

FusionInventory

Guillaume Rouse

<guillomovitch@gmail.com>

Journées francophones de Perl 2011



Sommaire

- 1 Solutions de gestion de parc informatique
- 2 Présentation de l'agent
 - Caractéristiques
 - Capacités
 - Objectifs
- 3 Implémentation de l'agent
 - Un lourd héritage
 - Stratégie de test

Avancement

- 1 Solutions de gestion de parc informatique
- 2 Présentation de l'agent
 - Caractéristiques
 - Capacités
 - Objectifs
- 3 Implémentation de l'agent
 - Un lourd héritage
 - Stratégie de test

GLPI

Objectifs

Application web de gestion de parc

- inventaire
- helpdesk

Détails

- application PHP
- modulaire



GLPI

Objectifs

Application web de gestion de parc

- inventaire
- helpdesk

Détails

- application PHP
- modulaire



OCS-NG

Objectifs

Solution autonome de gestion de parc

- inventaire
- déploiement logiciel

Détails

Coté serveur :

- interface agent Perl
- interface utilisateur PHP

Coté agent :

- agent Windows C
- agent Unix Perl

OCS_{next} generation
inventory

OCS-NG

Objectifs

Solution autonome de gestion de parc

- inventaire
- déploiement logiciel

Détails

Coté serveur :

- interface agent Perl
- interface utilisateur PHP

Coté agent :

- agent Windows C
- agent Unix Perl

OCS_{next} generation
inventory

Tracker

Objectifs

Inventaire automatisé pour GLPI de matériels sans agent

- découverte matériel
- interrogation distante

Détails

- plugin GLPI PHP
- agent Perl

Tracker

Objectifs

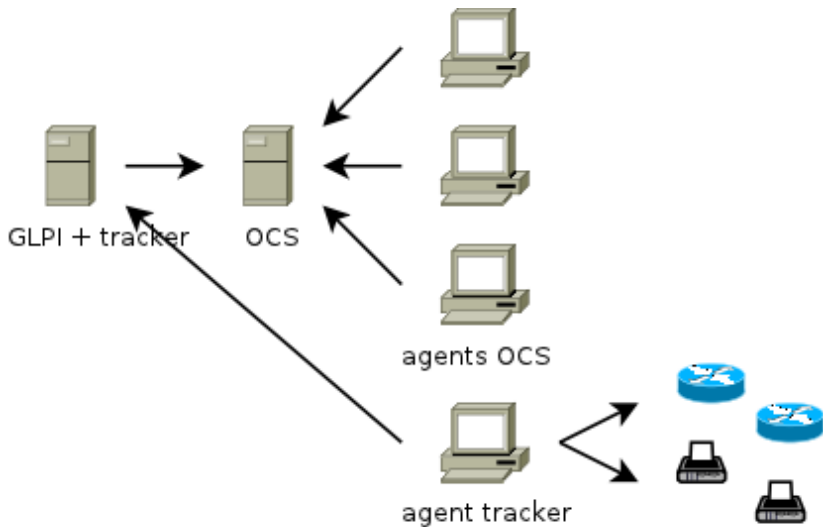
Inventaire automatisé pour GLPI de matériels sans agent

- découverte matériel
- interrogation distante

Détails

- plugin GLPI PHP
- agent Perl

Avant-hier



FusionInventory

Objectifs

Agent multifonction pour GLPI

- fusion de l'agent OCS Unix et de l'agent tracker
- modulaire

Détails

- plugin GLPI PHP
- agent Perl multi-plateformes



FusionInventory

Objectifs

Agent multifonction pour GLPI

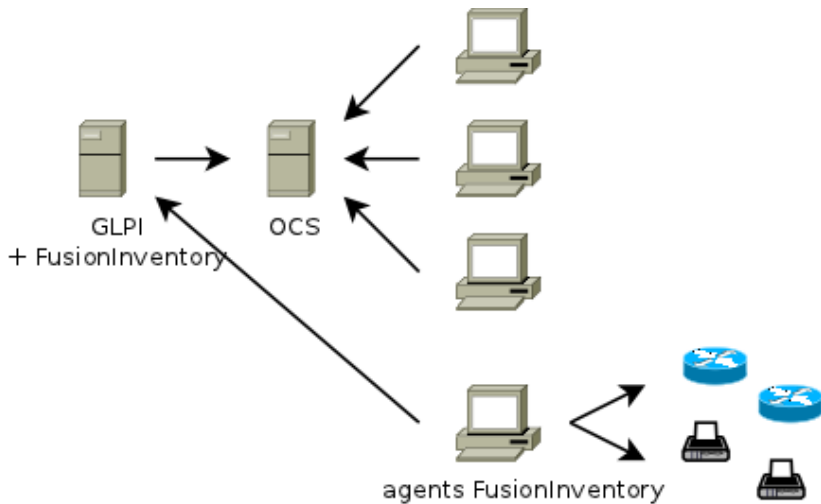
- fusion de l'agent OCS Unix et de l'agent tracker
- modulaire

Détails

- plugin GLPI PHP
- agent Perl multi-plateformes



Hier



Évolution du plugin GLPI

Branches

- 2.2.x
compatibilité GLPI 0.72.x
- 2.3.x
compatibilité GLPI 0.78.x, gestion des inventaires
- 2.4.x (en beta)
compatibilité GLPI 0.80.x

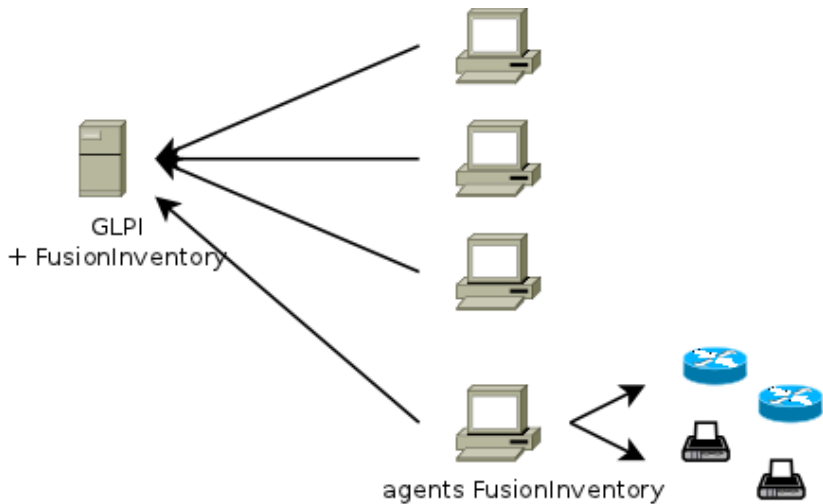
Capture d'écran du plugin GLPI

The screenshot displays the FusionInventory plugin interface within the GLPI web application. The interface is organized into several sections:

- FusionInventory**: Contains four main management areas: Agents management, Task management, Running jobs, and Unknown device.
- FusionInventory DEPLOY**: Focuses on deployment tasks, including Package management, Files management, and Deployment state.
- FusionInventory SHIMP**: Manages discovery and inventory criteria, including SNMP models, SNMP authentication, IP range configuration, Discovery criteria rules, Inventory criteria rules, Discovery status, and Network inventory status.
- FusionInventory INVENTORY**: Manages inventory data, including Import agent XML file, Criteria rules, Entity rules, and BlackList.

The interface includes a top navigation bar with tabs for Inventory, Assistance, Management, Tools, Plugins, Administration, and Setup. A search bar is located in the top right corner. The bottom of the interface features a set of navigation icons.

Aujourd'hui



Avancement

- 1 Solutions de gestion de parc informatique
- 2 **Présentation de l'agent**
 - Caractéristiques
 - Capacités
 - Objectifs
- 3 Implémentation de l'agent
 - Un lourd héritage
 - Stratégie de test

Avancement

- 1 Solutions de gestion de parc informatique
- 2 **Présentation de l'agent**
 - **Caractéristiques**
 - Capacités
 - Objectifs
- 3 Implémentation de l'agent
 - Un lourd héritage
 - Stratégie de test

Modularité

Composants

- moteur
- taches

Taches

- inventaire local
- wake on lan
- découverte réseau
- inventaire distant
- déploiement logiciel

Modularité

Composants

- moteur
- taches

Taches

- inventaire local
- wake on lan
- découverte réseau
- inventaire distant
- déploiement logiciel

Versatilité

Mode d'exécution

- processus temporaire
- processus résidant (daemon Unix, service Windows)

Mode de collecte du résultat

- stockage local
- envoi au serveur

Versatilité

Mode d'exécution

- processus temporaire
- processus résidant (daemon Unix, service Windows)

Mode de collecte du résultat

- stockage local
- envoi au serveur

Portabilité

Cible des résultats

- distribution
- exécution

Exécution

- Windows
- Linux
- MacOS
- BSD
- Solaris
- HPUX
- AIX

Distribution

- sources
- paquetages
- installeur interactif
- archive autonome

Portabilité

Cible des résultats

- distribution
- exécution

Exécution

- Windows
- Linux
- MacOS
- BSD
- Solaris
- HPUX
- AIX

Distribution

- sources
- paquetages
- installeur interactif
- archive autonome

Portabilité

Cible des résultats

- distribution
- exécution

Exécution

- Windows
- Linux
- MacOS
- BSD
- Solaris
- HPUX
- AIX

Distribution

- sources
- paquetages
- installeur interactif
- archive autonome

Interopérabilité

Serveur OCS

- compatibilité : version serveur 1.x
- fonctionnalités :
 - inventaire local
 - déploiement logiciel

Serveur GLPI

- version serveur 0.72 :
 - découverte réseau
 - inventaire distant
- version serveur 0.78 : fonctionnalités complètes

Interopérabilité

Serveur OCS

- compatibilité : version serveur 1.x
- fonctionnalités :
 - inventaire local
 - déploiement logiciel

Serveur GLPI

- version serveur 0.72 :
 - découverte réseau
 - inventaire distant
- version serveur 0.78 : fonctionnalités complètes

Avancement

- 1 Solutions de gestion de parc informatique
- 2 **Présentation de l'agent**
 - Caractéristiques
 - **Capacités**
 - Objectifs
- 3 Implémentation de l'agent
 - Un lourd héritage
 - Stratégie de test

Inventaire local

Composants matériels

- CPU, mémoire, etc...
- bus USB, PCI, SCSI
- périphériques connectés

Composants logiciels

- système d'exploitation
- liste de paquetages

Configuration

- paramètre réseau
- paramètre affichage
- variables d'environnement

Inventaire local

Composants matériels

- CPU, mémoire, etc...
- bus USB, PCI, SCSI
- périphériques connectés

Composants logiciels

- système d'exploitation
- liste de paquetages

Configuration

- paramètre réseau
- paramètre affichage
- variables d'environnement

Inventaire local

Composants matériels

- CPU, mémoire, etc...
- bus USB, PCI, SCSI
- périphériques connectés

Composants logiciels

- système d'exploitation
- liste de paquetages

Configuration

- paramètre réseau
- paramètre affichage
- variables d'environnement

Découverte réseau

Principe

- balayage d'une plage réseau
- identification des éléments trouvés

Balayage

- nmap
- NetBIOS
- SNMP

Identification

- SNMP
- attribution d'un modèle descriptif

Découverte réseau

Principe

- balayage d'une plage réseau
- identification des éléments trouvés

Balayage

- nmap
- NetBIOS
- SNMP

Identification

- SNMP
- attribution d'un modèle descriptif

Découverte réseau

Principe

- balayage d'une plage réseau
- identification des éléments trouvés

Balayage

- nmap
- NetBIOS
- SNMP

Identification

- SNMP
- attribution d'un modèle descriptif

Inventaire distant

Principe

- SNMP
- utilisation d'un modèle descriptif

Matériel réseau

- ports utilisés
- équipements branchés

Imprimantes

- compteurs d'impression
- niveaux d'encre

Inventaire distant

Principe

- SNMP
- utilisation d'un modèle descriptif

Matériel réseau

- ports utilisés
- équipements branchés

Imprimantes

- compteurs d'impression
- niveaux d'encre

Inventaire distant

Principe

- SNMP
- utilisation d'un modèle descriptif

Matériel réseau

- ports utilisés
- équipements branchés

Imprimantes

- compteurs d'impression
- niveaux d'encre

Déploiement logiciel

Principe

- téléchargement d'une charge utile
- exécution

Intérêt

- téléchargement pair à pair
- exécution immédiate
- utilisation des critères de sélection de GLPI

Déploiement logiciel

Principe

- téléchargement d'une charge utile
- exécution

Intérêt

- téléchargement pair à pair
- exécution immédiate
- utilisation des critères de sélection de GLPI

Avancement

- 1 Solutions de gestion de parc informatique
- 2 **Présentation de l'agent**
 - Caractéristiques
 - Capacités
 - **Objectifs**
- 3 Implémentation de l'agent
 - Un lourd héritage
 - Stratégie de test

Versions

Branches

- 2.0.x version initiale
- 2.1.x branche stable
ajout du support Windows, nettoyage syntaxique
- 2.2.x branche développement
nettoyage architecture, maintien de la compatibilité OCS
- 3.x branche expérimentale
changement architecture, abandon de la compatibilité OCS

Version stable

2.1.9

Avancement

- 1 Solutions de gestion de parc informatique
- 2 Présentation de l'agent
 - Caractéristiques
 - Capacités
 - Objectifs
- 3 Implémentation de l'agent
 - Un lourd héritage
 - Stratégie de test

Avancement

- 1 Solutions de gestion de parc informatique
- 2 Présentation de l'agent
 - Caractéristiques
 - Capacités
 - Objectifs
- 3 **Implémentation de l'agent**
 - **Un lourd héritage**
 - Stratégie de test

Support syslog

FusionInventory::LoggerBackend::Syslog

```
setlogsock('unix');  
openlog("fusioninventory-agent", 'cons,pid', $ENV{'USER'});  
syslog('debug', 'syslog_backend_enabled');  
closelog();
```

Extraction des logiciels installés sous HPUX

FusionInventory::Agent::Task::Inventory::OS::HPUX::Software

```
@softList = 'swlist | grep -v '^_PH' | grep -v '^#' | tr -s "\t" "_" | tr -s "_"'
```

Export de fonctions

FusionInventory::Agent::Task::Inventory

```
my $backendSharedFuncs = {  
    can_run => sub {  
        ...  
    },  
};  
  
foreach my $package (@packages) {  
    foreach my $func (keys %{$backendSharedFuncs}) {  
        $package->{$func} = $backendSharedFuncs->{$func};  
    }  
}
```

Grand chantier

Nettoyage du code

- syntaxe
- structuration
- architecture

Amélioration des performances

- mémoisation
- indexation

Amélioration fonctionnelles

- interface REST
- remplacement de XML par JSON
- modèle conceptuel homogène

Grand chantier

Nettoyage du code

- syntaxe
- structuration
- architecture

Amélioration des performances

- mémoisation
- indexation

Amélioration fonctionnelles

- interface REST
- remplacement de XML par JSON
- modèle conceptuel homogène

Grand chantier

Nettoyage du code

- syntaxe
- structuration
- architecture

Amélioration des performances

- mémoisation
- indexation

Amélioration fonctionnelles

- interface REST
- remplacement de XML par JSON
- modèle conceptuel homogène

Avancement

- 1 Solutions de gestion de parc informatique
- 2 Présentation de l'agent
 - Caractéristiques
 - Capacités
 - Objectifs
- 3 Implémentation de l'agent
 - Un lourd héritage
 - **Stratégie de test**

Suite de tests

État actuel

- 1330 tests, 76 fichiers
- couverture très hétérogène

Problèmes

- tester des interactions client-serveur
- tester l'analyse du résultat d'une commande
- tester la réponse à un environnement

Suite de tests

État actuel

- 1330 tests, 76 fichiers
- couverture très hétérogène

Problèmes

- tester des interactions client-serveur
- tester l'analyse du résultat d'une commande
- tester la réponse à un environnement

Tester des interactions client-serveur

Serveur de test

- Test::Apache2 :(
- Net::Server::HTTP :(
- HTTP::Server::Simple :)

Fonctionnalités additionnelles

- HTTP::Server::Simple::Authen
- IO::Socket::SSL
- HTTP::Proxy

Utilisation de véritable serveur

- serveur public ou privé
- test obligatoire ou conditionnel

Tester des interactions client-serveur

Serveur de test

- Test::Apache2 :(
- Net::Server::HTTP :(
- HTTP::Server::Simple :)

Fonctionnalité additionnelles

- HTTP::Server::Simple::Authen
- IO::Socket::SSL
- HTTP::Proxy

Utilisation de véritable serveur

- serveur public ou privé
- test obligatoire ou conditionnel

Tester des interactions client-serveur

Serveur de test

- Test::Apache2 :(
- Net::Server::HTTP :(
- HTTP::Server::Simple :)

Fonctionnalité additionnelles

- HTTP::Server::Simple::Authen
- IO::Socket::SSL
- HTTP::Proxy

Utilisation de véritable serveur

- serveur public ou privé
- test obligatoire ou conditionnel

Tester l'analyse du résultat d'une commande

Objectif

- tester le code qui analyse le résultat d'une commande

Code initial

```
my @output = 'command';  
foreach my $line (@output) {  
    if ($line =~ /^foo: (.*)/) {  
        $inventory->addItem($1);  
    }  
}
```

Tester l'analyse du résultat d'une commande

Objectif

- tester le code qui analyse le résultat d'une commande

Code initial

```
my @output = 'command';  
foreach my $line (@output) {  
    if ($line =~ /^foo: (.*)/) {  
        $inventory->addItem($1);  
    }  
}
```

Adapter le code au test

Code intermédiaire

```
$inventory->addItems($_) foreach getItems ();  
  
sub getItems {  
  my @output = 'command';  
  my @items;  
  foreach my $line (@output) {  
    push @items, $1 if $line =~ /^foo: (.*)/;  
  }  
  return @items  
}
```

Code final

```
$inventory->addItems($_) foreach getItems(command => 'command');  
  
sub getItems {  
  my $handle = getFileHandle(@_);  
  while (my $line = <$handle>) {  
    push @items, $1 if $line =~ /^foo: (.*)/;  
  }  
  close $handle;  
  return @items;  
}
```

Adapter le code au test

Code intermédiaire

```
$inventory->addItems($_) foreach getItems ();  
  
sub getItems {  
  my @output = 'command';  
  my @items;  
  foreach my $line (@output) {  
    push @items, $1 if $line =~ /^foo: (.*)/;  
  }  
  return @items  
}
```

Code final

```
$inventory->addItems($_) foreach getItems(command => 'command');  
  
sub getItems {  
  my $handle = getFileHandle(@_);  
  while (my $line = <$handle>) {  
    push @items, $1 if $line =~ /^foo: (.*)/;  
  }  
  close $handle;  
  return @items;  
}
```

Utiliser une couche d'abstraction

getFileHandle

```
sub getFileHandle {
    my %params = @_;

    my $handle;

    SWITCH: {
        if ($params{file}) {
            if (!open $handle, '<', $params{file}) {
                $params{logger}->error("Can't open file $params{file}:$_SERRNO");
                return;
            }
            last SWITCH;
        }
        if ($params{command}) {
            if (!open $handle, '|-', $params{command} . "_2>/dev/null") {
                $params{logger}->error("Can't run command $params{command}:$_SERRNO");
                return;
            }
            last SWITCH;
        }
        die "neither_command_nor_file_parameter_given";
    }

    return $handle;
}
```

Problème de disponibilité du résultat

Commandes et fichiers exotiques

- lsvpd sur AIX
- machinfo sur HPUX
- /proc/cpuinfo sur Linux Alpha

Campagnes de collecte d'échantillons

- répertoire ressources
- besoin d'organisation (tri, tracabilité, ...)

Problème sous Windows

- base de registre
- WMI

Problème de disponibilité du résultat

Commandes et fichiers exotiques

- lsvpd sur AIX
- machinfo sur HPUX
- /proc/cpuinfo sur Linux Alpha

Campagnes de collecte d'échantillons

- répertoire ressources
- besoin d'organisation (tri, tracabilité, ...)

Problème sous Windows

- base de registre
- WMI

Problème de disponibilité du résultat

Commandes et fichiers exotiques

- lsvpd sur AIX
- machinfo sur HPUX
- /proc/cpuinfo sur Linux Alpha

Campagnes de collecte d'échantillons

- répertoire ressources
- besoin d'organisation (tri, tracabilité, ...)

Problème sous Windows

- base de registre
- WMI

Tester la réponse à un environnement

Objectif

- tester le code qui sélectionne la commande à utiliser

Code initial

```
my @packages =
  -x '/bin/rpm' ? getRPMPackagesList (command => 'rpm_qa' ) :
  -x '/bin/dpkg' ? getDPKGPackagesList (command => 'dpkg_l' ) :
  -x '/bin/equery' ? getEqueryPackagesList (command => 'equery_list_i' ) :
  () ;
}
```

Tester la réponse à un environnement

Objectif

- tester le code qui sélectionne la commande à utiliser

Code initial

```
my @packages =
  -x '/bin/rpm'      ? getRPMPackagesList (command => 'rpm_qa'      ) :
  -x '/bin/dpkg'    ? getDPKGPackagesList (command => 'dpkg_-l'    ) :
  -x '/bin/equery' ? getEqueryPackagesList (command => 'equery_list_-i') :
                    () ;
}
```

Adapter son code

Code final

```
my @packages =
  canRun('/bin/rpm') ? getRPMPackagesList (command => 'rpm_qa' ) :
  canRun('/bin/dpkg') ? getDPKGPackagesList (command => 'dpkg_l' ) :
  canRun('/bin/equery') ? getEqueryPackagesList (command => 'equery_list_i' ) :
  () ;
}
```

Utiliser une couche d'abstraction

canRun

```
sub canRun {  
  my ($wanted) = @_;  
  return -x $wanted;  
}
```

Redéfinir des fonctions

mockGetRun

```
sub mockCanRun {  
  my (%params) = @_;  
  
  my $new = sub {  
    my $wanted = $_[0];  
    return $params{commands}->{$wanted};  
  };  
  
  no warnings 'redefine';  
  *FusionInventory::Agent::Tools::canRun = $new;  
}
```

Redéfinir des fonctions

mockGetFileHandle

```
sub mockGetFileHandle {
  my (%params) = @_;

  my $old = \&FusionInventory::Agent::Tools::getFileHandle;

  my $new = sub {
    my (%options) = @_;

    my $file = $params{commands}->{$wanted};

    if ($file) {
      print STDERR "file_'$file'_delivered\n";
      return $old->(@_, file => $file);
    } else {
      print STDERR "nothing_delivered\n";
      return;
    }
  };

  no warnings 'redefine';
  *FusionInventory::Agent::Tools::getFileHandle = $new;
}
```

Simuler un environnement

Code

```
package FusionInventory :: Test :: MockSystem :: Debian ;  
  
use FusionInventory :: Test :: MockSystem ;  
  
mockSystem(  
  commands => {  
    'dpkg' => 'resources/packaging/dpkg'  
  },  
  files => {  
    '/etc/debian_version' => 'resources/release/debian'  
  }  
);
```

Utilisation

```
$> perl -MFusionInventory::Test::MockSystem::Debian  
fusioninventory
```

Simuler un environnement

Code

```
package FusionInventory :: Test :: MockSystem :: Debian ;

use FusionInventory :: Test :: MockSystem ;

mockSystem(
  commands => {
    'dpkg' => 'resources/packaging/dpkg'
  },
  files => {
    '/etc/debian_version' => 'resources/release/debian'
  }
);
```

Utilisation

```
$> perl -MFusionInventory::Test::MockSystem::Debian
    fusioninventory
```