```
"$SERVICESTATE$" "$LONGDATETIME$" "$SERVICEOUTPUT$" "$HOSTNAME$"
"$_SERVICENORMALCHECKINTERVAL$" "$SERVICEPERFDATA$"
"$CONTACTEMAIL$" "$SERVICENOTIFICATIONNUMBER$"
"$SERVICECHECKCOMMAND$"
}
```

Annexes
Plugin de vérification d'accès apache2.

Ce plugin parcoure le fichier de log apache et compte le nombre de ligne des 5 dernières minutes vérifiant une certaine expression régulière passée en paramètre par ip. Si une ip totalise un certain pourcentage du total de requête ou qu'il y a un nombre total de requêtes trop élevé, une alerte sera déclenchée. Le programme dissocie les requêtes externes et locales.

Check_access.java

```
package check_apache2;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.io.RandomAccessFile;
import java.util.HashMap;
import java.util.Map;

/*
 * args[0] = chemin vers dossier de logs
 * args[1] = expression reguliere
 * args[2] = seuil warning
 * args[3] = seuil critique
 * args[4] = pourcenatge warning
 * args[5] = pourcenatge critique
 * args[6] = nom fichier temporaire
 */
public class Check_apache2 {

    private static int Count = 0, CountL = 0;
    private static String FTEMPNAME;
    private static int WARNINGTOTAL;
    private static int CRITICALTOTAL;
    private static int CRITICALPOURCENTAGE;
    private static int WARNINGPOURCENTAGE;
    private static int WARNINGTOTALLOCAL;
    private static int CRITICALTOTALLOCAL;
    private static int CRITICALPOURCENTAGELOCAL;
    private static int WARNINGPOURCENTAGELOCAL;
    private static String REGEX;
```

Check_access.java

```
    private static final String APACHELOG = "access.log";
    private static final String APACHELOG1 = "access.log.1";
    private static long numOctet = -1;
    private static HashMap<String, Integer> hmapIP = new HashMap<String, Integer>();

    public static void main(String[] args) {
        boolean doubleCheck = false;
        FTEMPNAME= System.getProperty("user.home") + "/" + args[6];
        REGEX = args[1];
        WARNINGTOTAL = Integer.parseInt(args[2].split(",")[0]);
        CRITICALTOTAL = Integer.parseInt(args[3].split(",")[0]);
        WARNINGPOURCENTAGE = Integer.parseInt(args[4].split(",")[0]);
        CRITICALPOURCENTAGE = Integer.parseInt(args[5].split(",")[0]);
        WARNINGTOTALLOCAL = Integer.parseInt(args[2].split(",")[1]);
        CRITICALTOTALLOCAL = Integer.parseInt(args[3].split(",")[1]);
        WARNINGPOURCENTAGELOCAL = Integer.parseInt(args[4].split(",")[1]);
        CRITICALPOURCENTAGELOCAL = Integer.parseInt(args[5].split(",")[1]);
        try
        {
            File logApache = new File(args[0]+APACHELOG);
            File fTemp = new File(FTEMPNAME);

             if(fTemp.exists())
            {
                //Si le fichier existe
                //On va lire le numero d'octet dans le fichier
                numOctet = ReadPosition();
            }
            else
            {
                //sinon on va noter le num d'octet de la derniere ligne pour le prochain appel
                  RandomAccessFile raf = new RandomAccessFile(logApache,"r");
                numOctet = raf.length();
                  raf.close();
                WritePosition(numOctet);
                System.out.println("OK : 0 Acces | Acces=0;0;0;0  Pourcentage=0%%;0;0;0\n");
                System.exit(0);
            }
            //On regarde si il y a eu une rotation depuis le dernier appel
            logApache = new File(args[0]+APACHELOG1);
              if(logApache.exists() && logApache.lastModified() > fTemp.lastModified())//Alors
c'est dans access.log.1 puis access.log rotation apache)
                doubleCheck = true;
            else//C'est juste dans access.log
                logApache = new File(args[0]+APACHELOG);

             //compter le nombre de ligne qui vérifie l'expression réguliere
            GetCount(logApache, numOctet);
            if(doubleCheck)//On va continuer le traitement dans apache.log si il y avait eu rotation
```

Check_access.java

```
        {
           logApache = new File(args[0]+APACHELOG);
           GetCount(logApache, 0);
        }
        //sauvergarder le num de ligne
        WritePosition(numOctet);

        //Donnée pour Nagios

        //Recuperation de l'ip qui a fait le plus de requete
         String maxIP = null, maxIPL = null;
        int maxReq = 0, maxReqL = 0;
        for(Map.Entry<String, Integer> entry : hmapIP.entrySet())
        {
           String cle = entry.getKey();
           Integer val = entry.getValue();
           if(isLocalIP(cle))
           {
              if(maxIPL == null || val.intValue() > maxReqL)
              {
                 maxIPL = cle;
                 maxReqL = val.intValue();
              }
           }
           else
           {
              if(maxIP == null || val.intValue() > maxReq)
              {
                 maxIP = cle;
                 maxReq = val.intValue();
              }
           }
        }
        int pct = 0, pctL = 0;
         if(Count != 0) //division par 0
              pct = (int) (((double)maxReq/(double)Count)*100.0);//calcul du pourcentage de
requete pour cette ip
        if(CountL != 0) //division par 0
           pctL = (int) (((double)maxReqL/(double)CountL)*100.0);//calcul du pourcentage de
requete pour cette ip
         String info = "";
        if(maxReq != 0)
           info += "Ext: " + Count + ", " + pct + "%% depuis " + maxIP + ".";
        if(maxReqL != 0)
           info += " Local: " + CountL + ", " + pctL + "%% depuis " + maxIPL ;
        if(maxReq ==0 && maxReqL == 0)
           info = "0 Acces";
        String perf = "ext=" + Count + ";" + WARNINGTOTAL + ";" + CRITICALTOTAL +
";0";
```

Check_access.java

```
                perf += " extpct=" + pct + "%%;" + WARNINGPOURCENTAGE + ";" +
CRITICALPOURCENTAGE + ";0"; //label=value;warning;critical;minimum label2=....
                 perf += " local=" + CountL + ";" + WARNINGTOTALLOCAL + ";" +
CRITICALTOTALLOCAL + ";0";
          perf += " localpct=" + pctL + "%%;" + WARNINGPOURCENTAGELOCAL + ";" +
CRITICALPOURCENTAGELOCAL + ";0\n";
             if(Count > CRITICALTOTAL || pct > CRITICALPOURCENTAGE || CountL >
CRITICALTOTALLOCAL || pctL > CRITICALPOURCENTAGELOCAL)
          {
             System.out.println(info + " | " + perf);
             System.exit(2);//CRITICAL
          }
          else if(Count > WARNINGTOTAL || pct > WARNINGPOURCENTAGE || CountL >
WARNINGTOTALLOCAL || pctL > WARNINGPOURCENTAGELOCAL)
          {
             System.out.println(info + " | " + perf);
             System.exit(1);//WARNING
          }
          System.out.println(info + " | " + perf);
          System.exit(0);//OK
       }
       catch (IOException ex) {
          System.out.println(ex.getMessage());
          System.exit(3);//UNKNOWN
       }
    }

    private static void GetCount(File logApache, long num) throws IOException
    {
      RandomAccessFile raf = new RandomAccessFile(logApache,"r");
      raf.seek(num);//On se met au début de la 1ere ligne a lire
       String line;
      while((line = raf.readLine()) != null)
       {
         if(line.matches(REGEX))
         {
            //On va maintenant incrementer le nombre de requete pour cette ip
              //Le compte est fait dans une hashmap contenant come clé l'ip et comme valeur le
nombre de requetes pour cette ip
             String ip = line.split(" ")[0];
            if(isLocalIP(ip))
               CountL++;
            else
               Count++;
            Integer nb = (Integer) hmapIP.get(ip);
            if(nb == null)
               nb = new Integer(0);
            hmapIP.put(ip, new Integer(nb.intValue() + 1));
         }
```

Check_access.java

```
      }
      numOctet = raf.getFilePointer();
      raf.close();
   }

   private static void WritePosition(long num) throws IOException
   {
                        PrintWriter pw = new PrintWriter(new BufferedWriter(new
FileWriter(FTEMPNAME)));
       pw.print(num);
       pw.close();
   }

   private static long ReadPosition() throws IOException
   {
      long num;
      BufferedReader in  = new BufferedReader(new FileReader(FTEMPNAME));
      num = Long.parseLong(in.readLine());
      in.close();
      return num;
   }

   private static boolean isLocalIP(String ip)
   {
      if(ip.startsWith("130.79.129") || ip.startsWith("130.79.128"))
          return true;
      return false;
   }
}
```