Intégration GLPI avec Shinken via des gestionnaires d'événements personnalisés

Shinken est les successeur de Nagios dans le domaine de la surveillance des réseaux et des ressources et GLPI est un système d'assistance basé sur ITIL, mais aucun des deux ne fournit une intégration immédiate. Pour couvrir le besoin d'alerter le service d'assistance lorsque Shinken a détecté une défaillance du système, on peut programmer l'envoi d'une alerte e-mail à une boîte aux lettre que GLPI pourrait récupérer cet e-mail et créer un nouveau ticket.

Mais cela risque d'innondé le service d'assistance de billets, il est donc préférable d'utiliser un gestionnaire d'événement de service et d'hôte personnalisé qui ouvre et ferme les tickets sur GLPI.

Les gestionnaires d'événements ci-dessous créeront de nouveaux tickets GLPI pour les événements Shinken et les fermeront une fois le problème résolu. Ces scripts sont écrits en PHP et nécessitent un accès direct à la base de données GLPI MySQL. Cette solution nécessite également le plugin de services Web sur le serveur GLPI.

- télécharger les scripts php du gestionnaire d'événements
- modifier nom d'utilisateur, mot de passe et adresse IP de serveur GPLI
- placer les fichiers dans le dossier `/var/lib/shinken/libexec/eventhandlers/`.

manage-host-tickets.php

```
<?php
// -----------------------------------------------------------------------
---------------------
// Script Name:         manage-host-tickets.php
// Script Location:     /usr/share/nagios3/plugins/eventhandlers/
// Description:         Creates or closes glpi tickets according to
host UP/Down states.
// Dependancies:        GLPI Webservices plugin.
//
// Required CommandLine Variables:
//      eventhost=$HOSTNAME$
//      event=$HOSTSTATE$
//      state=$HOSTSTATETYPE$
//      hostattempts=$HOSTATTEMPTS$
//      maxhostattempts=$MAXHOSTATTEMPTS$
//      hostproblemid=$HOSTPROBLEMID$
//      lasthostproblemid=$LASTHOSTPROBLEMID$

// Email Notifications:  For email notifications to work you must
assign either a category name

// -----------------------------------------------------------------------
---------------------
// Configurables:
// -----------------------------------------------------------------------
```

```
---------------------
$user =         "glpi";                            //GLPI User Account
- REQUIRED
$password =     "glpi";                          //GLPI User Password -
REQUIRED
$xmlhost =      "localhost/";                       //GLPI Server
HOSTNAME/IP - REQUIRED
$xmlurl =       "plugins/webservices/xmlrpc.php";    //Path to xmlrpc
on GLPI server - REQUIRED

// --------------------------------------------------------------------
---------------------
// Do Not Edit Below!
// --------------------------------------------------------------------
---------------------

$arg['method'] = "glpi.test";
$arg['url'] = $xmlurl;
$arg['host'] = $xmlhost;
$response = call_glpi($arg);
unset($arg);
$webservices_version = $response['webservices'];
    //0.2.0 Method added
    //1.2.0 Added type, source, requester and observer options

$eventval=array();
if ($argv>1) {
    for ($i=1 ; $i<count($argv) ; $i++) {
        $it = explode("=",$argv[$i],2);
        $it[0] = preg_replace('/^--/','',$it[0]);
        $eventval[$it[0]] = (isset($it[1]) ? $it[1] : true);
    }
}

$eventhost=$eventval['eventhost'];
$event=$eventval['event'];
$state=$eventval['state'];
$hostattempts=$eventval['hostattempts'];
$maxhostattempts=$eventval['maxhostattempts'];
$hostproblemid=$eventval['hostproblemid'];
$lasthostproblemid=$eventval['lasthostproblemid'];
unset($eventval);

function call_glpi($args) {
   global $deflate,$base64;
   $url=$args['url'];
   $host=$args['host'];

   echo "+ Calling {$args['method']} on http://$host/$url\n";

   if (isset($args['session'])) {
      $url_session = $url.'?session='.$args['session'];
   } else {
      $url_session = $url;
   }

   $header = "Content-Type: text/xml";

   if (isset($deflate)) {
      $header .= "\nAccept-Encoding: deflate";
   }

   $request = xmlrpc_encode_request($args['method'], $args);
   $context = stream_context_create(array('http' => array('method'  =>
"POST",
                                                          'header'  =>
$header,
                                                          'content' =>
$request)));

   $file = file_get_contents("http://$host/$url", false, $context);
   if (!$file) {
      die("+ No response\n");
   }

   if (in_array('Content-Encoding: deflate', $http_response_header)) {
      $lenc=strlen($file);
      echo "+ Compressed response : $lenc\n";
      $file = gzuncompress($file);
      $lend=strlen($file);
      echo "+ Uncompressed response : $lend
(".round(100.0*$lenc/$lend)."%)\n";
   }
   $response = xmlrpc_decode($file);
   if (!is_array($response)) {
      echo $file;
      die ("+ Bad response\n");
   }
   if (xmlrpc_is_fault($response)) {
       echo("xmlrpc error(".$response['faultCode']."):
".$response['faultString']."\n");
   } else {
      return $response;
   }
}

if (!extension_loaded("xmlrpc")) {
   die("Extension xmlrpc not loaded\n");
}

# What state is the HOST in?
switch ($event) {
    case "UP":
        # The host just came back up - perhaps we should close the
ticket...
        if ($lasthostproblemid != 0) {
            $arg['method'] = "glpi.doLogin";
            $arg['url'] = $xmlurl;
            $arg['host'] = $xmlhost;
            $arg['login_password'] = $password;
            $arg['login_name'] = $user;

            $response = call_glpi($arg);
            $session = $response['session'];

            unset($arg);
            unset($response);
            if (!empty($session)){
                $arg['method'] = "glpi.listTickets";
                $arg['url'] = $xmlurl;
                $arg['host'] = $xmlhost;
                $arg['session'] = $session;
                $arg['order'] = "id";
                $arg['status'] = '1';

                $response = call_glpi($arg);

                unset($arg);
                foreach ($response as $ticket) {
                    if ($ticket['name'] == "$eventhost is down!") {
                        $fields = array ('Ticket' => array (array ('id'
=> $ticket['id'], 'status' => '6')));
                        $arg['method'] = "glpi.updateObjects";
                        $arg['url'] = $xmlurl;
                        $arg['host'] = $xmlhost;
                        $arg['session'] = $session;
                        $arg['fields'] = $fields;

                        $response = call_glpi($arg);
                        unset($arg);
                        unset($response);
                    }
                }
            }
            $arg['method'] = "glpi.doLogout";
            $arg['url'] = $xmlurl;
            $arg['host'] = $xmlhost;
            $arg['session'] = $session;

            $response = call_glpi($arg);
            unset($arg);
            unset($response);
        }
        break;
    //case "UNREACHABLE":
        # We don't really care about warning states, since the host is
probably still running...
    case "DOWN":
        # Aha!  The host appears to have a problem - perhaps we should
open a ticket...
        # Is this a "soft" or a "hard" state?
            switch ($state) {
                # We're in a "soft" state, meaning that Nagios is in
the middle of retrying the
                # check before it turns into a "hard" state and
contacts get notified...

                //case "SOFT":
                    # We don't want to open a ticket on a "soft" state.

                case "HARD":
                    if ($lasthostproblemid != 1) {
                        if ($hostattempts == $maxhostattempts){
                            $arg['method'] = "glpi.doLogin";
                            $arg['url'] = $xmlurl;
                            $arg['host'] = $xmlhost;
                            $arg['login_password'] = $password;
                            $arg['login_name'] = $user;

                            $response = call_glpi($arg);
                            $session = $response['session'];

                            unset($arg);
                            unset($response);
                            if (!empty($session)) {
                                $title = "$eventhost is down!";
                                $content = "$eventhost is down.  Please
check that the server is up and responding";
                                $arg['method'] = "glpi.createTicket";
                                $arg['url'] = $xmlurl;
                                $arg['host'] = $xmlhost;
                                $arg['session'] = $session;
                                $arg['title'] = $title;
                                $arg['content'] = $content;
                                $arg['urgancy'] = 5;
                                $arg['use_email_notification'] = 1;
                                $response = call_glpi($arg);
                                unset($arg);
                                unset($response);
                            }
                            $arg['method'] = "glpi.doLogout";
                            $arg['url'] = $xmlurl;
                            $arg['host'] = $xmlhost;
                            $arg['session'] = $session;
                            $response = call_glpi($arg);
                            unset($arg);
                            unset($response);
                        }
                    }
                //end state cases
            }
    //end event cases
}


?>
```

manage-service-tickets.php

```
<?php
// -----------------------------------------------------------------------
--------------------
// Script Name:         manage-service-tickets.php
// Script Location:     /usr/share/nagios3/plugins/eventhandlers/
// Description:         Creates or closes glpi tickets according to
service UP/Down states.
// Dependancies:        GLPI Webservices plugin.
//
// Required CommandLine Variables:
//      eventhost=$HOSTNAME$
//      event=$SERVICESTATE$
//      state=$SERVICESTATETYPE$
//      service=$SERVICEDISPLAYNAME$
//      serviceattempts=$SERVICEATTEMPTS$
//      maxserviceattempts=$MAXSERVICEATTEMPTS$
//      serviceproblemid=$SERVICEPROBLEMID$
//      lastserviceproblemid=$LASTSERVICEPROBLEMID$

// -----------------------------------------------------------------------
---------------------
// Configurables:
// -----------------------------------------------------------------------
---------------------
$user =         "glpi";                             //GLPI User Account
- REQUIRED
$password =     "glpi";                         //GLPI User Password -
REQUIRED
$xmlhost =      "localhost/";                       //GLPI Server
HOSTNAME/IP - REQUIRED
$xmlurl =       "plugins/webservices/xmlrpc.php";    //Path to xmlrpc
on GLPI server - REQUIRED



// -----------------------------------------------------------------------
```

```
---------------------
// Do Not Edit Below!
// ----------------------------------------------------------------------
---------------------

$arg['method'] = "glpi.test";
$arg['url'] = $xmlurl;
$arg['host'] = $xmlhost;
$response = call_glpi($arg);
unset($arg);
$webservices_version = $response['webservices'];
    //0.2.0 Method added
    //1.2.0 Added type, source, requester and observer options

$eventval=array();
    if ($argv>1) {
       for ($i=1 ; $i<count($argv) ; $i++) {
          $it = explode("=",$argv[$i],2);
          $it[0] = preg_replace('/^--/','',$it[0]);
          $eventval[$it[0]] = (isset($it[1]) ? $it[1] : true);
       }
    }

print_r($eventval);

$eventhost=$eventval['eventhost'];
$event=$eventval['event'];
$hoststate=$eventval['hoststate'];
$service=$eventval['service'];
$servicestate=$eventval['state'];
$serviceattempts=$eventval['serviceattempts'];
$maxserviceattempts=$eventval['maxserviceattempts'];
$servicestate=$eventval['servicestate'];
$lastservicestate=$eventval['lastservicestate'];
$servicecheckcommand=$eventval['servicecheckcommand'];
$longserviceoutput=$eventval['longserviceoutput'];

unset($eventval);



function call_glpi($args) {
   global $deflate,$base64;
   $url=$args['url'];
   $host=$args['host'];

   echo "+ Calling {$args['method']} on http://$host/$url\n";

   if (isset($args['session'])) {
      $url_session = $url.'?session='.$args['session'];
   } else {
      $url_session = $url;
   }

   $header = "Content-Type: text/xml";

   if (isset($deflate)) {
      $header .= "\nAccept-Encoding: deflate";
   }

   $request = xmlrpc_encode_request($args['method'], $args);
   $context = stream_context_create(array('http' => array('method'  =>
"POST",
                                                          'header'  =>
$header,
                                                          'content' =>
$request)));

   $file = file_get_contents("http://$host/$url", false, $context);
   if (!$file) {
      die("+ No response\n");
   }

   if (in_array('Content-Encoding: deflate', $http_response_header)) {
      $lenc=strlen($file);
      echo "+ Compressed response : $lenc\n";
      $file = gzuncompress($file);
      $lend=strlen($file);
      echo "+ Uncompressed response : $lend
(".round(100.0*$lenc/$lend)."%)\n";
   }
   $response = xmlrpc_decode($file);
   if (!is_array($response)) {
      echo $file;
      die ("+ Bad response\n");
   }
   if (xmlrpc_is_fault($response)) {
       echo("xmlrpc error(".$response['faultCode']."):
".$response['faultString']."\n");
   } else {
      return $response;
   }
}

if (!extension_loaded("xmlrpc")) {
   die("Extension xmlrpc not loaded\n");
}



// What state is the SERVICE in?
if (($hoststate == "UP")) {  // Only open tickets for services on hosts
```

```
that are UP
    echo "Host is UP \n";
    switch ($event) {
        case "OK":
            echo "Event is OK \n";
            if (($lastservicestate == "CRITICAL")) {
                // The service just came back up - perhaps we should
close the ticket...
                $arg['method'] = "glpi.doLogin";
                $arg['url'] = $xmlurl;
                $arg['host'] = $xmlhost;
                $arg['login_password'] = $password;
                $arg['login_name'] = $user;

                $response = call_glpi($arg);
                $session = $response['session'];

                unset($arg);
                unset($response);
                if (!empty($session)) {
                    $arg['method'] = "glpi.listTickets";
                    $arg['url'] = $xmlurl;
                    $arg['host'] = $xmlhost;
                    $arg['session'] = $session;
                    $arg['order'] = "id";
                    $arg['status'] = 1;

                    $response = call_glpi($arg);
                    echo "XXX1: "; print_r($arg);

                    unset($arg);

                    foreach ($response as $ticket) {

                        echo $ticket['name']."\n";
                        echo "$service on $eventhost is in a Critical
State!\n";

                        if ($ticket['name'] == "$service on $eventhost
is in a Critical State!") {
                            $fields = array ('Ticket' => array (array
('id' => $ticket['id'], 'status' => '6')));
                            $arg['method'] = "glpi.updateObjects";
                            $arg['url'] = $xmlurl;
                            $arg['host'] = $xmlhost;
                            $arg['session'] = $session;
                            $arg['fields'] = $fields;

                            echo "XXX2: ";print_r($arg);

                            $response = call_glpi($arg);
                            echo "XXX2: ";print_r($arg);

                            unset($arg);
                            unset($response);
                        }
                    }
                }
                $arg['method'] = "glpi.doLogout";
                $arg['url'] = $xmlurl;
                $arg['host'] = $xmlhost;
                $arg['session'] = $session;

                $response = call_glpi($arg);
                unset($arg);
                unset($response);
            }
            break;
        case "CRITICAL":
            echo "Event is Critical \n";
            # Aha!  The service appears to have a problem - perhaps we
should open a ticket...
            # Is this a "soft" or a "hard" state?
            echo "servicestate=".$servicestate."\n";
            switch ($servicestate) {
                //case "HARD":
                case "CRITICAL":
                    echo "$serviceattempts == $maxserviceattempts\n";

                    if ($serviceattempts == $maxserviceattempts){
                        $arg['method'] = "glpi.doLogin";
                        $arg['url'] = $xmlurl;
                        $arg['host'] = $xmlhost;
                        $arg['login_password'] = $password;
                        $arg['login_name'] = $user;

                        $response = call_glpi($arg);
                        $session = $response['session'];

                        unset($arg);
                        unset($response);
                        if (!empty($session)) {
                            $title = "$service on $eventhost is in a
Critical State!";
                            $content = "$service on $eventhost is in a
Critical State.  Please check that the service or check is running and
responding correctly \n
                                        Host \t\t\t = $eventhost \r
                                        Service Check \t = $service \r
                                        State \t\t\t = $event \r
                                        Check Attempts \t =
$serviceattempts/$maxserviceattempts \r
```

```
                                        Check Command \t =
$servicecheckcommand \r
                                        Check Output \t =
$serviceoutput \r
                                        $longserviceoutput \r
                            ";
                            echo $content."\n";

                            $arg['method'] = "glpi.createTicket";
                            $arg['url'] = $xmlurl;
                            $arg['host'] = $xmlhost;
                            $arg['session'] = $session;
                            $arg['title'] = $title;
                            $arg['content'] = $content;
                            $arg['urgancy'] = 5;
                            $arg['use_email_notification'] = 1;
                            $response = call_glpi($arg);
                            unset($arg);
                            unset($response);
                        }
                        $arg['method'] = "glpi.doLogout";
                        $arg['url'] = $xmlurl;
                        $arg['host'] = $xmlhost;
                        $arg['session'] = $session;

                        $response = call_glpi($arg);
                        unset($arg);
                        unset($response);
                    }
                    break;
            }
            break;
    } //end event cases
}


?>
```

Créer ensuite les fichier **manage-host-tickets.cfg** et **manage-service-tickets.cfg** dans `/etc/shinken/commands/` pour inclure les commandes suivantes:

manage-host-tickets.cfg

```
# définition de la commande 'manage-host-tickets'
define command{
command_name manage-host-tickets
command_line php /var/lib/shinken/libexec/eventhandlers/manage-host-
tickets.php event="$HOSTSTATE$" state="$HOSTSTATETYPE$"
eventhost="$HOSTNAME$" hostattempts="$HOSTATTEMPT$"
maxhostattempts="$MAXHOSTATTEMPTS$" hostproblemid="$HOSTPROBLEMID$"
lasthostproblemid="$LASTHOSTPROBLEMID$"
}
```

manage-service-tickets.cfg

```
define command{
command_name manage-service-tickets
command_line php /var/lib/shinken/libexec/eventhandlers/manage-service-
tickets.php event="$SERVICESTATE$" state="$SERVICESTATETYPE$"
hoststate="$HOSTSTATE$" eventhost="$HOSTNAME$"
service="$SERVICEDISPLAYNAME$" serviceattempts="$SERVICEATTEMPT$"
maxserviceattempts="$MAXSERVICEATTEMPTS$" servicestate="$SERVICESTATE$"
lastservicestate="$LASTSERVICESTATE$"
servicecheckcommand="$SERVICECHECKCOMMAND$"
serviceoutput="$SERVICEOUTPUT$" longserviceoutput="$LONGSERVICEOUTPUT$"
```

Ensuite, modifier le fichier `/etc/shinken/hosts/localhost.cfg`:

```

define host{
        name                            generic-host    ; The name of this
host template
        notifications_enabled           1           ; Host notifications are
enabled
        event_handler_enabled           1           ; Host event handler is
enabled
        flap_detection_enabled          1           ; Flap detection is
enabled
        failure_prediction_enabled      1           ; Failure prediction is
enabled
        process_perf_data               1           ; Process performance
data
        retain_status_information       1           ; Retain status
information across program restarts
        retain_nonstatus_information    1           ; Retain non-status
information across program restarts
        check_command                   check-host-alive
        event_handler                    manage-host-tickets
        max_check_attempts      1
        notification_interval   0
        notification_period     24x7
        notification_options    d,u,r
        contact_groups          admins
        register                        0           ; DONT REGISTER THIS
DEFINITION - ITS NOT A REAL HOST, JUST A TEMPLATE!
        }
```

Ajouter dans le fichier `/etc/shinken/services/services.cfg` le service suivant:

```
define service{
        name                            generic-service ; The 'name' of this
service template
        active_checks_enabled           1       ; Active service checks are
enabled
        passive_checks_enabled          1       ; Passive service checks are
enabled/accepted
        parallelize_check               1       ; Active service checks
should be parallelized (disabling this can lead to major performance
problems)
        obsess_over_service             1       ; We should obsess over this
service (if necessary)
        check_freshness                 0       ; Default is to NOT check
service 'freshness'
        notifications_enabled           1       ; Service notifications are
enabled
        event_handler_enabled           1       ; Service event handler is
enabled
        flap_detection_enabled          1       ; Flap detection is enabled
        failure_prediction_enabled      1       ; Failure prediction is
enabled
        process_perf_data               1       ; Process performance data
        retain_status_information       1       ; Retain status information
across program restarts
        retain_nonstatus_information    1       ; Retain non-status
information across program restarts
        notification_interval           0        ; Only send notifications
on status change by default.
        event_handler                    manage-service-tickets
        is_volatile                     0
        check_period                    24x7
        normal_check_interval           5
        retry_check_interval            1
        max_check_attempts              4
        notification_period             24x7
        notification_options            w,u,c,r
        contact_groups                  admins
        register                        0       ; DONT REGISTER THIS
DEFINITION - ITS NOT A REAL SERVICE, JUST A TEMPLATE!
        }
```

GLPI recevra désormais des tickets d'assistance lorsque des alertes sont créées dans SHINKEN et ces tickets seront supprimés lorsque le service ou l'hôte aura été restauré. GLPI peut désormais gérer les notifications appropriées.