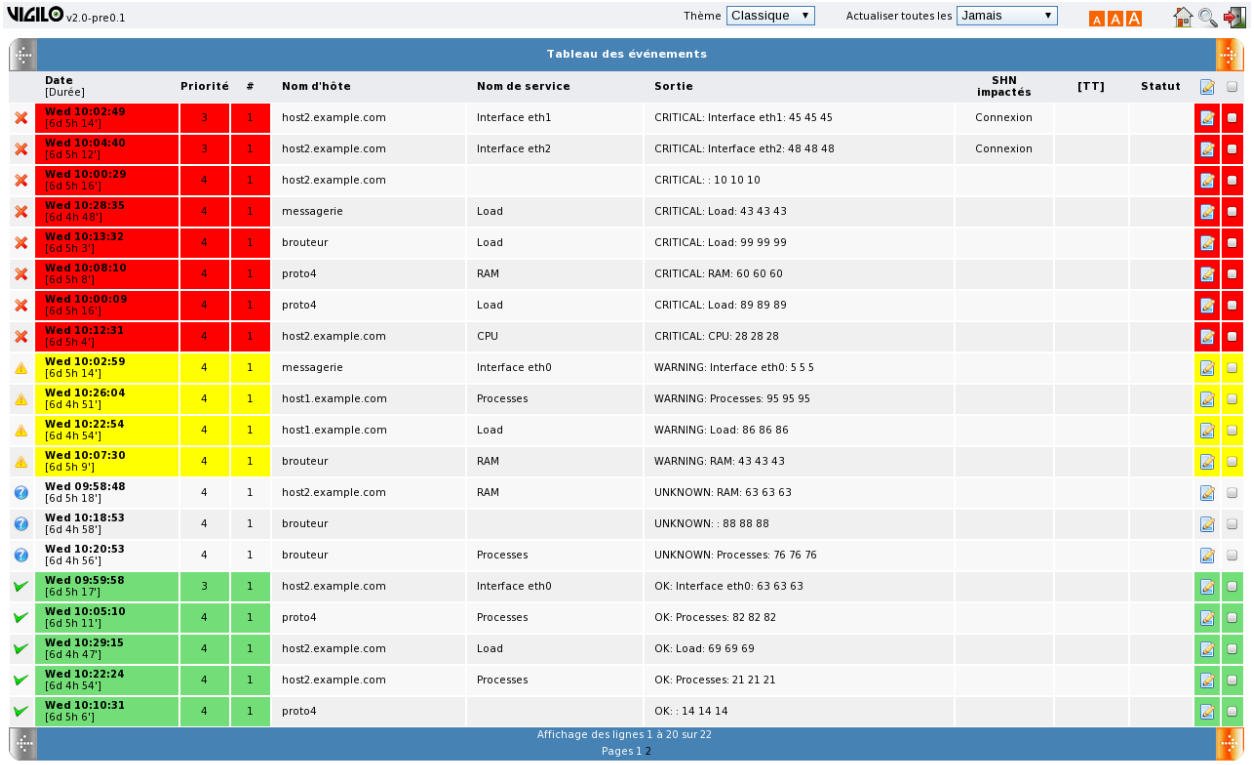
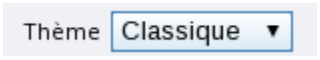


Fig. 5.4 – Tableau des évènements.

Choix d'un thème

Plusieurs thèmes sont disponibles par défaut dans VigiBoard, dont un destiné aux personnes malvoyantes et un autre prévu pour un usage dans des conditions de faible luminosité. Un menu déroulant situé en haut de l'interface permet de passer de l'un à l'autre sans avoir à recharger la page.

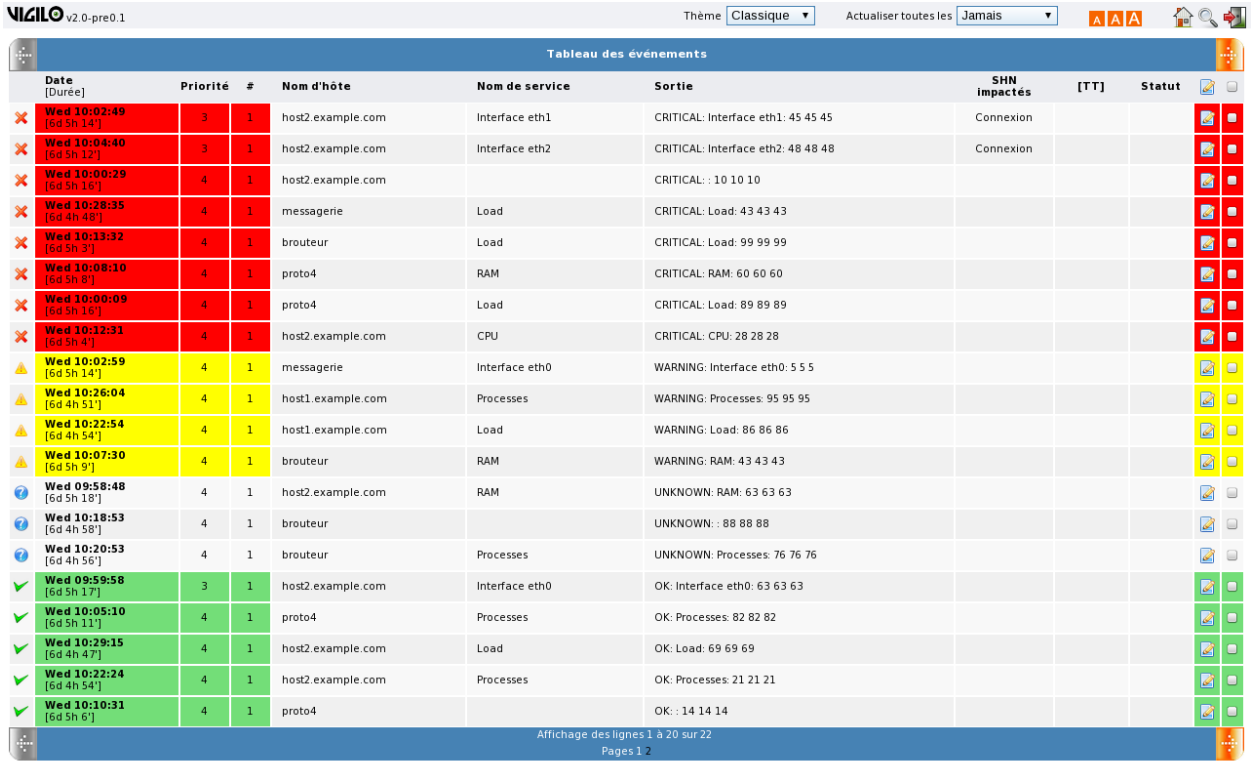


Le réglage du thème est ensuite conservé dans les autres écrans de l'application.

Thème Classique

Le thème *Classique* est utilisé par défaut dans VigiBoard. Il est particulièrement adapté aux consoles de supervision grâce à son code couleur qui fait ressortir l'état courant de l'hôte ou du service concerné par un événement :

- Le rouge indique que l'hôte ou le service concerné est dans un état critique (hôte en panne, service inopérant, etc.) ;
- Le jaune indique un état dans lequel l'hôte ou le service concerné lève un avertissement (temps de réponse plus lent que la normale, consommation excessive de ressources, etc.) ;
- L'absence de couleur signifie un état inconnu (impossibilité de contacter l'hôte ou de tester l'état du service par exemple) ;
- Le vert représente un état dans lequel l'hôte ou le service concerné est fonctionnel (état nominal).



Date [Durée]	Priorité	#	Nom d'hôte	Nom de service	Sortie	SHN impactés	[TT]	Statut
Wed 10:02:49 [6d 5h 14']	3	1	host2.example.com	Interface eth1	CRITICAL: Interface eth1: 45 45 45	Connexion		
Wed 10:04:40 [6d 5h 12']	3	1	host2.example.com	Interface eth2	CRITICAL: Interface eth2: 48 48 48	Connexion		
Wed 10:00:29 [6d 5h 16']	4	1	host2.example.com		CRITICAL: : 10 10 10			
Wed 10:28:35 [6d 4h 48']	4	1	messagerie	Load	CRITICAL: Load: 43 43 43			
Wed 10:13:32 [6d 5h 3']	4	1	brouteur	Load	CRITICAL: Load: 99 99 99			
Wed 10:08:10 [6d 5h 8']	4	1	proto4	RAM	CRITICAL: RAM: 60 60 60			
Wed 10:00:09 [6d 5h 16']	4	1	proto4	Load	CRITICAL: Load: 89 89 89			
Wed 10:12:31 [6d 5h 4']	4	1	host2.example.com	CPU	CRITICAL: CPU: 28 28 28			
Wed 10:02:59 [6d 5h 14']	4	1	messagerie	Interface eth0	WARNING: Interface eth0: 5 5 5			
Wed 10:26:04 [6d 4h 51']	4	1	host1.example.com	Processes	WARNING: Processes: 95 95 95			
Wed 10:22:54 [6d 4h 54']	4	1	host1.example.com	Load	WARNING: Load: 86 86 86			
Wed 10:07:30 [6d 5h 9']	4	1	brouteur	RAM	WARNING: RAM: 43 43 43			
Wed 09:58:48 [6d 5h 18']	4	1	host2.example.com	RAM	UNKNOWN: RAM: 63 63 63			
Wed 10:18:53 [6d 4h 58']	4	1	brouteur		UNKNOWN: : 88 88 88			
Wed 10:20:53 [6d 4h 56']	4	1	brouteur	Processes	UNKNOWN: Processes: 76 76 76			
Wed 09:59:58 [6d 5h 17']	3	1	host2.example.com	Interface eth0	OK: Interface eth0: 63 63 63			
Wed 10:05:10 [6d 5h 11']	4	1	proto4	Processes	OK: Processes: 82 82 82			
Wed 10:29:15 [6d 4h 47']	4	1	host2.example.com	Load	OK: Load: 69 69 69			
Wed 10:22:24 [6d 4h 54']	4	1	host2.example.com	Processes	OK: Processes: 21 21 21			
Wed 10:10:31 [6d 5h 6']	4	1	proto4		OK: : 14 14 14			

Fig. 5.5 – Thème Classique

Thème Malvoyant

Ce thème est, comme son nom l'indique, destiné aux personnes malvoyantes ou souffrant de troubles de perception des couleurs. L'interface est ainsi épurée et dépourvue de couleurs.

Thème Nocturne

Le thème *Nocturne* est conçu pour un affichage dans des conditions de faible luminosité afin notamment de permettre la supervision de nuit. Ses couleurs ambrées visent ainsi à reposer l'œil de l'opérateur, mais sont en revanche contre-indiquées en cas de fort éclairage.

Rafraîchissement automatique de la page

Il est parfois souhaitable de rafraîchir régulièrement la page afin de prendre en compte la détection de nouveaux évènements par Vigilo. Une case à cocher placée en haut à droite de l'interface permet d'activer ou de désactiver le rafraîchissement automatique.

Le texte accompagnant la case à cocher peut varier en fonction de la configuration retenue par votre administrateur. De même, l'état initial de cette case à cocher est configuré par l'administrateur. Ce paramétrage est mémorisé durant toute votre session de consultation de VigiBoard. Il n'est donc pas nécessaire de le reconfigurer à chaque changement de page.

Note : Le rafraîchissement automatique est temporairement mis en pause lorsqu'une boîte de dialogue est affichée à l'écran. Dans ce cas, la case à cocher reste cochée, mais son effet est inhibée tant que toutes les boîtes de dialogue

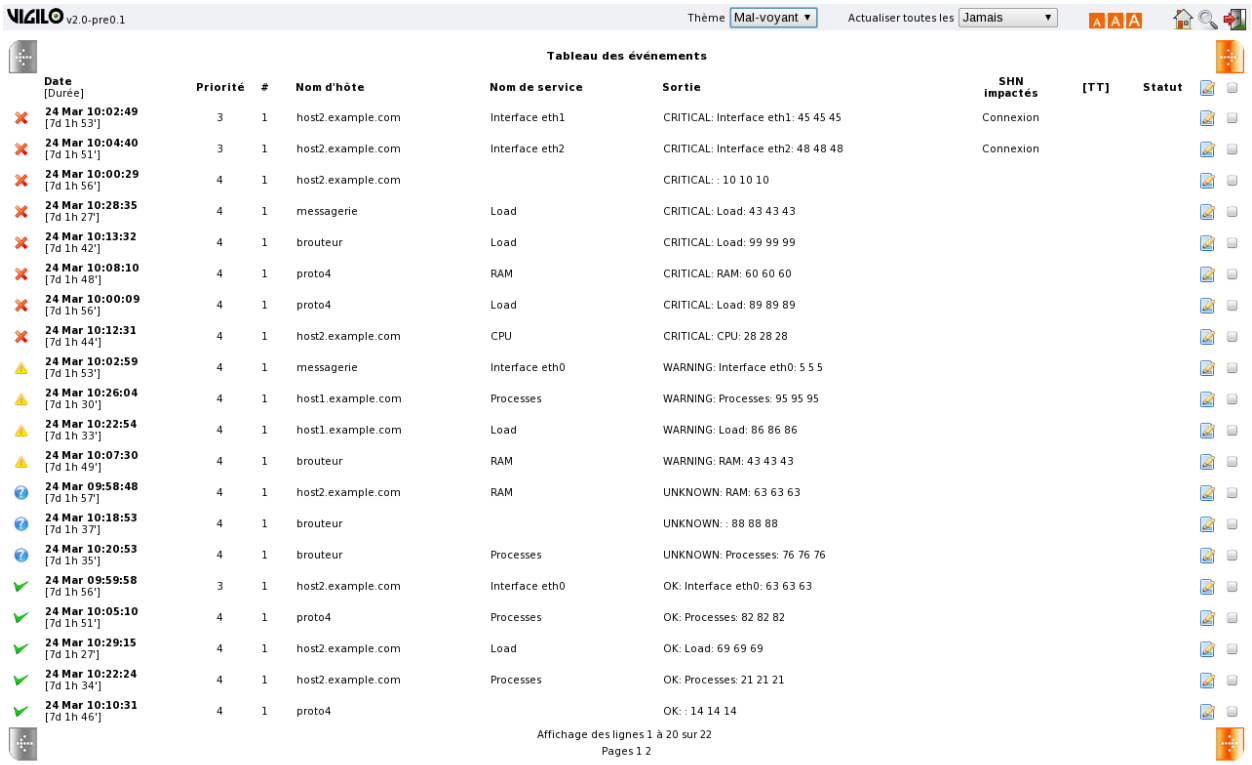
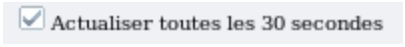


Fig. 5.6 – Thème Malvoyant

Fig. 5.7 – Thème Nocturne



affichées à l'écran n'ont pas été refermées.

Réglage de la taille de la police

La taille de la police est paramétrable en cliquant sur l'une des trois icônes représentées ci-dessous et situées elles aussi en haut à droite de l'interface :

```
..  figure:: img/fontsize_selection.png
```

Trois tailles de polices de caractères sont disponibles : petite, moyenne, grande.

Filtrage des événements

Il est possible de restreindre le nombre d'évènements affichés dans le tableau grâce à l'outil de filtrage de VigiBoard. Celui-ci est accessible en cliquant sur l'icône ci-dessous, placée en haut à droite de l'interface :

L'outil se présente sous la forme d'une boîte de dialogue permettant de filtrer les évènements par groupes d'hôtes/services, par noms d'hôte ou de service, par message d'erreur, par numéro de ticket d'incident, par date de début/fin, etc.

Note : À noter qu'un mécanisme d'auto-complétion permet de n'avoir à taper que les trois premières lettres de chaque champ avant de se voir proposer une liste de choix. L'outil autorise en outre l'utilisation des caractères jokers `?` et `*` pour représenter respectivement un caractère quelconque et une suite de caractères quelconques.

Édition d'un événement

L'ouverture de la boîte de dialogue d'édition d'un événement se fait en cliquant sur l'icône ci-dessous, placée à la fin de la ligne de l'évènement en question :

```
..  figure:: img/edit_icon.png
```

Il est alors possible de modifier l'identifiant du ticket d'incident associé à cet événement, et de changer l'état d'acquittement de l'évènement (« Non pris en compte », « Pris en compte », « Fermé »).

Un état supplémentaire est disponible pour les utilisateurs disposant en plus de la permission d'administration de VigiBoard. Il s'agit de l'état « Forcer à Fermé ». Lorsque cet état est choisi, l'incident est automatiquement clos (dans l'historique, le dernier état de l'hôte ou du service concerné est passé à « OK »). Cette option permet par exemple de traiter le cas où un équipement qui faisait l'objet d'une alerte a été retiré du parc supervisé. Étant donné que l'état de cet équipement ne peut plus être interrogé, il ne pourra jamais repasser à un état nominal et l'événement associé ne peut pas être supprimé par des méthodes conventionnelles.

Note : Notez qu'il est possible d'éditer plusieurs évènements simultanément en cochant les cases situées à la fin de leurs lignes respectives, puis en cliquant sur une icône semblable à la précédente, mais située cette fois à la droite de la ligne d'en-tête du tableau.

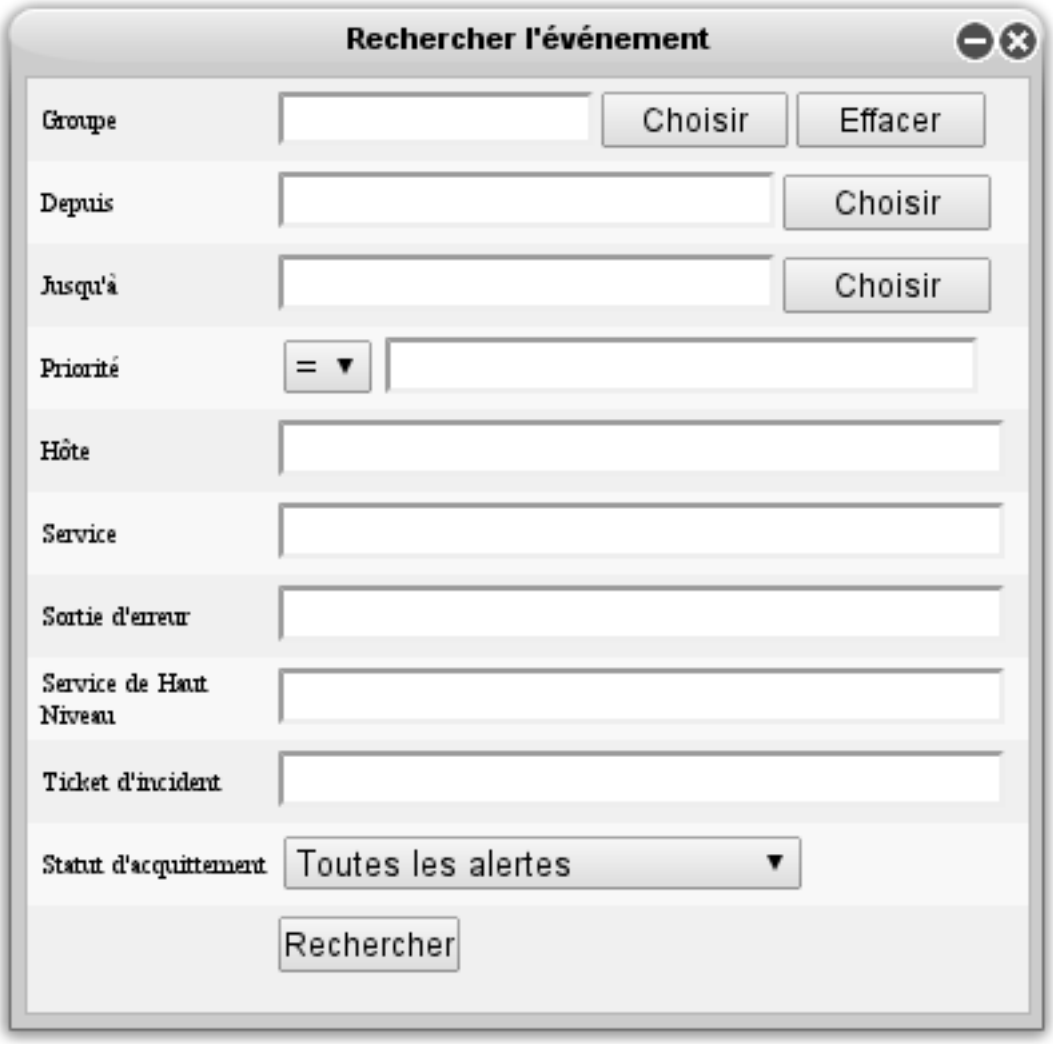


Fig. 5.8 – Outil de recherche de VigiBoard.

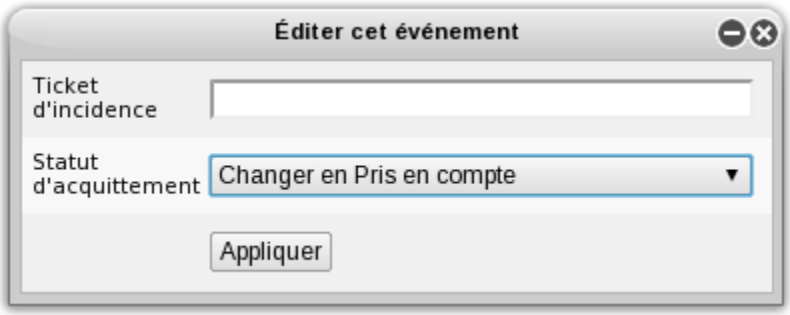


Fig. 5.9 – Dialogue d'édition d'un évènement.

Avertissement : Si vous sélectionnez plusieurs événements et que vous cliquez sur l'icône d'édition située à la fin de la ligne d'un événement, vous n'éditerez que les informations se rapportant à cet événement bien précis.

5.2.3 Fonctionnalités avancées

En cliquant sur l'icône située au début de la ligne d'un évènement (qui représente l'état de l'élément supervisé sur lequel cet incident est survenu), une boîte de dialogue détaillant les informations sur l'évènement en question s'ouvre.

Dans le premier tiers de cette boîte de dialogue sont visibles l'état initial et l'état courant de l'événement, ainsi que le pire état de l'élément supervisé sur lequel l'évènement est survenu.

Le second tiers donne ensuite accès à plusieurs fonctionnalités avancées :

- L'historique détaillé de l'évènement corrélé, c'est à dire en réalité l'historique de l'évènement brut (alerte) causant cet évènement corrélé (se rapporter au glossaire pour la distinction entre événements bruts et corrélés) ;
- La liste des évènements bruts masqués rattachés à cet événement corrélé ;
- L'historique de tous les évènements corrélés survenus sur cet élément supervisé.

Enfin, le dernier tiers est constitué d'une liste de liens vers des applications externes, telles que Nagios ou VigiGraph.

Historique d'un événement brut

Cette page détaille – comme son nom l'indique – l'historique d'un événement brut, et liste notamment les alertes Nagios successives qui s'y rapportent. Lorsque cet événement brut est la cause d'un événement corrélé, les modifications apportées par les opérateurs au statut de l'évènement corrélé en question sont également visibles.

Liste des évènements bruts masqués

Sur cette page sont affichés tous les évènements bruts rattachés à un événement corrélé, sans en être la cause directe. Ils sont généralement la conséquence d'un autre événement brut plus grave, détecté par le corrélateur comme la cause

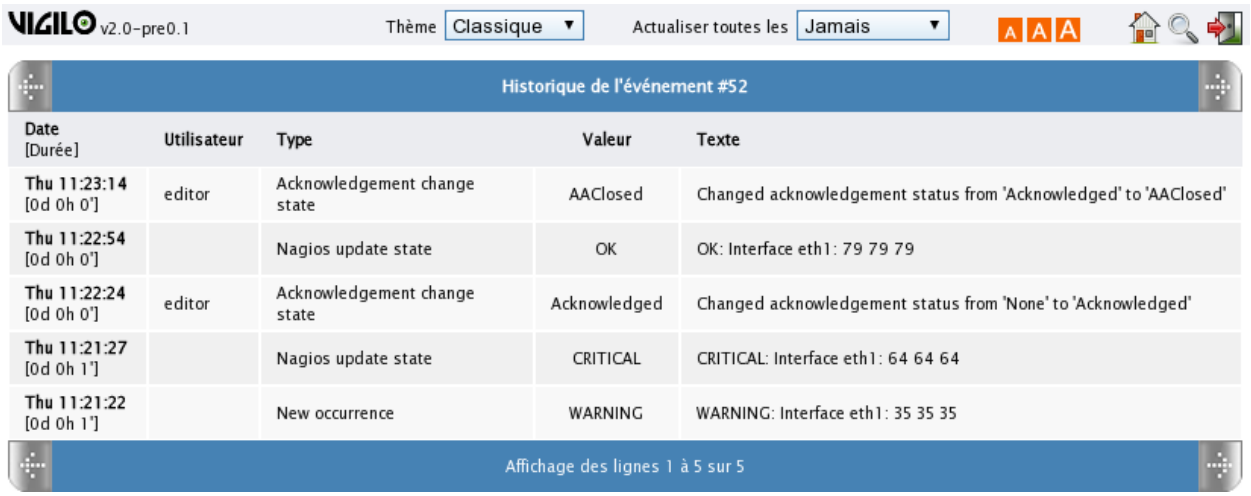


Fig. 5.10 – Historique d'un évènement brut.

principale de l'évènement corrélé.

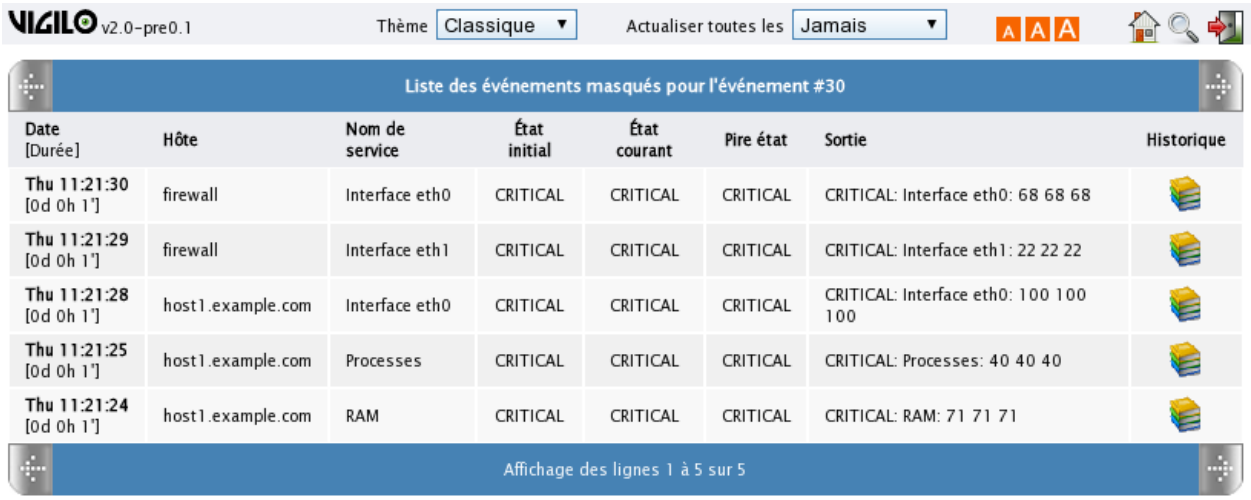


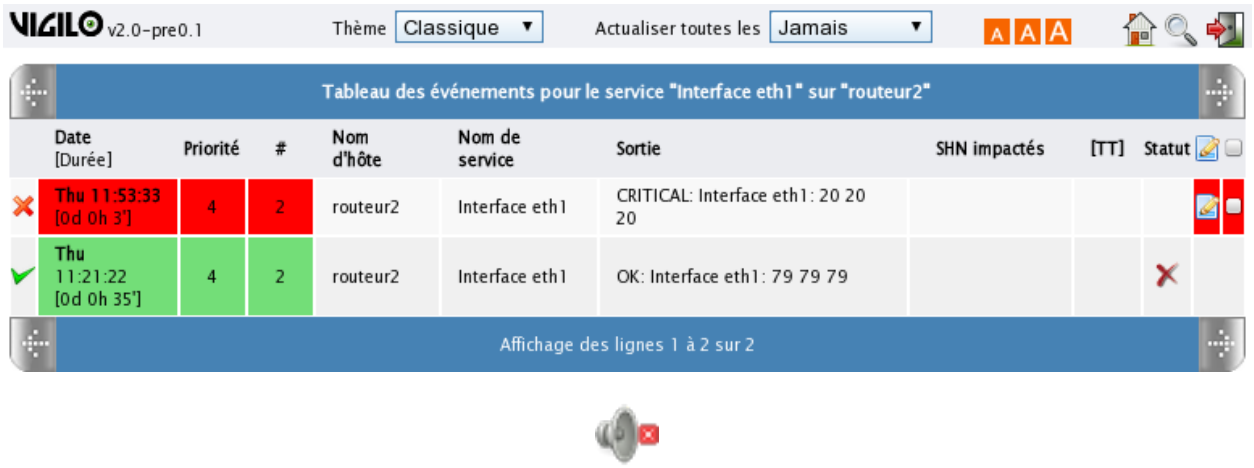
Fig. 5.11 – Liste des évènements bruts masqués.

Historique d'un hôte ou d'un service

Cette page permet de visualiser instantanément l'historique complet des évènements corrélés survenus sur un élément supervisé donné (hôte ou service), afin par exemple de détecter un problème récurrent. Ces évènements sont triés par ordre anti-chronologique.

Inhibition des alertes

Cette fonctionnalité permet d'empêcher l'apparition de nouvelles alertes dans VigiBoard pour un hôte ou un service donné. La page listant les règles d'inhibition (aussi appelées règles de mise en silence) actives est accessible en cliquant sur l'icône ci-dessous située en haut à droite de l'interface :



Les règles sont triées par défaut par ordre décroissant de dernière mise à jour, comme on peut le voir sur la capture ci-dessous :

Depuis cette page, il est alors possible d'éditer ou de supprimer une règle existante en cliquant sur l'une des deux icônes située en début de ligne :

L'utilisateur est également en mesure d'ajouter une nouvelle règle en cliquant sur le bouton prévu à cet effet situé juste au-dessus du tableau :

Note : Les alertes déjà traitées (c'est à dire soit déjà ignorées, soit déjà affichées dans VigiBoard) ne sont pas impactées par les modifications faites sur les règles (ie. aucun traitement n'est réalisé a posteriori pour générer des alertes ou pour détruire des alertes déjà présentes). De même, ce filtrage n'impacte que la production d'alertes, il ne concerne pas l'état des hôtes/services (donc l'élément peut être `CRITICAL` dans VigiMap mais ne pas avoir d'alarme correspondante dans VigiBoard).

5.2.4 Annexes

Glossaire - Terminologie

Ce chapitre recense les différents termes techniques employés dans ce document et donne une brève définition de chacun de ces termes.

API Interface logicielle de programmation, permettant à un développeur d'enrichir la liste des fonctionnalités proposées par un logiciel.

CGI Interface standard de communication entre un serveur web et un programme capable de générer une réponse HTTP valide. Il s'agit par exemple de l'interface retenue par Nagios (< 3.3) pour la génération de ses pages web.

CSS Feuilles de styles permettent de modifier la représentation graphique des éléments d'une page web. La version généralement supportée par les navigateurs est la version 2, définie par le document disponible sur http://www.w3.org/TR/CSS2/.

CSV À l'origine, désigne un format textuel de transfert de données dans lequel les entrées sont séparées par des retours chariot et les champs par des virgules (comma). De nos jours, désigne plus généralement un format

tabulé pour l'échange de données en vue de leur traitement dans un logiciel de type tableur ou par un traitement automatisé (scripts).

DN Identifiant unique dans le cadre d'un annuaire LDAP.

Événement brut Alerte envoyée par Nagios au corrélateur de Vigilo pour analyse.

Événement corrélé Incident détecté par Vigilo suite à la corrélation des alertes Nagios (évènements bruts). Un événement corrélé est causé par un unique événement brut (exemple : la panne d'un routeur), mais de nombreux autres évènements bruts peuvent lui être rattachés (exemple : les alertes indiquant que les serveurs situés derrière le routeur en panne sont indisponibles). Ces événements secondaires rattachés à l'événement corrélé sont alors appelés « événements bruts masqués ».

KDC Serveur permettant un transfert sécurisé des clés de chiffrement utilisées pour les communications entre divers services. Ce serveur est notamment utilisé lors des échanges initiaux du protocole Kerberos.

LDAP Protocole pour l'interrogation d'un annuaire, servant généralement à recenser les utilisateurs autorisés d'un système et les différentes propriétés associées à ces utilisateurs.

OS Système d'exploitation.

Nagios Composant libre de supervision système et réseau.

RRD Base de données de taille fixe utilisant des fichiers circulaires, dont les données sont progressivement compressées (avec perte) au fur et à mesure de leur vieillissement.

RRDtool Composant libre de gestion de bases RRD (stockage, restitution, génération de graphiques).

SGBD[R] Logiciel permettant d'héberger des bases de données [relationnelles] sur une machine.

SQL Langage de requêtes structuré pour l'interrogation d'une base de données relationnelle.

URL Chaîne de caractères permettant d'identifier une ressource, par exemple une page web sur Internet. Exemple : http://www.vigilo-nms.com/

WSGI Une interface pour la communication entre une application et un serveur web, similaire à CGI. Il s'agit de l'interface utilisée par Vigilo.

5.3 Guide d'utilisation

5.3.1 Introduction

Présentation

En plus de la gestion des alertes, Vigilo sait gérer la remontée et le stockage des informations de performance du parc supervisé. La métrologie est ainsi collectée dans une base de données RRD et **VigiGraph** permet ensuite d'accéder aux graphes construits à partir de ces données. Il est possible de rechercher dans l'ensemble des données collectées, d'afficher les graphes correspondants, de les comparer entre eux, de zoomer, etc.

Les fonctions avancées offertes par VigiGraph permettent d'intégrer les actions préventives qui découlent des observations effectuées quotidiennement par les analystes du système, comme par exemple des mises à niveau de la capacité mémoire ou de serveurs faisant suite à des baisses de performances ou des saturations continues constatées sur certains équipements

Principales fonctionnalités

— Menus de recherche sur tous les groupes, hôtes et indicateurs ;

— Affichage de plusieurs graphes en simultané pour plusieurs hôtes ;
— Possibilité pour chaque utilisateur de sauvegarder ses graphes ;
— Menu de navigation au sein de chaque graphe : période, avant, arrière, zoom, etc.
— Possibilité d'export CSV et d'impression.

5.3.2 Démarrage rapide

VigiGraph fournit une interface graphique accessible depuis un navigateur Web. La configuration par défaut est suffisante pour un démarrage rapide.

Accès à l'interface

L'utilisation de VigiGraph se fait simplement en accédant, via votre navigateur, à l'adresse indiquée par votre administrateur. Par exemple : http://supervision.example.com/vigigraph/.

Authentification

Note : Dans le cas où un mécanisme d'authentification externe a été défini par votre administrateur, il se peut qu'aucune authentification ne vous soit demandée, même lorsqu'il s'agit de votre première connexion. Le reste de ce chapitre décrit le cas où une authentification interne a lieu et ne s'applique donc pas au cas de figure cité ci-dessus. Contactez votre administrateur pour plus d'information sur la configuration utilisée.

Si vous ne vous êtes jamais connecté sur VigiGraph ou si vous n'êtes plus authentifié, le formulaire d'authentification de la figure suivante s'affiche :

Fig. 5.12 – Écran d'authentification.

Selon le compte utilisateur auquel vous vous connecterez, vous disposerez d'un accès à plus ou moins d'hôtes et de services (et donc d'informations). Les données d'authentification demandées ici vous ont normalement été transmises par votre administrateur.

— Saisir les données d'authentification en renseignant les zones de saisie « Identifiant » et « Mot de passe ».
— Valider la saisie en cliquant sur le bouton « Connexion » (entouré en rouge sur la figure suivante).

Fig. 5.13 – Bouton de validation des données d'authentification.

En cas de succès, la page d'accueil s'affiche. Sinon, le formulaire d'authentification s'affiche à nouveau, avec un message précisant la nature de l'erreur :

Fig. 5.14 – Formulaire après un échec de l'authentification.

5.3.3 Fonctionnalités

Page d'accueil

Une fois la page d'authentification passée, la page d'accueil s'affiche, comme comme sur la figure qui suit. Il s'agit de l'interface principale de VigiGraph.

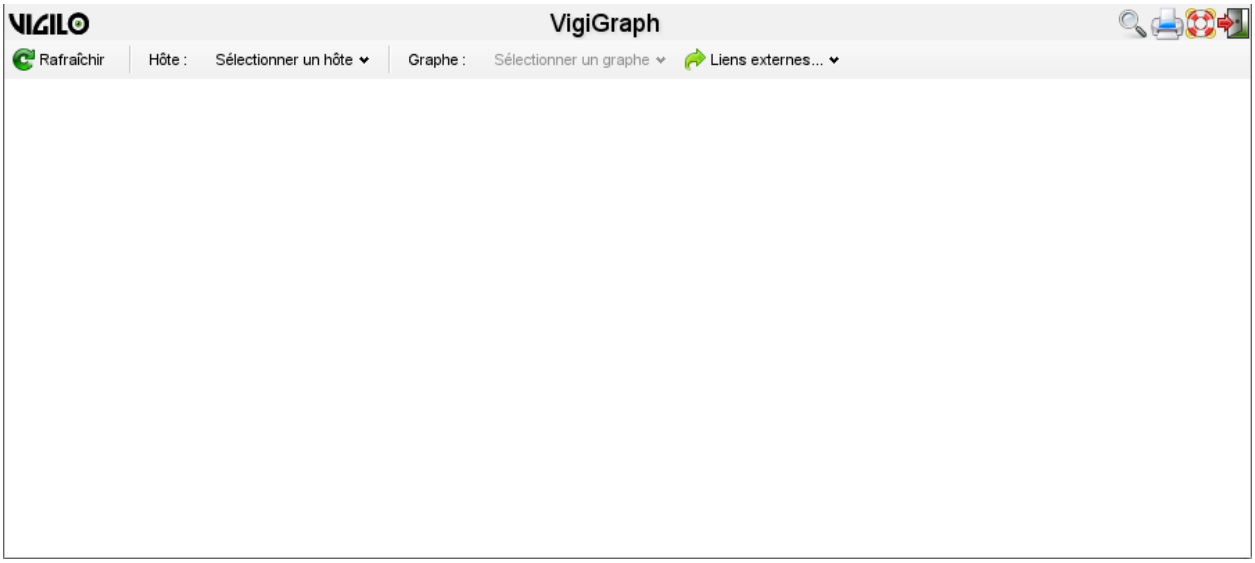


Fig. 5.15 – Page d'accueil.

Cette page permet à la fois la recherche, la visualisation, et l'impression des graphes.

Sélection d'un hôte

L'accès à la plupart des fonctionnalités de VigiGraph nécessite que le champ « Hôte » de la barre de sélection soit renseigné.

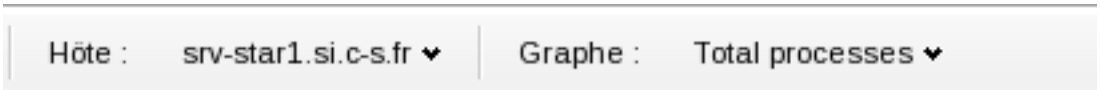


Fig. 5.16 – Barre de sélection.

Après un premier clic sur le bouton « Sélectionner un hôte », une boîte de dialogue permet de choisir l'hôte souhaité parmi l'arborescence des groupes configurés dans Vigilo.

Sélection d'un graphe à afficher

Une fois un hôte sélectionné, le bouton « Sélectionner un graphe » devient actif.

Un clic sur ce bouton permet de choisir un graphe à afficher parmi la liste de graphes diponibles pour cet hôte. La fenêtre de visualisation s'ouvre alors, comme le montre l'illustration suivante :

Cette fenêtre est décrite plus en détail dans la section *Visualisation d'un graphe*.

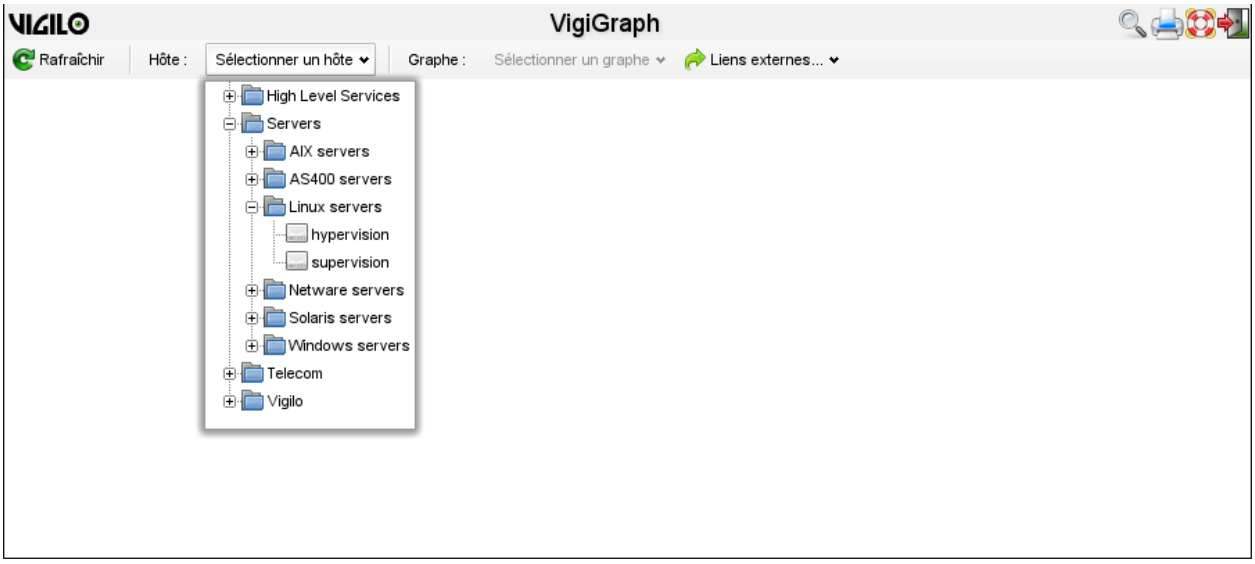


Fig. 5.17 – Sélection de l'hôte.

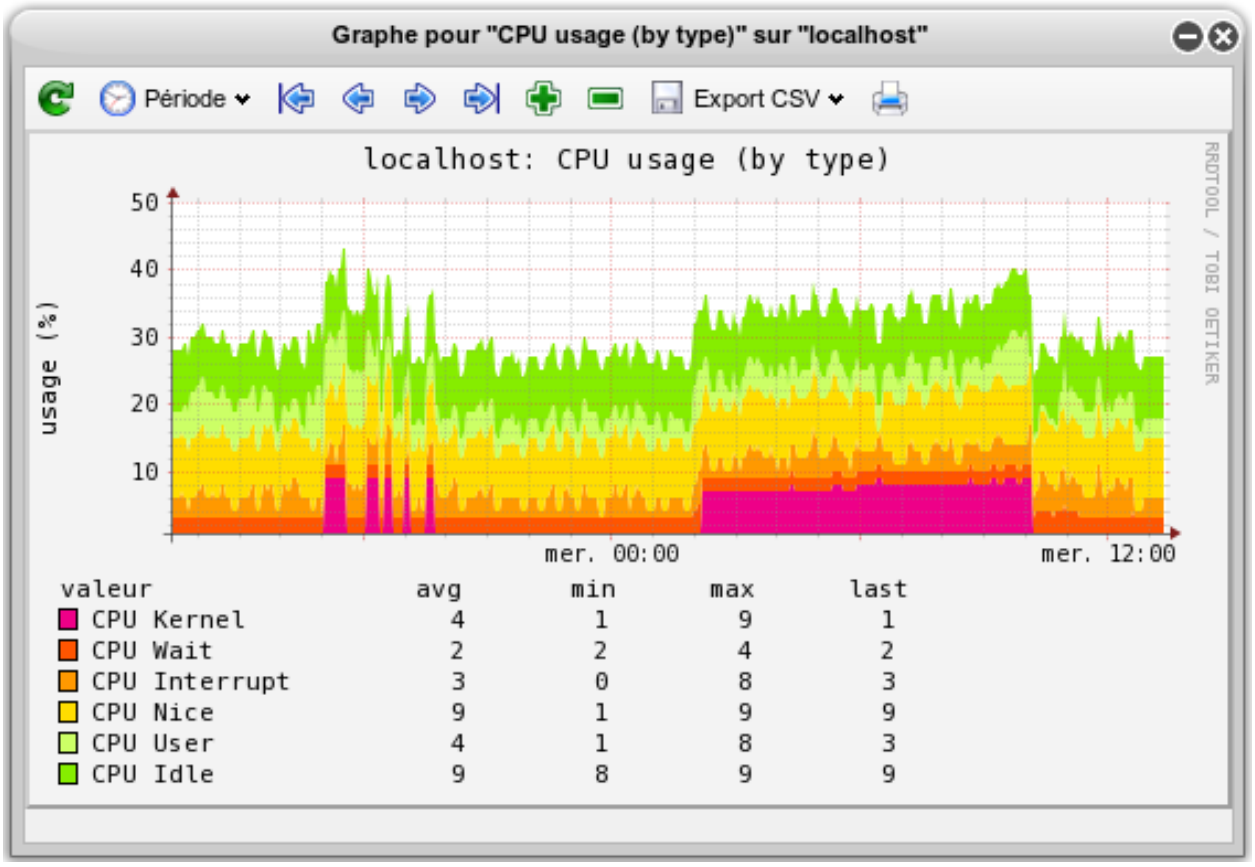


Fig. 5.18 – Affichage du graphe après sélection de l'hôte et du graphe.

Note : Il est tout à fait possible de sélectionner plusieurs graphes de plusieurs hôtes pour un affichage simultané.

Recherche

Cette fenêtre permet de rechercher parmi la liste des couples Hôtes/Graphes disponibles. L'activation s'effectue par un clic sur l'icône en forme de loupe  située en haut à droite de l'interface.

La fenêtre de recherche se présente au départ comme sur l'illustration suivante :

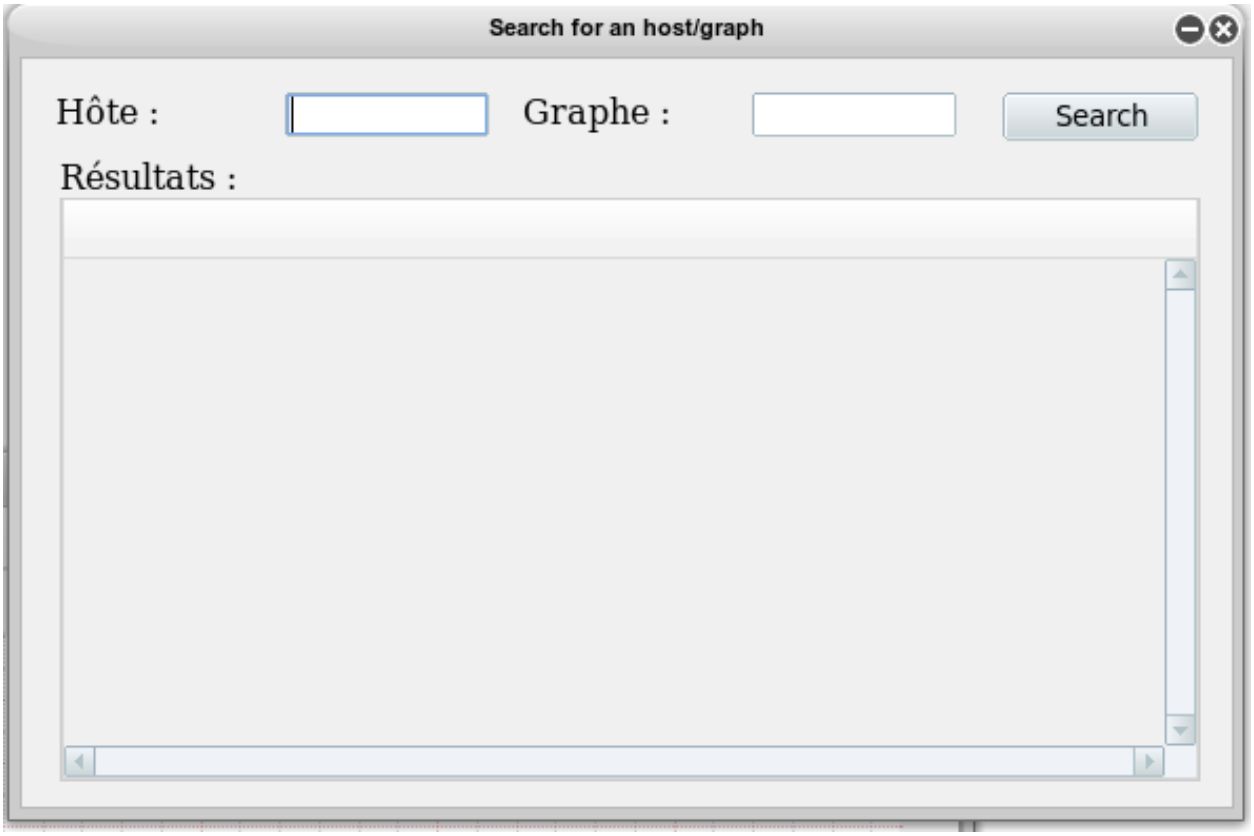


Fig. 5.19 – État initial du formulaire de recherche.

Les caractères « ? » et « * » permettent de représenter un caractère quelconque ou une suite de caractères quelconques, respectivement.

Un clic sur le bouton « Rechercher » provoque l'affichage de la liste complète des couples Hôtes/Graphes déclarés en base de données et correspondant aux motifs de recherche saisis.

Si aucun texte n'a été saisi dans le champ « Hôte » et le champ « Graphe », la liste complète de tous les couples Hôtes/Graphes disponibles est affichée.

Les champs de saisie « Hôte » et « Graphe » permettent la saisie de critères de recherche sur un serveur ou un graphe particulier.

et/ou :

Les deux critères peuvent être combinés afin de n'afficher que certains graphes d'un ensemble d'hôtes.

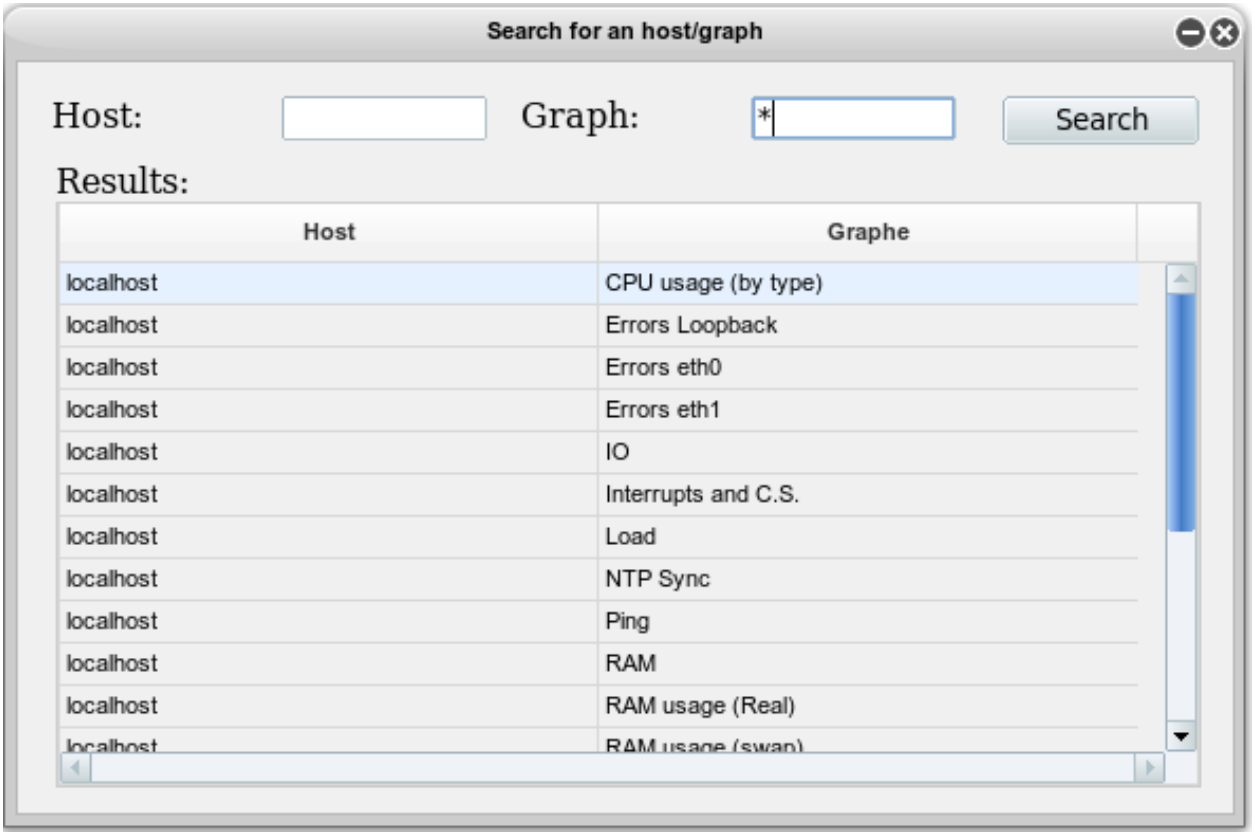


Fig. 5.20 – Liste de tous les graphes disponibles.

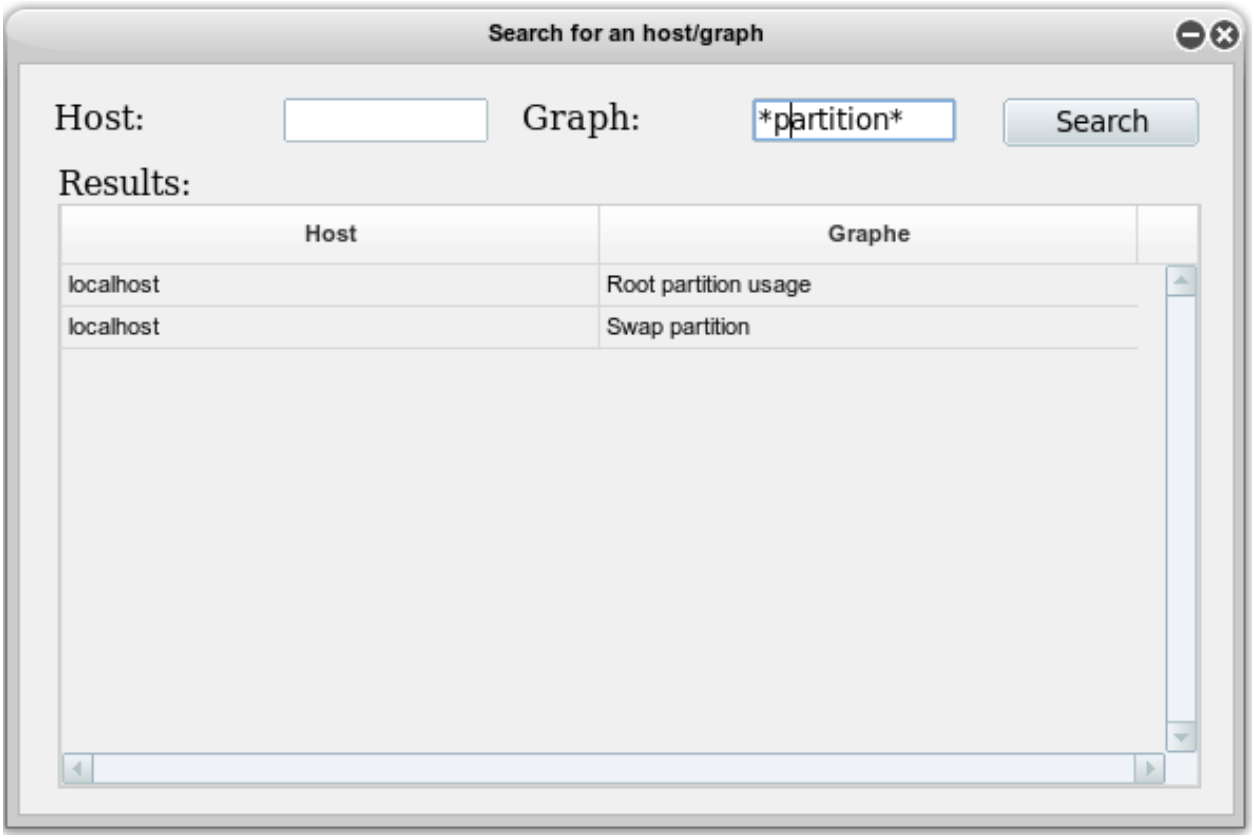


Fig. 5.21 – Recherche des graphes portant sur une partition.

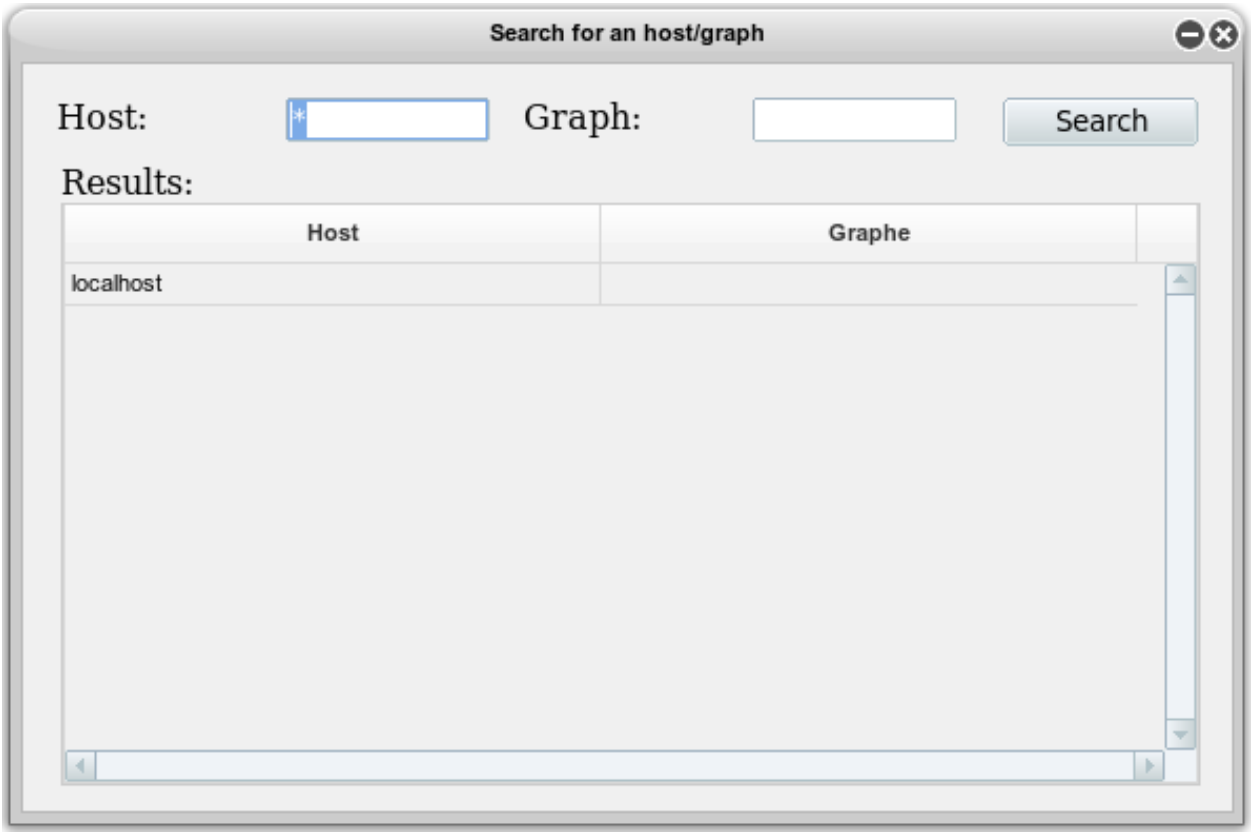


Fig. 5.22 – Recherche parmi les hôtes.

La sélection d'un élément de cette liste est possible par un clic sur la ligne correspondante. Ce clic provoque l'actualisation de la fenêtre « Sélection d'un hôte » afin de sélectionner l'hôte (et éventuellement le graphe) cliqué. Si l'élément cliqué correspond à un graphe, le graphe est affiché à l'écran, comme sur l'illustration suivante :

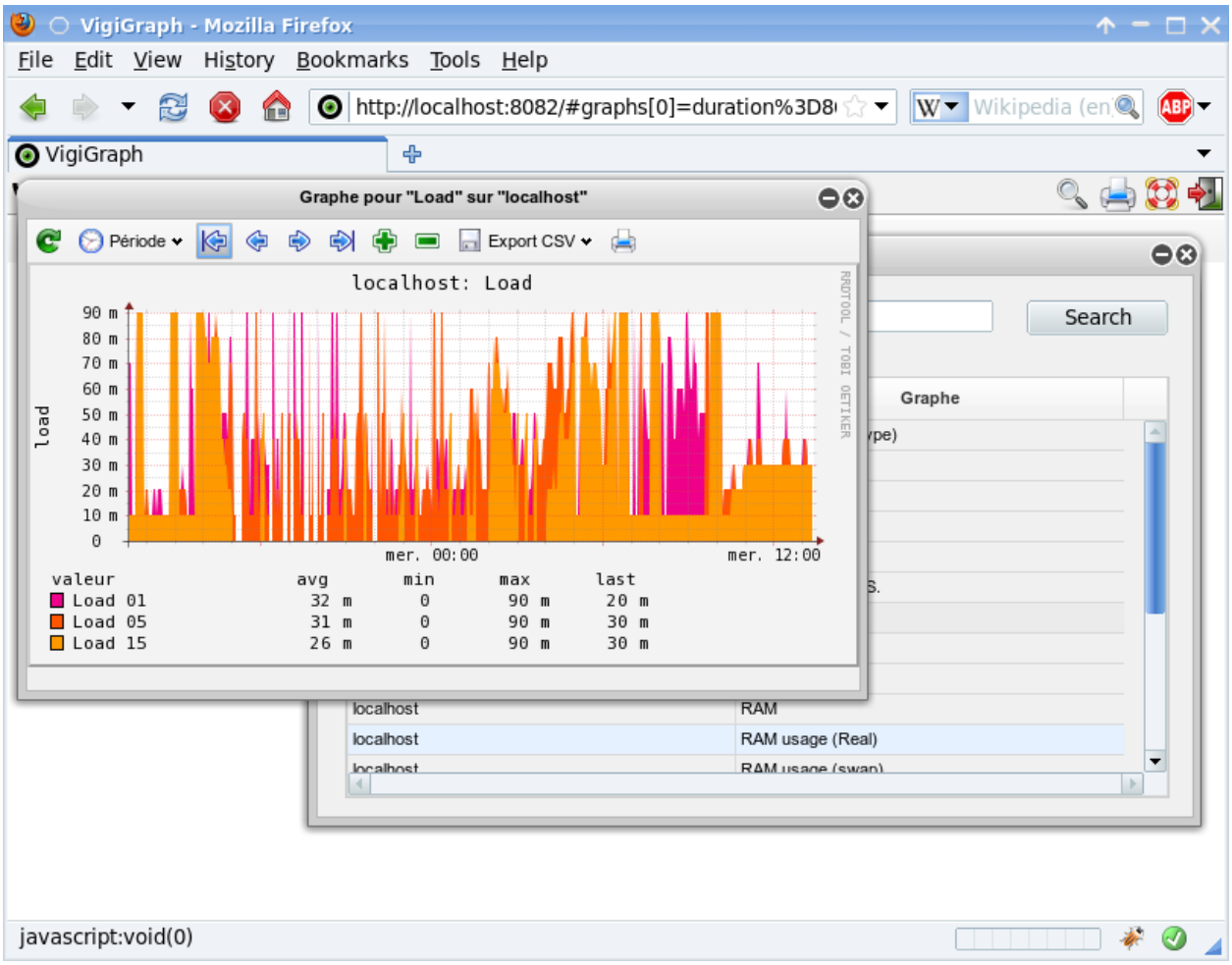


Fig. 5.23 – Affichage d'un graphe suite à un clic dans la fenêtre de recherche.

Visualisation d'un graphe

L'affichage des graphes s'effectue dans une fenêtre distincte, semblable à celle présentée dans l'illustration suivante :

Cette fenêtre comporte :

- une barre de titre rappelant le nom de l'hôte et du graphe affiché ;
- des boutons de gestion de l'affichage du graphe ;
- la zone d'affichage à proprement parler.

Le graphe donne l'évolution des valeurs dans le temps, avec un intervalle de temps par défaut qui couvre les dernières 24 heures.

L'utilisateur peut modifier cette durée en cliquant sur les boutons et .

Ceci provoque le rafraîchissement des données sur une durée respectivement diminuée ou agrandie avec un facteur 2.

La sélection de la plage de temps est aussi possible :

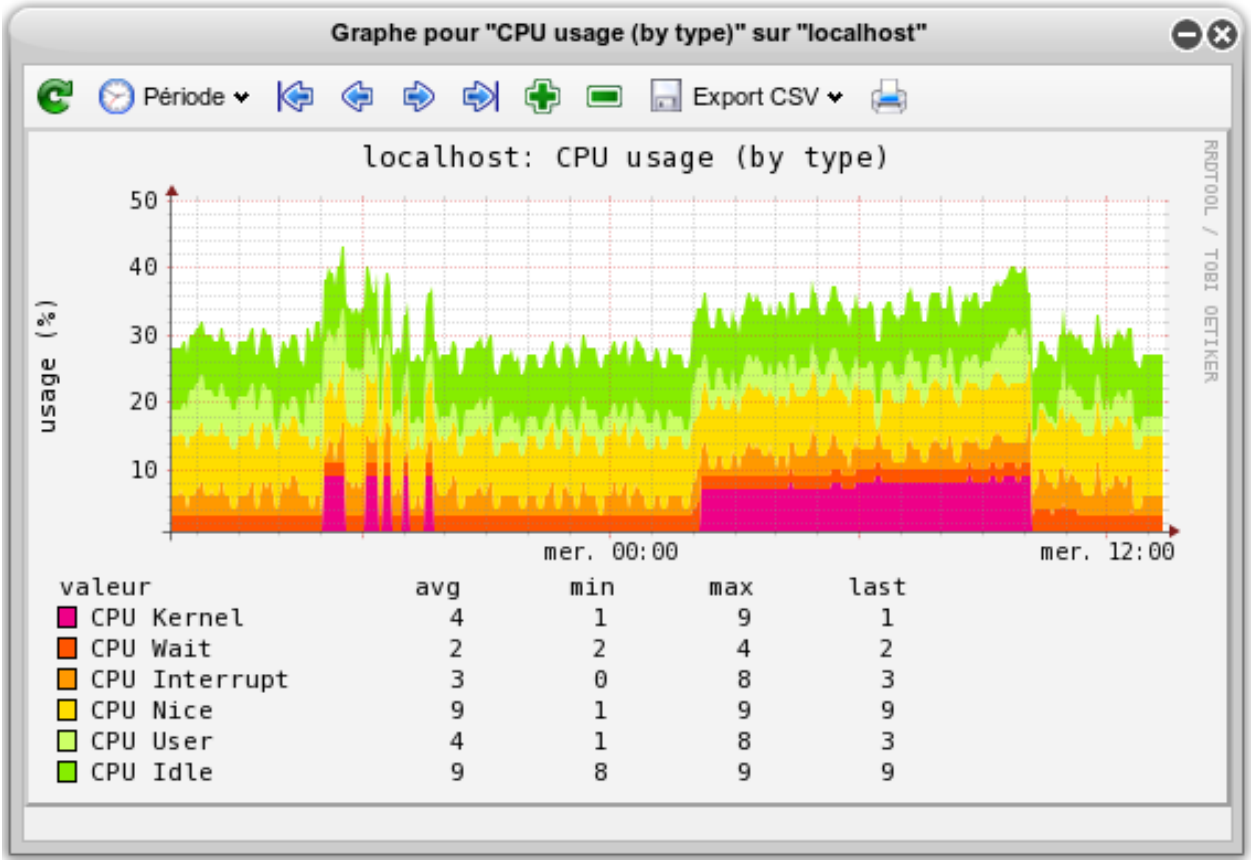


Fig. 5.24 – Fenêtre de visualisation d'un graphe.

— par un clic sur un des boutons  ,  ,  , et  (respectivement plage de début, plage précédente, plage suivante et plage de fin) ;

— ou par un choix dans la liste obtenue après un clic sur le bouton « Période »  .
Ceci provoque le rafraîchissement des données sur la durée courante.

Dans l'illustration, le bouton  a été utilisé afin de se replacer au tout début de la période d'enregistrement. Partant de cette situation, l'utilisation du bouton  donne par exemple le graphe suivant :

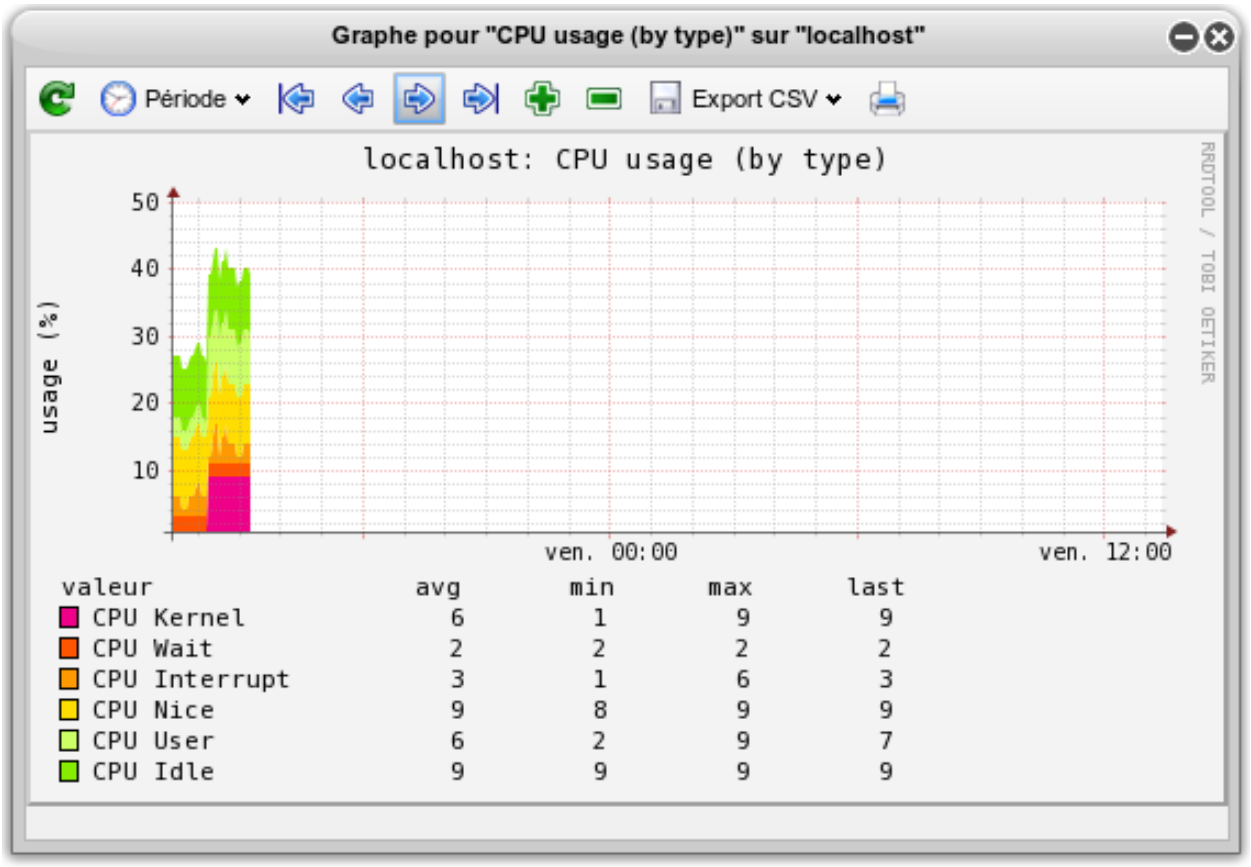


Fig. 5.25 – Progression dans le temps.

Le tableau suivant liste l'ensemble des boutons disponibles dans la fenêtre de visualisation, et détaille leur fonctionnement :

Tableau 5.3 – Liste des boutons

Bouton	Intitulé de la bulle d'aide	Détail
	Recharger le graphe	Rafraîchissement automatique du graphe selon la fréquence configurée : — bouton enfoncé → rafraîchissement automatique — bouton relâché → pas de rafraîchissement
	Menu de choix de la période	Rafraîchissement du graphe avec adaptation à la plage de temps
	Début du graphe	Rafraîchissement du graphe avec les valeurs de la première plage de temps
	Section précédente	Rafraîchissement du graphe avec les valeurs de la plage de temps précédente
	Section suivante	Rafraîchissement du graphe avec les valeurs de la plage de temps suivante
	Fin du graphe	Rafraîchissement du graphe avec les valeurs de la dernière plage de temps
	Zoomer	Rafraîchissement du graphe avec agrandissement de la plage de temps (x 2)
	Dézoomer	Rafraîchissement du graphe avec diminution de la plage de temps (/ 2)
	Export CSV	Export du graphe
	Imprimer le graphe	Impression du graphe

Consultation d'une page de supervision

Cette fonctionnalité permet de consulter la page Nagios de l'hôte actuellement sélectionné (voir la section *Sélection d'un hôte*).

L'activation s'effectue par un clic sur le bouton « Liens externes » dans la barre de sélection de l'hôte, suivi par un clic sur le bouton « Page Nagios ».

L'affichage s'effectue dans une page distincte, comme le montre l'illustration suivante :

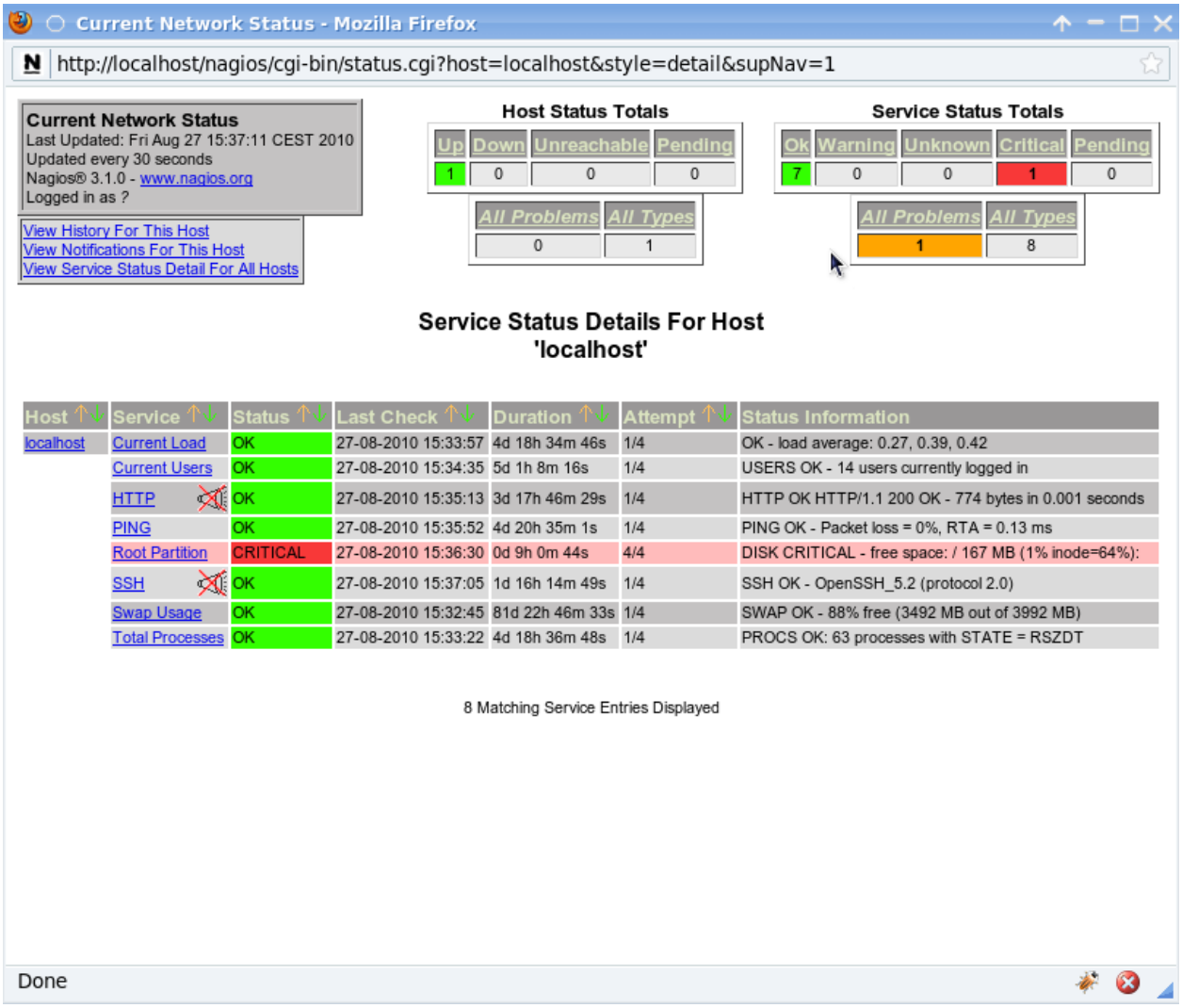


Fig. 5.26 – Exemple d'une page de supervision Nagios.

Consultation de l'ensemble des graphes de métrologie d'un hôte

Cette fonctionnalité permet de consulter l'ensemble des graphes de métrologie de l'hôte actuellement sélectionné (voir la section *Sélection d'un hôte*).

L'activation s'effectue par un clic sur le bouton « Liens externes » dans la barre de sélection de l'hôte, suivi par un clic sur le bouton « Page de métrologie ».

L'affichage s'effectue dans une page distincte, comme le montre l'illustration suivante :

Fig. 5.27 – Vue synthétique de tous les graphes d'un hôte.

Impression d'un graphe

Cette fonctionnalité permet d'imprimer le graphe courant, tel qu'il est actuellemnet présenté à l'écran.

L'activation de l'impression s'effectue par un clic sur le bouton de la fenêtre de visualisation du graphe.

Une fenêtre d'impression s'affiche alors.

Après renseignement des paramètres d'impression et validation, cette fenêtre se referme et l'impression commence.

Note : Si plusieurs graphes sont affichés à l'écran au moment de l'activation de l'impression, ils seront tous imprimés.

Export d'un graphe

Cette fonctionnalité permet d'exporter le graphe courant.

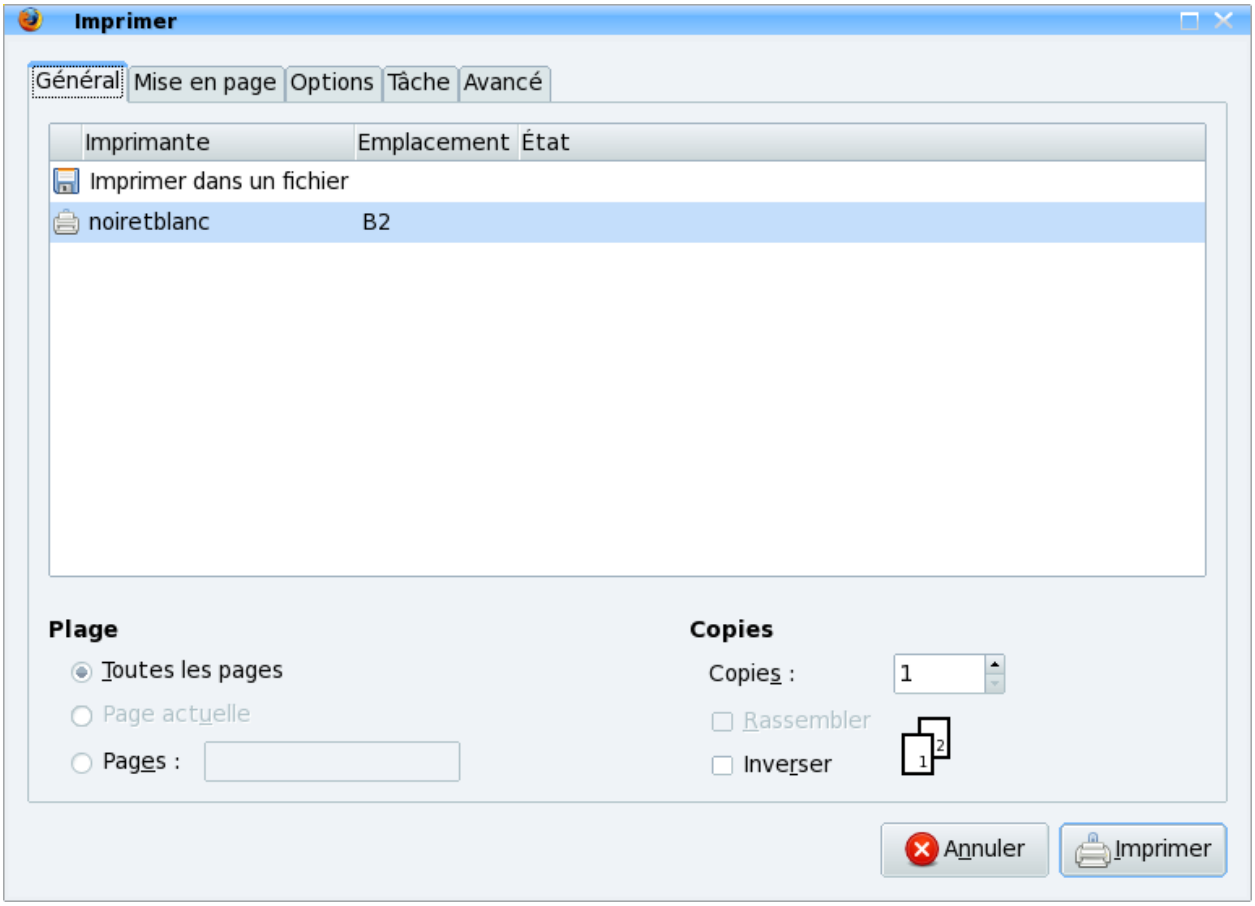


Fig. 5.28 – Fenêtre d'impression d'un graphe.

L'activation de l'export s'effectue par un clic sur le bouton  de la fenêtre de visualisation du graphe, suivi par un clic sur nom de l'indicateur à exporter (ou sur le label « Tous » pour exporter simultanément tous les indicateurs du graphe).

La fenêtre d'export s'affiche alors, comme sur l'illustration suivante :

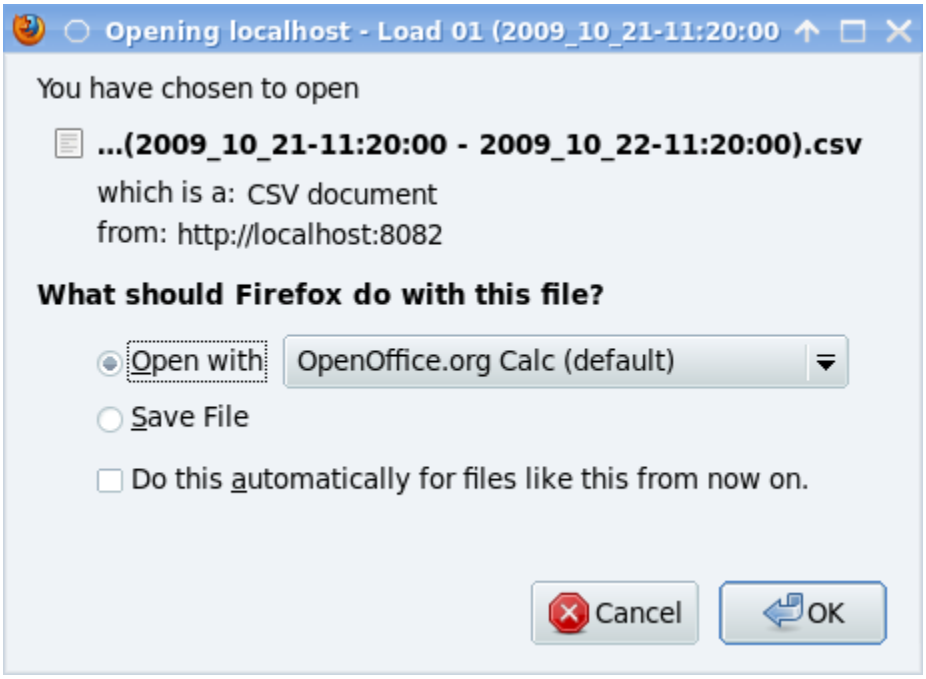


Fig. 5.29 – Fenêtre de sauvegarde des données exportées.

Une fois l'une des options validée, la fenêtre se referme. Le fichier généré est consultable à l'aide d'un tableur. Le contenu se présente comme suit :

La sortie par défaut d'un fichier au format CSV se caractérise comme suit :

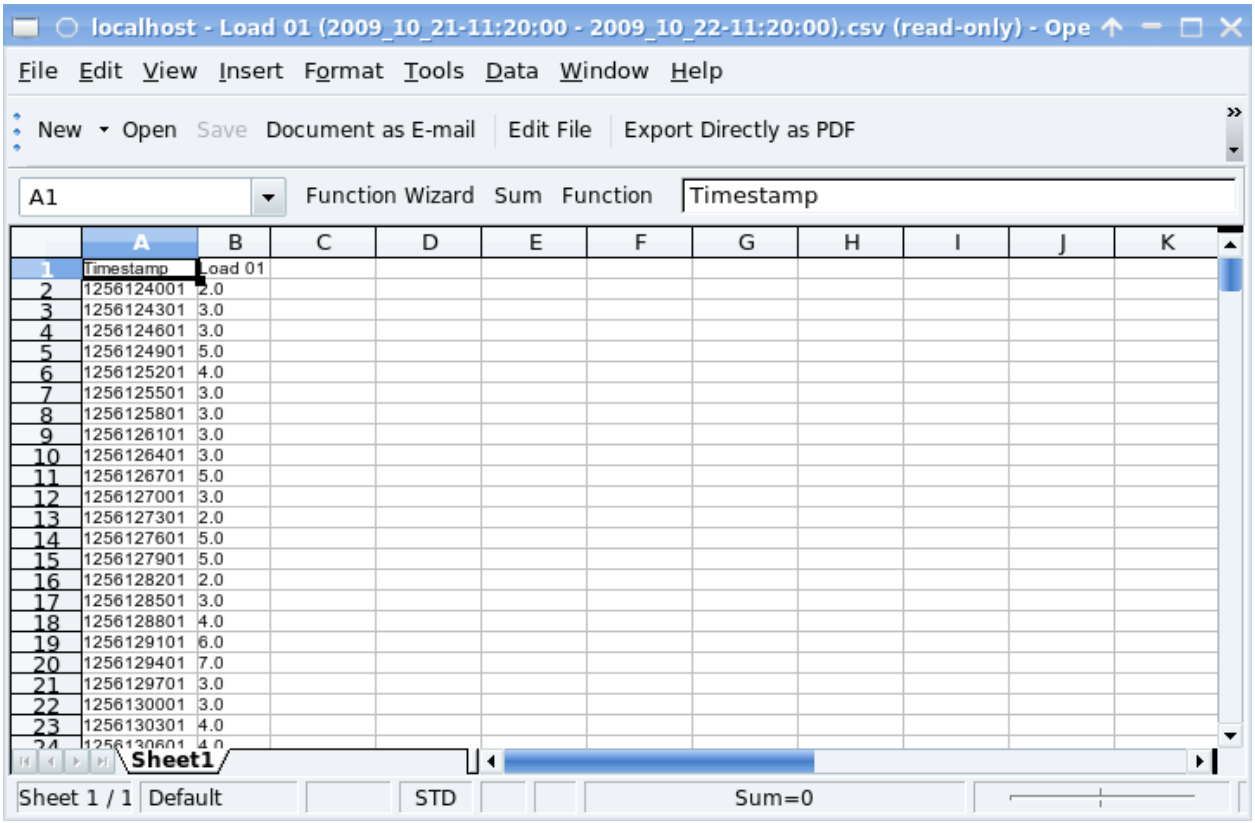
- Les valeurs sont entourées par des guillemets droits (") et séparées par des points virgules (;).
- Le contre-oblique () est utilisé en tant que caractère d'échappement.
- La première ligne du fichier contient un en-tête composé des libellés des champs dans leur ordre d'apparition (« Timestamp » puis l'indicateur ou la série d'indicateurs exportés).
- Les lignes suivantes correspondent aux valeurs de chaque indicateur à l'instant identifié par la valeur de la colonne « Timestamp ».
- Les lignes sont triées par ordre chronologique croissant.

L'administrateur peut choisir de paramétrer différemment l'export (notamment les caractères utilisés pour la séparation des valeurs, leur délimitation ou le caractère d'échappement). Consultez votre administrateur afin de connaître le format exact des fichiers générés.

Rafraîchissement automatique d'un graphe

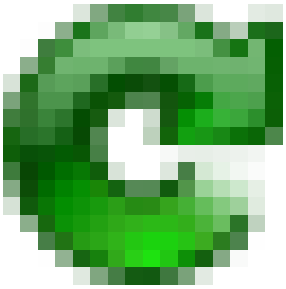
Cette fonctionnalité permet de réactualiser un graphe périodiquement.

La période figure dans un fichier de configuration ; elle est exprimée en secondes. Par défaut, elle est initialisée à 30 secondes. Cette donnée n'est pas modifiable via l'application. Elle est gérée par l'administrateur système.



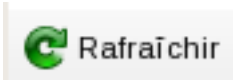
	A	B
1	Timestamp	Load 01
2	1256124001	2.0
3	1256124301	3.0
4	1256124601	3.0
5	1256124901	5.0
6	1256125201	4.0
7	1256125501	3.0
8	1256125801	3.0
9	1256126101	3.0
10	1256126401	3.0
11	1256126701	5.0
12	1256127001	3.0
13	1256127301	2.0
14	1256127601	5.0
15	1256127901	5.0
16	1256128201	2.0
17	1256128501	3.0
18	1256128801	4.0
19	1256129101	6.0
20	1256129401	7.0
21	1256129701	3.0
22	1256130001	3.0
23	1256130301	4.0

Fig. 5.30 – Données exportées vues dans un tableur.

L'activation s'effectue par un clic sur le bouton  de la fenêtre de visualisation du graphe. Le bouton apparaît alors enfoncé.

La désactivation s'effectue aussi par un clic sur le même bouton. Celui-ci apparaît alors relâché. L'illustration suivante présente l'état du bouton selon que le rafraîchissement automatique est activé ou non :

Illustration : Rafraîchissement automatique activé (à gauche) et désactivé (à droite)

Un bouton global 

permet également d'activer ou de désactiver le rafraîchissement automatique sur l'ensemble des graphes visibles à l'écran.

Le comportement de ce bouton est similaire à celui du bouton positionné sur chaque fenêtre de graphe : un clic permet d'activer le rafraîchissement (le bouton apparaît alors enfoncé), un nouveau clic désactive le rafraîchissement (le bouton apparaît alors relâché).

Sauvegarde de l'affichage

Cette fonctionnalité permet de sauvegarder un ensemble de graphes affichés à l'écran pour des opérations ultérieures.

La sauvegarde s'effectue via le navigateur, en ajoutant un marque-page. Par exemple, sous Firefox :

- Cliquer sur le menu « Marque-Pages ».
- Choisir l'option « Marquer cette page ».
- Saisir un nom pour le marque-page (ou laisser la valeur par défaut).
- Choisir un dossier de destination (par exemple « Barre personnelle »).
- Valider par un clic sur le bouton « Terminer ».

Le rappel de la sélection s'effectue également via le navigateur ; par exemple, sous Firefox :

- Cliquer sur le menu « Marque-Pages ».
- Sélectionner l'élément créé précédemment lors de l'ajout du marque-page .

Moteur de recherche

En plus de la fonctionnalité de recherche accessible directement depuis la page d'accueil (voir la section *Recherche*), VigiGraph est livré avec un moteur de recherche qui peut être utilisé directement depuis la barre de recherche de votre navigateur.

La procédure qui suit décrit l'ajout et l'utilisation de ce moteur de recherche depuis Firefox ; dans le cas où vous souhaiteriez utiliser un autre navigateur pour accéder à VigiGraph, reportez-vous au manuel dudit navigateur pour savoir comment adapter cette procédure.

Pour enregistrer le moteur de recherche sous Firefox (cette manipulation n'est à effectuer qu'une seule fois) :

- Cliquer sur le bouton à gauche de la zone des moteurs de recherche. → La liste des moteurs actuellement enregistrés s'affiche, ainsi que des actions possibles.
- Cliquer sur la ligne « Ajouter "Recherche Vigilo" ». → Le moteur de recherche de VigiGraph est ajouté à la liste des moteurs de recherche disponibles et la liste disparaît de l'écran.

Une fois le moteur de recherche enregistré, il apparaît dans le navigateur sous la forme d'une zone de saisie.

Pour effectuer une recherche :

— Cliquer sur le bouton à gauche et choisir « Recherche Vigilo » (représenté par l'icône de Vigilo : un œil vert) dans la liste des moteurs de recherche.
— Saisir un intitulé dans la zone de saisie.
— Cliquer sur le bouton en forme de loupe à droite ou valider à l'aide de la touche « Entrée ».

→ **Une page présentant des informations sur les serveurs répondant à la requête** s'affiche alors.

Un exemple de résultats obtenus lors de la recherche du terme « local » est présenté sur l'illustration suivante :

Fig. 5.31 – Résultat de l'utilisation du moteur de recherche.

Dans cette page, sur chaque ligne, figurent :

— le nom de l'équipement ;
— un lien vers la page de supervision (voir la section *Consultation d'une page de supervision*) ;
— un lien vers la métrologie de l'hôte, c'est-à-dire la liste de ses graphes (voir la section *Consultation de l'ensemble des graphes de métrologie d'un hôte*).

Un clic sur l'un de ces liens remplace la page courante par la page demandée.

5.3.4 Résolution de problèmes

5.3.5 Annexes

Glossaire - Terminologie

Ce chapitre recense les différents termes techniques employés dans ce document et donne une brève définition de chacun de ces termes.

API Interface logicielle de programmation, permettant à un développeur d'enrichir la liste des fonctionnalités proposées par un logiciel.

CGI Interface standard de communication entre un serveur web et un programme capable de générer une réponse HTTP valide. Il s'agit par exemple de l'interface retenue par Nagios (< 3.3) pour la génération de ses pages web.

CSS Feuilles de styles permettent de modifier la représentation graphique des éléments d'une page web. La version généralement supportée par les navigateurs est la version 2, définie par le document disponible sur http://www.w3.org/TR/CSS2/.

CSV À l'origine, désigne un format textuel de transfert de données dans lequel les entrées sont séparées par des retours chariot et les champs par des virgules (comma). De nos jours, désigne plus généralement un format tabulé pour l'échange de données en vue de leur traitement dans un logiciel de type tableur ou par un traitement automatisé (scripts).

DN Identifiant unique dans le cadre d'un annuaire LDAP.

Événement brut Alerte envoyée par Nagios au corrélateur de Vigilo pour analyse.

Événement corrélé Incident détecté par Vigilo suite à la corrélation des alertes Nagios (évènements bruts). Un événement corrélé est causé par un unique événement brut (exemple : la panne d'un routeur), mais de nombreux autres évènements bruts peuvent lui être rattachés (exemple : les alertes indiquant que les serveurs situés derrière le routeur en panne sont indisponibles). Ces événements secondaires rattachés à l'événement corrélé sont alors appelés « événements bruts masqués ».

KDC Serveur permettant un transfert sécurisé des clés de chiffrement utilisées pour les communications entre divers services. Ce serveur est notamment utilisé lors des échanges initiaux du protocole Kerberos.

LDAP Protocole pour l'interrogation d'un annuaire, servant généralement à recenser les utilisateurs autorisés d'un système et les différentes propriétés associées à ces utilisateurs.

OS Système d'exploitation.

Nagios Composant libre de supervision système et réseau.

RRD Base de données de taille fixe utilisant des fichiers circulaires, dont les données sont progressivement compressées (avec perte) au fur et à mesure de leur vieillissement.

RRDtool Composant libre de gestion de bases RRD (stockage, restitution, génération de graphiques).

SGBD[R] Logiciel permettant d'héberger des bases de données [relationnelles] sur une machine.

SQL Langage de requêtes structuré pour l'interrogation d'une base de données relationnelle.

URL Chaîne de caractères permettant d'identifier une ressource, par exemple une page web sur Internet. Exemple : http://www.vigilo-nms.com/

WSGI Une interface pour la communication entre une application et un serveur web, similaire à CGI. Il s'agit de l'interface utilisée par Vigilo.

Liste des URL

Le tableau suivant recense les URL disponibles dans VigiGraph. Les paramètres obligatoires de l'URL sont indiqués en gras (dans la colonne du milieu).

Tableau 5.4 – Liste des URL

Fonctionnalité	URL	Détail
Interface principale	<prefix_url>/	L'interface principale d'accès à VigiGraph s'affiche.
Recherche d'un hôte ou d'un ensemble d'hôtes.	<prefixe_url>/rpc/searchHost ?**query=<intitulé_query>	VigiGraph affiche l'ensemble des hôtes dont le nom contient <intitulé_query>, sans distinction majuscules/minuscules.
Image d'un graphe au format PNG.	<prefixe_url>/rpc/getImage_png ?**host=<intitulé_host>&**graph=<intitulé_graph>&sta	Paramètres : <intitulé_host> = serveur cible - <intitulé_graph> = graphe - <intitulé_start> = date-heure de début du graph - <intitulé_duration> = durée du graphe
Export CSV.	<prefixe_url>/rpc/exportCSV ?**host=<intitulé_host>&**indicator**=<intitulé_indicator>	Paramètres : — <intitulé_host> = serveur cible — <intitulé_indicator> = indicateur associé à un graphe Particularités : — Pour un export sur l'ensemble des indicateurs, <intitulé_indicator> est renseigné avec la chaîne « all». Exemple : localhost :8082/rpc/exportCSV ?host=par.linux0&indicator=

Les intitulés du type ... permettent le paramétrage de l'URL. Ils sont de type chaîne de caractères.

CHAPITRE 6

Développement

Table des matières :

6.1 Guide de développement

Ce document a pour objectif de présenter le fonctionnement du connector-nagios aux développeurs désireux d'étendre les fonctionnalités du connecteur.

6.1.1 Installation

Pré-requis logiciels

Ce document suppose que vous disposez déjà d'une installation fonctionnelle du connector-nagios, dont vous souhaitez étendre les fonctionnalités.

Reportez-vous au manuel administrateur du connector-nagios et à la procédure d'installation générale de Vigilo pour plus d'information sur la manière d'installer le connector-nagios.

6.1.2 Format des messages transmis au connecteur

Le chapitre qui suit détaille le format des différents messages que le composant « connector-nagios » est capable de traiter. Bien que ces messages soient principalement envoyés au connecteur par l'outil de supervision Nagios, le format utilisé est très simple ce qui permet par exemple de générer des alertes depuis n'importe quel autre programme.

Les messages sont composés de plusieurs champs, séparés par des barres verticales (« | », soit la combinaison de touches AltGr+6).

Messages de changement d'état

Les messages de changement d'état contiennent les informations nécessaires pour connaître le nouvel état d'un élément supervisé du parc.

Chacun de ces messages est de la forme :

```
event|<horodatage>|<hôte>|<service>|<état>|<message>
```

Le premier champ est fixe et permet d'identifier le type de message. Pour un changement d'état, le code associé est « event ».

Le second champ contient un horodatage de l'événement (timestamp), sous la forme d'un horodatage UNIX (le nombre de secondes écoulées depuis le 1er Janvier 1970).

Le troisième champ contient le nom de l'hôte (serveur, routeur, etc.) concerné par le changement d'état.

Le quatrième champ contient le nom du service concerné par le changement d'état sur l'hôte précité. Si le changement d'état concerne directement l'hôte, utilisez une chaîne vide ou utilisez le mot clé « HOST » en guise de nom de service impacté.

Le cinquième champ contient le nom du nouvel état Nagios dans lequel se trouve l'élément (en majuscules). Les états valides dépendent du type d'élément (hôte ou service). Les différents états possibles sont les suivants :

UP (hôte) / OK (service)

: L'hôte ou le service se trouve dans son état nominal.

UNREACHABLE (hôte) / UNKNOWN (service)

: L'état de l'hôte ou du service ne peut pas être déterminé.

WARNING (service uniquement)

: Le service est dans un état d'alerte mais continue de fonctionner.

DOWN (hôte) / CRITICAL (service)

: L'hôte ne répond plus ou le service est dans un état critique dans lequel il risque de ne plus pouvoir remplir ses fonctions.

Le sixième champ contient un message décrivant la raison du changement d'état. Le message ne doit contenir que des caractères imprimables (en particulier, il ne doit pas contenir de retour à la ligne) et ne peut pas contenir le caractère utilisé comme séparateur de champs (« | »).

Exemple d'un changement d'état indiquant un retour à la normale du la charge sur 5 minutes (service « Load 05 ») sur le serveur « host1.example.com » :

```
event|1290420699|host1.example.com|Load 05|OK|OK: Load at 0.7
```

Il est usuel de reprendre le nom de l'état Nagios (ici, « OK ») dans le message d'état (dans cet exemple, « OK : Load at 0.7 »), mais il ne s'agit en rien d'une obligation.

Messages de métrologie

Les messages de métrologie contiennent des indications sur les performances d'un élément du parc.

Chacun de ces messages est de la forme :

```
perf|<horodatage>|<hôte>|<service>|<valeur>
```

Le premier champ est fixe et permet d'identifier le type de message. Pour une donnée de métrologie, le code associé est « perf ».

Le second champ contient un horodatage de l'événement (timestamp), sous la forme d'un horodatage UNIX (le nombre de secondes écoulées depuis le 1er Janvier 1970).

Le troisième champ contient le nom de l'hôte (serveur, routeur, etc.) concerné par le changement d'état.

Le quatrième champ contient le nom du service concerné par le changement d'état sur l'hôte précité. Si le changement d'état concerne directement l'hôte, utilisez une chaîne vide ou utilisez le mot clé « HOST » en guise de nom de service impacté.

Le cinquième champ contient la nouvelle valeur associée à cet indicateur de métrologie. Il s'agit d'un entier ou d'une nombre flottant, en fonction du type d'indicateur.

Exemple d'un message contenant une donnée de métrologie sur le service « TCP Connections » (le nombre de connexions TCP ouvertes sur un serveur) de l'hôte « host1.example.com » :

```
perf|1290421397|host1.example.com|TCP Connections|10
```

6.1.3 Simulation de pannes

Afin de simuler des scénarios de pannes ou la réception de données de métrologie, vous pouvez utiliser l'outil « socat » de Linux et les indications sur le format des différents messages acceptés par le connecteur (voir chapitre *Format des messages transmis au connecteur*).

Exemple d'utilisation de la commande « socat » pour envoyer une alerte sur l'état de l'hôte « host1.example.com » :

```
echo "event|`date +%s`|host1.example.com||DOWN|DOWN: indisponible" \
    | socat - UNIX-CONNECT:/var/lib/vigilo/connector-nagios/send.sock
```

Notez l'utilisation d'une pipe UNIX (« | ») pour passer le message formaté par la commande « echo » à la commande « socat ».

La commande « date » est exécuté grâce à des apostrophes inversées (« ` ») afin de récupérer un horodatage dans le format attendu.

Le socket UNIX « /var/lib/vigilo/connector-nagios/send.sock » est passé en argument à la méthode « UNIX-CONNECT » de socat. Cette valeur correspond à l'option « listen_unix » de la configuration du connector-nagios. Elle doit être adaptée en fonction de votre installation.

6.2 Guide de développement

6.2.1 Ajout de sous-contrôleurs personnalisés

Les différentes interfaces web de Vigilo intègrent un contrôleur spécial appelé `custom` et qui permet quant à lui d'intégrer des sous-contrôleurs personnalisés.

Ces sous-contrôleurs doivent être déclarés en tant que points d'entrée Python, au sein du groupe *`application`*`.controllers`, où *application* correspond au nom en minuscules de l'application (par exemple, "vigiboard"). De plus, le nom du point d'entrée doit respecter les règles de nommage suivantes :

- Il doit être unique.
- Il doit commencer par un caractère alphabétique (a-zA-Z).
- Il ne doit être constitué que de caractères alphanumériques ou du symbole underscore (a-zA-Z0-9_).

Le nom du point d'entrée deviendra le nom permettant d'accéder au sous-contrôleur une fois celui-ci chargé. La valeur associée au point d'entrée doit être le nom de la classe définissant le contrôleur à charger. Celle-ci doit obligatoirement hériter de la classe `vigilo.turbogears.controllers.BaseController` pour être reconnue comme étant un contrôleur valide.

Dans cet exemple, on supposera que la classe `ExampleController` contenue dans le module Python `my_ext.foo` est un contrôleur qui étend l'application VigiBoard en fournissant une méthode `test` qui affiche une page web d'exemple. Pour ajouter ce contrôleur en tant que sous-contrôleur de `custom` avec le nom `example`, on utilisera la section `entry_points` suivante dans le fichier `setup.py` de définition du projet :

```
entry_points={
    'vigiboard.controllers': [
        'example = my_ext.foo:ExampleController',
        # Autres contrôleurs personnalisés à charger.
    ],
    # Autres groupes de points d'entrée.
},
```

Une fois l'extension installée, il est nécessaire de recharger le processus Apache à l'aide de la commande suivante :

```
# Sous RedHat Enterprise Linux ou équivalent
service httpd reload
# Sous Debian ou équivalent
service apache2 reload
```

La méthode `test` du nouveau sous-contrôleur est alors accessible à partir de l'adresse http://example.com/vigilo/vigiboard/custom/example/test (en supposant que VigiBoard a été installé à l'adresse http://example.com/vigilo/vigiboard/)

6.2.2 Annexes

Glossaire - Terminologie

Ce chapitre recense les différents termes techniques employés dans ce document et donne une brève définition de chacun de ces termes.

API Interface logicielle de programmation, permettant à un développeur d'enrichir la liste des fonctionnalités proposées par un logiciel.

CGI Interface standard de communication entre un serveur web et un programme capable de générer une réponse HTTP valide. Il s'agit par exemple de l'interface retenue par Nagios (< 3.3) pour la génération de ses pages web.

CSS Feuilles de styles permettent de modifier la représentation graphique des éléments d'une page web. La version généralement supportée par les navigateurs est la version 2, définie par le document disponible sur http://www.w3.org/TR/CSS2/.

CSV À l'origine, désigne un format textuel de transfert de données dans lequel les entrées sont séparées par des retours chariot et les champs par des virgules (comma). De nos jours, désigne plus généralement un format tabulé pour l'échange de données en vue de leur traitement dans un logiciel de type tableur ou par un traitement automatisé (scripts).

DN Identifiant unique dans le cadre d'un annuaire LDAP.

Événement brut Alerte envoyée par Nagios au corrélateur de Vigilo pour analyse.

Événement corrélé Incident détecté par Vigilo suite à la corrélation des alertes Nagios (évènements bruts). Un événement corrélé est causé par un unique événement brut (exemple : la panne d'un routeur), mais de nombreux autres évènements bruts peuvent lui être rattachés (exemple : les alertes indiquant que les serveurs situés derrière le routeur en panne sont indisponibles). Ces événements secondaires rattachés à l'événement corrélé sont alors appelés « événements bruts masqués ».

KDC Serveur permettant un transfert sécurisé des clés de chiffrement utilisées pour les communications entre divers services. Ce serveur est notamment utilisé lors des échanges initiaux du protocole Kerberos.

LDAP Protocole pour l'interrogation d'un annuaire, servant généralement à recenser les utilisateurs autorisés d'un système et les différentes propriétés associées à ces utilisateurs.

OS Système d'exploitation.

Nagios Composant libre de supervision système et réseau.

RRD Base de données de taille fixe utilisant des fichiers circulaires, dont les données sont progressivement compressées (avec perte) au fur et à mesure de leur vieillissement.

RRDtool Composant libre de gestion de bases RRD (stockage, restitution, génération de graphiques).

SGBD[R] Logiciel permettant d'héberger des bases de données [relationnelles] sur une machine.

SQL Langage de requêtes structuré pour l'interrogation d'une base de données relationnelle.

URL Chaîne de caractères permettant d'identifier une ressource, par exemple une page web sur Internet. Exemple : http://www.vigilo-nms.com/

WSGI Une interface pour la communication entre une application et un serveur web, similaire à CGI. Il s'agit de l'interface utilisée par Vigilo.

6.3 Manuel développeur

6.3.1 Chargement des fichiers XML

Les fichiers XML se situent dans le répertoire `/etc/vigilo/vigiconf/conf.d/`.

On y trouve notamment les répertoires principaux suivants :
- définition des hôtes `/etc/vigilo/vigiconf/conf.d/hosts`
- definition des groupes d'hôtes `/etc/vigilo/vigiconf/conf.d/groups/`
- définition des modèles d'équipements `/etc/vigilo/vigiconf/conf.d/templates/`

La documentation sur la syntaxe des fichiers de configuration XML est disponible dans le *manuel utilisateur de VigiConf*. Le lecteur s'y référera au besoin.

Les fichiers XML sont parcourus par VigiConf. La structure d'exécution est la suivante :
- Analyse des fichier XML.
- Prise en compte des tests de supervision Python.
- Exécution des générateurs.

6.3.2 Création d'un nouveau test de supervision

Introduction

Les tests de supervision de VigiConf sont écrits en Python. Ils permettent de générer des fichiers de configuration concernant une ou plusieurs applications.

Organisation des tests de supervision

Principe des classes d'équipements

Les tests fournis par Vigilo sont regroupés dans le répertoire suivant : `/usr/lib/pythonx.y/site-packages/vigilo/vigiconf/tests/`

On y trouve d'autres répertoires :

— all/
— linux/
— lmsensors/
— ucd/

Ces répertoires correspondent à des classes d'équipements et contiennent des tests de supervision spécifiques à un type (famille/gamme/modèle) d'équipement particulier. Par exemple, le dossier `linux/` contient des tests de supervision capables de fonctionner sur toutes les distributions Linux.

Le dossier `all/` est un cas particulier : les tests de supervision qu'il contient peuvent être appliqués à n'importe quel type d'équipement (système ou réseau).

Note : Par convention, les noms des classes d'équipements ne peuvent contenir que les caractères alphanumériques (a-z0-9) et l'underscore (_). De plus, nous recommandons d'inscrire les noms des classes en minuscules.

Avertissement : Pour des raisons historiques, certaines classes d'équipements actuellement fournies avec Vigilo contiennent des traits d'union (-) dans leur nom. Cet usage est cependant déconseillé et pourrait ne plus être supporté dans les futures versions de Vigilo. Utilisez le caractère underscore (_) à la place.

Les fichiers contenus dans ces répertoires portent des noms représentatifs du test de supervision appliqué (par exemple, `CPU.py` pour un test de supervision portant sur l'état du processeur sur l'équipement).

Le même nom de fichier peut apparaître dans plusieurs classes d'équipements différentes (par exemple, `linux/CPU.py` et `ucd/CPU.py`).

Ajout ou personnalisation d'un test

L'administrateur de la solution a la possibilité de définir ses propres tests de supervision ou bien de redéfinir les tests fournis par défaut. Les ajouts/surcharges de tests doivent se faire dans le dossier `/etc/vigilo/vigiconf/conf.d/tests` en respectant la même arborescence *`classe/test`*`.py`.

Ainsi, le test portant sur le processeur de la classe d'équipements "maclasse" se trouvera dans le fichier `/etc/vigilo/vigiconf/conf.d/tests/maclasse/CPU.py`.

De même, on pourra redéfinir le comportement du test de supervision d'un processeur sur une distribution Linux en créant un fichier `/etc/vigilo/vigiconf/conf.d/tests/linux/CPU.py` contenant le code modifié.

Structure du code d'un test

Tous les tests de supervision héritent de la classe `vigilo.vigiconf.lib.confclasses.test.Test` et contiennent une méthode `add_test()` qui intègre la logique du test.

Exemple de test de supervision de la charge du processeur sur un nouveau modèle d'équipements de la marque "exemple". Le test sera placé dans `/etc/vigilo/vigiconf/conf.d/tests/exemple/CPU.py`.

```
# -*- coding: utf-8 -*-
from vigilo.vigiconf.lib.confclasses.test import Test

class CPU(Test):
    """
    Description de ce que fait le test, par exemple :
    Supervision de la charge du processeur sur un équipement
    de marque exemple.
    """
```

```
def add_test(self, warn=60, crit=80):
    """
    Description des arguments acceptés par ce test de supervision.

    @param warn: Seuil de charge CPU en pourcentage
        au-delà duquel un avertissement sera levé
        par Vigilo dans le bac à événements.
    @type warn: C{float}
    @param crit: Seuil de charge CPU en pourcentage
        au-delà duquel une alerte critique sera levée
        par Vigilo dans le bac à événements.
    @type crit: C{float}
    """
    # Généralement le code commence par convertir les paramètres
    # reçus vers les types attendus, comme ci-dessous.
    warn = self.as_float(warn)
    crit = self.as_float(crit)

    # Code intégrant la logique de test de la charge CPU
    # sur ce type d'équipements.
```

Note : Si vous souhaitez redistribuer votre test de supervision à l'équipe en charge du développement de Vigilo par la suite, nous vous recommandons d'adopter la syntaxe de documentation Epytext, comme dans l'exemple précédent. Il s'agit du format de documentation de l'API adopté par l'équipe pour le développement des nouveaux tests de supervision.

Note : Si vous créez un test de supervision similaire à un test déjà existant (par exemple, un nouveau test portant sur le CPU), nous vous recommandons de garder la même signature pour la fonction `add_test()`, afin de rendre homogène la configuration du test pour l'utilisateur, quelle que soit la classe d'équipements sous-jacente.

La classe **doit** avoir exactement le même nom que le fichier dans lequel elle se trouve. Il s'agit également du nom qui devra être utilisé par l'administrateur dans les fichiers de configuration XML pour pouvoir appliquer ce test de supervision à un équipement.

Les arguments passés à la méthode `add_test()` seront ceux indiqués dans les fichiers XML de configuration qui utilisent ce test.

Avertissement : Les arguments passés à la fonction `add_test()` seront systématiquement des chaînes de caractères, y compris pour des valeurs numériques (par exemple, des seuils d'alerte).

Le code de la méthode doit donc effectuer les conversions nécessaires avant d'utiliser ces valeurs. Le chapitre *Méthodes de l'instance de test* décrit les méthodes de la classe `Test` pouvant être utilisées afin de réaliser les conversions adéquates.

Méthodes de l'instance de test

Chaque instance de la classe `Test` ou d'une classe dérivée possède plusieurs méthodes outils.

Ces méthodes peuvent être regroupées en 2 catégories :

— Les méthodes permettant de convertir les paramètres passés à un test vers un type donné (entier, flottant, booléen).
— Les méthodes permettant d'ajouter les éléments de configuration nécessaires pour réaliser les tests à proprement parler, récupérer des informations sur les performances (métrologie), générer des graphiques à partir de ces informations, etc.

La liste suivante décrit les méthodes utilisables pour effectuer la conversion des paramètres d'un test.

as_bool() Convertit la valeur passée en argument en booléen. Les valeurs `1`, `true`, `on` et `yes` représentent la valeur Python `True`, tandis que les valeurs `0`, `false`, `off` et `no` représentent la valeur Python `False`. Toute autre valeur génèrera une erreur d'analyse.

Note : L'analyse effectuée est insensible à la casse (ie. `yes` == `Yes`).

Note : Si la valeur donnée est déjà un booléen, elle est retournée sans qu'aucune conversion ne soit effectuée.

---

as_float() Convertit la valeur passée en argument en nombre flottant. Si la valeur ne peut être convertie, une erreur d'analyse est levée.

Note : Si la valeur donnée est déjà un flottant, elle est retournée sans qu'aucune conversion ne soit effectuée.

---

as_int() Convertit la valeur passée en argument en nombre entier. Si la valeur ne peut être convertie, une erreur d'analyse est levée.

Note : Si la valeur donnée est déjà un entier, elle est retournée sans qu'aucune conversion ne soit effectuée.

La liste suivante décrit les méthodes appartenant à la deuxième catégorie (manipulation des tests de supervision et de la métrologie).

add_collector_metro() Ajoute un service passif de métrologie à Nagios.

add_collector_service() Ajoute un service passif à Nagios et ajoute un test supplémentaire au collector (SNMP).

add_collector_service_and_metro() Cette méthode est un raccourci qui correspond à une succession d'appels aux méthodes suivantes :
— `add_collector_service()`
— `add_collector_metro()`

add_collector_service_and_metro_and_graph() Cette méthode est un raccourci qui correspond à une succession d'appels aux méthodes suivantes :
— `add_collector_service()`
— `add_collector_metro()`
— `add_graph()`

add_external_sup_service() Ajoute un service actif à Nagios.

add_graph() Ajoute un graphique à Vigigraph.

add_metro_service() Ajoute un test Nagios sur les valeurs contenues dans les fichiers RRD.

add_perfdata() Déclare une donnée de performance reçu par le connecteur de métrologie, permettant ainsi de faire le lien entre les données de performance et les base RRDs de métrologie.

add_perfdata_handler() Déclare une donnée de performance générée par un module Nagios dans Vigilo, permettant ainsi de faire le lien entre les données de performance des modules Nagios et les bases RRDs de métrologie.

add_trap() Ajoute un service passif dans Nagios, dont l'état changera sur réception d'un trap SNMP. Cette méthode ajoute également des éléments de configuration dans le fichier `snmptt.conf` utilisé pour traiter les traps SNMP.

Note : Un service Nagios actif est un service pour lequel l'exécution du test de supervision associé est déclenchée par Nagios. Un service Nagios passif est un service dont le test de supervision n'est pas déclenché par Nagios. À la place, la valeur d'état du service est calculée indépendamment de Nagios, puis injectée dans celui-ci. Par exemple, les services dont l'état est déterminé grâce à SNMP sont déclarés comme des services passifs dans Nagios. Le collector SNMP de Vigilo (service actif) est appelé par Nagios, se charge de calculer les états de ces services, puis envoie ces états à Nagios.

Note : Toutes ces fonctions appellent la méthode add qui va ajouter/modifier la hash_map. La structure de la hash_map est disponible au début du fichier /lib/confclasses/host.py. La méthode add est également déclarée dans ce même fichier, elle va simplement ajouter un élément, dans l'une des clefs de la hash_map. En cas de non existence de la clef, elle l'ajoute. Le développeur peut appeler directement la méthode **add** dans son test Python, mais cela nécessite une connaissance de la structure de la hash_map. La hash_map est l'équivalent d'un dictionnaire Python, et est parcourue par les générateurs pour la génération de configuration. Avec un seul test Python, on peut, par exemple, ajouter des services passifs/actifs à Nagios, tout en plaçant des informations dans la hash_map, qui sera lu par notre générateur qui créera un fichier de configuration spécifique.

6.3.3 Générateurs de fichiers de configuration

Un certain nombre de générateurs de fichiers de configuration sont fournis par défaut avec VigiConf. Ces générateurs vont créer des fichiers de configuration pour les applications et composants qui interagissent avec Vigilo (collecteur, connecteur de métrologie, Nagios, etc.).

Pour créer un nouveau générateur de fichiers de configuration, vous devez créer un nouveau point d'entrée sous la section `vigilo.vigiconf.generators`. Pour l'heure, le nom de la clé associée au point d'entrée n'a pas d'importance.

Les générateurs doivent hériter de la classe `vigilo.vigiconf.lib.generators.Generator` ou d'une sous-classe (par exemple, `vigilo.vigiconf.lib.generators.file.FileGenerator`).

À minima, le générateur doit surcharger la méthode `generate()` de la classe `Generator` afin d'effectuer les traitements nécessaires à la création d'une nouvelle configuration.

La méthode `generate_host()` est également utilisable. Cette méthode génère la configuration pour un hôte et le serveur de supervision qui lui est associé.

Les générateurs de fichiers de configuration sont regroupés dans le répertoire `/usr/lib/python`*x.y*`/site-packages/vigilo/vigiconf/applications/`, où *x.y* représente la version de Python utilisée (par exemple, 2.5 ou 2.6).

Architecture d'un générateur

L'ajout d'un générateur se fait par la création d'un nouveau répertoire dans `applications/`. Le dossier est organisé selon l'arborescence suivante :

```
./applications/premier_generateur/
    |- __init__.py
    |- generator.py
    |- templates/
    |   |- template_un.tpl
    |   `- template_deux.tpl
    `- validate.sh
```

Le fichier `__init__.py`

Ce fichier est quasiment identique entre les différents générateurs et décrit l'application pour laquelle des fichiers de configuration peuvent être générés.

Le fichier doit contenir une classe Python héritant de la classe `vigilo.vigiconf.lib.application.Application`. Cette classe ne contient aucune méthode, mais simplement des attributs permettant de décrire l'application.

Par exemple, pour une application nommée `Foobar` effectuant une collecte d'états, le fichier `__init__.py` pourrait contenir un code semblable à l'extrait suivant :

```
# -*- coding: utf-8 -*-

from __future__ import absolute_import

from vigilo.vigiconf.lib.application import Application
from . import generator

class Foobar(Application):
    name = "foobar"
    priority = -1
    validation = "validate.sh"
    start_command = "start.sh"
    stop_command = "stop.sh"
    generator = generator.FoobarGenerator
    group = "collect"
```

Le rôle des différents attributs de cette classe est décrit dans le tableau ci-dessous.

Tableau 6.1 – Attributs des sous-classes de `Application`

Attribut	Rôle
`dbonly`	Drapeau indiquant si l'application manipule des données externes à la base de données Vigilo. Ce drapeau permet de paralléliser l'exécution des applications qui ne travaillent que sur la base de données Vigilo et de désactiver leur exécution lorsqu'une bascule de serveur est nécessaire (haute disponibilité).
`defaults`	Paramêtres de configuration par défaut de l'application et indépendants de la configuration du parc.
`generator`	La classe Python correspondant au générateur de configuration à proprement parler pour cette application.
`group`	Le groupe fonctionnel auquel l'application appartient. Les groupes actuellement définis par Vigilo sont : — `collect`, pour les applications relatives à la collecte d'états ou de données de métrologie (Nagios, Collector, etc.). — `metrology`, pour les applications qui sont responsables du stockage et de la restitution des données de métrologie (connector-metro et VigiRRD). — `interface`, pour les interfaces graphiques dont le contenu est partiellement généré automatiquement grâce à VigiConf (VigiMap).
`name`	Un nom unique pour faire référence à cette application. Le nom ne doit contenir **que** des caractères en minuscules.
`priority`	Priorité pour l'ordonnancement du redémarrage. L'application avec la priorité la plus élevée sera qualifiée et déployée en premier.
`start_command`	Une commande qui sera exécutée sur les serveurs de supervision où l'application est installée afin de relancer l'application. Cet attribut peut valoir `None` si l'application n'a pas besoin d'être arrêtée puis relancée pour prendre en compte une nouvelle configuration.
`stop_command`	Une commande qui sera exécutée sur les serveurs de supervision où l'application est installée afin d'arrêter l'application. Cet attribut peut valoir `None` si l'application n'a pas besoin d'être arrêtée puis relancée pour prendre en compte une nouvelle configuration.
`validation`	Une commande qui sera exécutée sur la machine où VigiConf est installé et sur les serveurs de supervision où l'application est installée afin de vérifier la validité des fichiers de configuration générés.

Avertissement : Le nom de la classe (`Foobar` dans notre exemple) **doit** commencer par une majuscule.

Note : Les groupes fonctionnels sont utilisés pour répartir la supervision du parc entre différents serveurs de supervision. Cette répartition est effectuée grâce à la configuration située dans le fichier `/etc/vigilo/vigiconf/`

`conf.d/general/appgroups-server.py`.

Le dossier `templates/`

Le dossier `templates/` contient un ensemble de fichiers de « modèles » portant l'extension `.tpl` (pour « template »).

Reportez-vous à la section sur l'*Utilisation des modèles* dans un générateur pour plus d'information sur le mécanisme des modèles.

Le fichier `generator.py`

Ce fichier contient le générateur à proprement parler. C'est lui qui va générer le(s) fichier(s) de configuration de l'application.

Imports utiles

Le fichier `generator.py` importe généralement plusieurs objets et modules de Vigilo, comme indiqué dans l'extrait de code suivant :

```
# Si l'application utilise des données provenant du fichier
# de configuration de VigiConf (/etc/vigilo/vigiconf/settings.ini).
from vigilo.common.conf import settings

# Cet import donne accès à la hashmap utilisée pour transmettre
# la configuration des différents hôtes.
from vigilo.vigiconf import conf

# Import de la classe permettant de générer des fichiers de configuration.
from vigilo.vigiconf.lib.generators import FileGenerator

# Pour des besoins plus spécifiques, on peut également importer
# directement la classe de base des générateurs
#from vigilo.vigiconf.lib.generators import Generator
```

Classe du générateur

Le fichier du générateur doit contenir une classe qui hérite de la classe `vigilo.vigiconf.lib.generators.Generator` ou de l'une de ses sous-classes (par exemple `vigilo.vigiconf.lib.generators.FileGenerator`).

Note : Par convention, la classe correspondant au générateur pour une application nommée `Foobar` dans Vigilo s'appelle `FoobarGen`.

Cette classe doit définir au minimum deux méthodes :

- `generate()`
- `generate_host()`

La méthode `generate()` est appelée exactement une fois pour chaque serveur de supervision vers laquelle une nouvelle configuration doit être déployée, au début de la génération. La plupart du temps, cette méthode se contente de

réinitialiser la liste des fichiers de configuration à générer (attribut `_files`), puis appelle la méthode `generate()` de la classe parente.

Par exemple, le générateur de configuration pour Nagios contient :

```
def generate(self):
    # pylint: disable-msg=W0201
    self._files = {}
    self._graph = None
    super(NagiosGen, self).generate()
```

La méthode `generate_host()` est appelée pour chaque hôte pour lequel une configuration doit être générée. La méthode reçoit en argument le nom du serveur pour lequel une configuration doit être générée, ainsi que le nom du serveur de supervision vers lequel cette configuration sera déployée :

```
def generate_host(self, hostname, vserver):
    # Le code permettant de générer la configuration pour l'hôte
    # "hostname" sur le serveur de supervision "vserver" se trouve ici.
    pass
```

Les méthodes du générateur seront donc appelées successivement selon le motif suivant :

```
# Début de la nouvelle configuration pour un serveur de supervision.
generator.generate()
# Génération de la configuration pour les différents équipements
# supervisés par le serveur "sup1.example.com"
generator.generate_host("serveur1.example.com", "sup1.example.com")
generator.generate_host("serveur2.example.com", "sup1.example.com")
generator.generate_host("serveur3.example.com", "sup1.example.com")

# Début de la nouvelle configuration pour un serveur de supervision.
generator.generate()
# Génération de la configuration pour les différents équipements
# supervisés par le serveur "sup2.example.com"
generator.generate_host("serveur4.example.com", "sup2.example.com")
generator.generate_host("serveur5.example.com", "sup2.example.com")
generator.generate_host("serveur6.example.com", "sup2.example.com")

# etc.
```

Utilisation des modèles

Pour générer le ou les fichiers de configuration dont il est responsable, un générateur peut utiliser le mécanisme des modèles de documents proposé par Vigilo. Les modèles de documents sont des fichiers portant l'extension « .tpl » et placés dans le répertoire `templates/` à côté du générateur.

Un modèle de document peut représenter la totalité d'un fichier de configuration ou bien simplement un fragment de ce fichier. Dans le second cas, cela signifie que plusieurs modèles de documents devront être utilisés successivement afin de générer un fichier de configuration complet.

Un modèle de document est similaire à « un texte à trous », au sens où il s'agit d'un exemple du contenu d'un fichier de configuration dans lequel certains champs dont le contenu est variable (par exemple, un nom de machine) sont remplacés par des champs de formatage Python (`%(foo)s`). Le contenu des champs de formatage Python sera substitué par les valeurs adéquates lors de l'utilisation du modèle par le générateur.

Exemple de modèle (ici, le modèle `host.tpl` responsable de la création de la configuration d'un hôte dans Nagios) :

```
define host{
    use %(hostTPL)s
    host_name %(name)s
    alias %(name)s
    address %(address)s
    %(hostGroups)s
    %(parents)s
    %(generic_hdirectives)s
}
```

Dans cet exemple, on peut voir que le modèle attend plusieurs variables :

- Le nom du modèle Nagios à utiliser : `hostTPL`.
- Le nom de l'équipement en cours de configuration : `name`.
- L'adresse de l'équipement : `address`.
- D'autres variables : `hostGroups`, `parents`, etc.

L'utilisation d'un modèle de documents dans un générateur est plutôt simple. L'extrait de code (simplifié) suivant donne un exemple d'utilisation de plusieurs modèles par le générateur Nagios :

```
def generate_host(self, hostname, vserver):
    # Détermination du chemin jusqu'au fichier de configuration à générer.
    # - self.baseDir correspond à la racine du dossier de génération
    #   des configurations.
    # - vserver contient le nom du serveur de supervision vers lequel
    #   ce fichier de configuration sera déployé.
    # L'affectation du chemin dans un attribut de la classe (self.fileName)
    # est facultative : on aurait pu utiliser une variable locale.
    self.fileName = os.path.join(self.baseDir, vserver, "nagios",
                                 "nagios.cfg")

    if self.fileName not in self._files:
        self._files[self.fileName] = {}

    # Récupération de la configuration relative à l'hôte
    # pour lequel on est en train de générer une configuration.
    h = conf.hostsConf[hostname]

    # Si le fichier de configuration n'existe pas encore
    # (ie. on commence la création de la configuration).
    if not os.path.exists(self.fileName):
        # Alors, on utilise la méthode templateCreate() pour créer
        # le fichier de configuration à partir du modèle contenu
        # dans "templates/header.tpl".
        self.templateCreate(self.fileName, self.templates["header"] {,
                "confid": conf.confid,
            })

    # Préparation des variables qui seront passées
    # au modèle "templates/host.tpl".
    newhash = self._prepare_template_variables(h.copy())

    # Ajout de la configuration de l'hôte au fichier de configuration,
    # en utilisant le modèle "templates/host.tpl".
    self.templateAppend(self.fileName, self.templates['host'], newhash)
```

En somme, la signature des méthodes `templateCreate()` et `templateAppend()` est identique, les 2 méthodes acceptant les arguments suivants (dans cet ordre) :

- Le nom du fichier de configuration qui sera créé (`templateCreate()`) ou complété

(`templateAppend()`).
— Le nom du modèle de document à utiliser, **sans extension**.
— Un dictionnaire dont les clés sont les champs de substitution définis dans le modèle de document et les valeurs sont celles que doivent prendre les champs de substitution.

Le fichier de validation

Ce fichier correspond à la valeur de l'attribut `validation` *dans la classe qui décrit l'application* et son nom doit donc être adapté en conséquence. En général, il s'agit d'un simple script shell appelé `validate.sh`.

Le but de ce fichier est d'effectuer des vérifications permettant de s'assurer que la nouvelle configuration est valide (bon format, bonnes options, etc.).

Ce fichier sera exécuté une première fois sur la machine qui héberge VigiConf (phase de validation), puis une seconde fois sur la machine de supervision finale (phase de qualification) afin d'être absolument certain que la nouvelle configuration pourra être appliquée.

Une rapide lecture du contenu des scripts des autres générateurs facilite la création d'un nouveau script de validation.

6.3.4 Générateur de cartes automatiques

VigiConf fournit un mécanisme permettant de générer automatiquement des cartes, construites à partir des données présentes dans la configuration (groupes, hôtes, services ou toute autre donnée).

Un générateur de cartes automatiques est une classe contenue dans le paquet `automaps` et dérivant de la classe de base `AutoMap`.

CHAPITRE 7

Glossaire

— genindex
— search

Index

Symbols

A

C

D

E

I

J

K

L

N

O

R

S

U

W

X