Accueil de Samuel Pacault

Samuel Pacault

Exported on 06/22/2018

Table of Contents

1 A mon propos (Samuel Pacault)

Vous pouvez modifier cette page pour y inclure des informations supplémentaires vous concernant.

E-mail: samuel.pacault@keosys.com

Récemment mis à jour

Les formats logs (see page 25)
il y a 2 minutes • mis à jour par Samuel Pacault (see page 0) • afficher les modifications[1]

PATTERNS LOGSTASH (see page 17)
il y a 8 minutes • mis à jour par Samuel Pacault (see page 0) • afficher les modifications[2]

FILEBEAT (see page 23)
il y a 39 minutes • mis à jour par Samuel Pacault (see page 0) • afficher les modifications[3]

FTP filter (see page 12)
il y a environ 6 heures • mis à jour par Samuel Pacault (see page 0) • afficher les modifications[4]

ear filter (see page 6)
hier, à 5:49 PM • mis à jour par Samuel Pacault (see page 0) • afficher les modifications[5]

export/mail/audit filters (see page 7)
hier, à 5:48 PM • mis à jour par Samuel Pacault (see page 0) • afficher les modifications[6]

Import_data filter (see page 14)
hier, à 5:18 PM • mis à jour par Samuel Pacault (see page 0) • afficher les modifications[7]

Import patient filter (see page 16)
juin 19, 2018 • mis à jour par Samuel Pacault (see page 0) • afficher les modifications[8]

FILEBEAT[9]
juin 18, 2018 • commenté par Matthieu Guibert[10]

FILTRES LOGSTASH (see page 6)
juin 12, 2018 • mis à jour par Samuel Pacault (see page 0) • afficher les modifications[11]

PATTERNS LOGSTASH (see page 17)

---

1 http://confluence.keosys.local/pages/diffpagesbyversion.action?pageId=31359681&selectedPageVersions=28&selectedPageVersions=29
2 http://confluence.keosys.local/pages/diffpagesbyversion.action?pageId=31359159&selectedPageVersions=42&selectedPageVersions=43
3 http://confluence.keosys.local/pages/diffpagesbyversion.action?pageId=31359571&selectedPageVersions=1&selectedPageVersions=2
4 http://confluence.keosys.local/pages/diffpagesbyversion.action?pageId=31359272&selectedPageVersions=16&selectedPageVersions=17
5 http://confluence.keosys.local/pages/diffpagesbyversion.action?pageId=31359569&selectedPageVersions=10&selectedPageVersions=9
6 http://confluence.keosys.local/pages/diffpagesbyversion.action?pageId=31359483&selectedPageVersions=12&selectedPageVersions=13
7 http://confluence.keosys.local/pages/diffpagesbyversion.action?pageId=31359665&selectedPageVersions=7&selectedPageVersions=8
8 http://confluence.keosys.local/pages/diffpagesbyversion.action?pageId=31359176&selectedPageVersions=29&selectedPageVersions=30
9 http://confluence.keosys.local/display/~spa/FILEBEAT?focusedCommentId=31359735#comment-31359735
10 http://confluence.keosys.local/display/~mgu
11 http://confluence.keosys.local/pages/diffpagesbyversion.action?pageId=31359174&selectedPageVersions=1&selectedPageVersions=2

juin 08, 2018 • mis à jour par Matthieu Guibert[12] • afficher les modifications[13]

ELK (see page 6)
juin 07, 2018 • créé par Samuel Pacault (see page 0)

Ma première page (see page 29)
juin 07, 2018 • mis à jour par Samuel Pacault (see page 0) • afficher les modifications[14]

Samuel Pacault[15]
juin 07, 2018 • mis à jour par Samuel Pacault (see page 0)

Accueil de Samuel Pacault (see page 0)
juin 07, 2018 • créé par Samuel Pacault (see page 0)

12 http://confluence.keosys.local/display/~mgu
13 http://confluence.keosys.local/pages/diffpagesbyversion.action?pageId=31359159&selectedPageVersions=19&selectedPageVersions=20
14 http://confluence.keosys.local/pages/diffpagesbyversion.action?pageId=31359129&selectedPageVersions=3&selectedPageVersions=4
15 http://confluence.keosys.local/spaces/viewspace.action?key=%7Espa

2 Naviguer dans l'espace

Rechercher

3 ELK

3.1 FILTRES LOGSTASH

3.1.1 ear filter

ear_filter

```
filter {
  if [fields][type] == "ear" {

    grok {
      # Si match avec "source"(chemin du fichier log), créé le champ : "date et platform" et applique les
fonctions suivantes appartenant au plugin grok.
      match => { "source" => [ "/.*/%{DATA:platform}/.*\.%{DATE_SRC_EAR:date}" ] }
    }

    grok {
      # Si match avec "message", créé le champ : "time" et applique les fonctions suivantes appartenant au
plugin grok.
      match => { "message" => [ "^%{TIME:time}" ] }

      # Ajoute un champs "logtimestamp" qui a la valeur des champs "dateeu" et "time" créé
précédemment.    ! Ne pas oublier l'espace !
      add_field => { "logtimestamp" => "%{date} %{time}" }
    }

    # Plugin permettant de gérer les dates. (Entre dans le plugin si match dans grok)
    date {
      # Si "logtimestamp" match avec le bon format date, applique les fonctions suivantes appartenant au
plugin "date".(Pour voir les différents formats dates, voir la page "pattern Logstash".)
      match => [ "logtimestamp", "YYYY-MM-dd HH:mm:ss,SSS" ]

      # Envoie "logtimestamp" dans le paramètre "@timestamp" de kibana, pour que la ligne en question ai
une date correspondant a sa date de création.
      target => "@timestamp"
      locale => "en"
      timezone => "Europe/Paris"
    }


    grok {
      # Si match avec "message", créé les champs : "log-level, class, thread, app, protocole, name,
source2, action, username, patient, visit et payload " et applique les fonctions suivantes appartenant au
plugin grok.
      match => { "message" => [ "^%{TIME}%{SPACE}%{LOGLEVEL:log-level}%{SPACE}\[%{DATA:class}\]%{SPACE}\(%
{DATA:thread}\)%{SPACE}\[%{APP:app}\]%{SPACE}\[TrailService\]%{SPACE}\[%{PNS}\]%{SPACE}%{ACTION2}%{SPACE}(\
[%{PATIENTVISIT}\]|\[%{USERNAME}\])%{SPACE}%{GREEDYDATA:payload}" ] }
```

```
      # Ajoute un tag "TrailService"
      add_tag => [ "TrailService" ]
    }

    # Si match pas avec le premier filtre il reçoit le tag "_grokparsefailure" et entre dans cette
condition.
    if "_grokparsefailure" in [tags] {

      grok {
        # Si match avec "message", créé les champs : "log-level, class, thread, app, protocole, patient,
visit, action et payload " et applique les fonctions suivantes appartenant au plugin grok.
        match => { "message" => [ "^%{TIME}%{SPACE}%{LOGLEVEL:log-level}%{SPACE}\[%{DATA:class}\]%{SPACE}\
(%{DATA:thread}\)%{SPACE}\[%{APP:app}\]%{SPACE}\[%{PROTOCOL}\]%{SPACE}\[%{PATIENTVISIT}\]%{SPACE}\[%
{DATA:action}\]%{GREEDYDATA:payload}" ] }

        # Ajoute le tag "format1".
        add_tag => [ "format1" ]

        # Supprime les tags "_grokparsefailure" car message match avec le bon format.
        remove_tag => [ "_grokparsefailure" ]
      }
    }
    # Si match pas avec le premier et deuxième filtre, il contient toujours le tags "_grokparsefailure" et
entre dans cette condition.
    if "_grokparsefailure" in [tags] {

      grok {
        # Si match avec "message", créé les champs : "log-level, class, thread, app(si existe) et payload "
et applique les fonctions suivantes appartenant au plugin grok.
        match => { "message" => [ "^%{TIME}%{SPACE}%{LOGLEVEL:log-level}%{SPACE}\[%{DATA:class}\]%{SPACE}\
(%{DATA:thread}\)%{SPACE}(\[%{APP:app}\])?%{SPACE}%{GREEDYDATA:payload}" ] }

        # Ajoute le tag "format2".
        add_tag => [ "format2" ]

        # Supprime les tags "_grokparsefailure" car message match avec le bon format.
        remove_tag => [ "_grokparsefailure" ]
      }
    }

    mutate {
      # Supprime les champs "date", "time" et "logtimestamp".
      remove_field => [ "logtimestamp", "date", "time"  ]
      remove_tag => [ "beats_input_codec_plain_applied" ]
    }
  }
}
```

3.1.2 export/mail/audit filters

Les fichiers logs "export_data, mail_app, audit_app" sont constitué de la même manière, c'est pourquoi je les réunis.

export_filter

```
filter {
  if [fields][type] == "export" {

    grok {
      # Si match avec "source"(chemin du fichier log), créé le champ : "platform".
      match => { "source" => [ "/.*/%{DATA:platform}/" ] }
    }

    # Copie "message"(une ligne du fichier log) dans un nouveau champ "message1" pour pouvoir travailler
dessus sans impacter le message original.
    mutate {
      copy => {"message" => "message1"}
    }

    # Supprime les debuts de lignes de wrapper de "message1" SI PRESENTE !
    mutate {
      gsub => ["message1", "^.*\|.*\|.*\| ", ""]
    }

    grok {
      # Si match avec "message"(une ligne), créé les champs : "dateeu, time, log-level et payload" et
applique les fonctions suivantes appartenant au plugin grok.
      match => [ "message1", "%{DATE_EU:dateeu}%{SPACE}%{TIME:time}%{SPACE}%{LOGLEVEL:log-level}%{SPACE}-%
{SPACE}%{GREEDYDATA:payload}" ]

      # Ajoute un champs "logtimestamp" qui a la valeur des champs "dateeu" et "time" créé
précédemment.    ! Ne pas oublier l'espace
      add_field => { "logtimestamp" => "%{dateeu} %{time}" }
    }

    # Plugin permettant de gérer les dates. (Entre dans le plugin si match dans grok)
    date {

        # Si "logtimestamp" match avec le bon format date, applique les fonctions suivantes appartenant au
plugin "date". (Pour voir les différents formats dates, voir la page "pattern Logstash".)
        match => [ "logtimestamp", "dd/MM/YYYY HH:mm:ss,SSS" ]

        # Envoie "logtimestamp" dans le paramètre "@timestamp" de kibana, pour que la ligne en question ai
une date correspondant a sa date de création.
        target => "@timestamp"
        locale => "en"
        timezone => "Europe/Paris"
    }

    # Si match pas avec le premier filtre il reçoit le tags "_grokparsefailure" et entre dans cette
condition.
    if "_grokparsefailure" in [tags] {

      grok {
```

```
        # Si match avec "source"(chemin du fichier log), créé le champ : "date et platform" et applique les fonctions suivantes appartenant au plugin grok.
        match => { "source" =>  [ "%{DATE_SRC:date}" ] }
      }

      date {
        # Si "date" match avec le bon format date, applique les fonctions suivantes appartenant au plugin "date".   ! Le "time" n'est pas obligatoire par defaut "00:00:00,000" (Pour voir les différents formats dates, voir la page "pattern Logstash".)
        match => [ "date", "YYYYMMdd" ]

        # Envoie "date" dans le paramètre "@timestamp" de kibana, pour que la ligne en question ai une date correspondant a sa date de création.
        target => "@timestamp"
        locale => "en"
        timezone => "Europe/Paris"
      }
    }

    mutate {
        # Supprime les champs inutile "date", "dateeu", "time", "message1" et "logtimestamp".
        remove_field => [ "logtimestamp", "dateeu", "time", "date", "message1" ]
        remove_tag => [ "beats_input_codec_plain_applied" ]
    }
  }
}
```

mail_filter

```
filter {
  if [fields][type] == "mail" {

    grok {
      # Si match avec "source"(chemin du fichier log), créé le champ : "platform".
      match => { "source" => [ "/.*/%{DATA:platform}/" ] }
    }

    # Copie "message"(une ligne du fichier log) dans un nouveau champ "message1" pour pouvoir travailler dessus sans impacter le message original.
    mutate {
      copy => {"message" => "message1"}
    }

    # Supprime les debuts de lignes de wrapper de "message1" SI PRESENTE !
    mutate {
      gsub => ["message1", "^.*\|.*\|.*\| ", ""]
    }

    grok {
      # Si match avec "message"(une ligne), créé les champs : "dateeu, time, log-level et payload" et applique les fonctions suivantes appartenant au plugin grok.
      match => [ "message1", "%{DATE_EU:dateeu}%{SPACE}%{TIME:time}%{SPACE}%{LOGLEVEL:log-level}%{SPACE}-%{SPACE}%{GREEDYDATA:payload}" ]
```

```
      # Ajoute un champs "logtimestamp" qui a la valeur des champs "dateeu" et "time" créé précédemment.    ! Ne pas oublier l'espace
      add_field => { "logtimestamp" => "%{dateeu} %{time}" }
    }

    # Plugin permettant de gérer les dates. (Entre dans le plugin si match dans grok)
    date {

        # Si "logtimestamp" match avec le bon format date, applique les fonctions suivantes appartenant au plugin "date". (Pour voir les différents formats dates, voir la page "pattern Logstash".)
        match => [ "logtimestamp", "dd/MM/YYYY HH:mm:ss,SSS" ]

        # Envoie "logtimestamp" dans le paramètre "@timestamp" de kibana, pour que la ligne en question ai une date correspondant a sa date de création.
        target => "@timestamp"
        locale => "en"
        timezone => "Europe/Paris"
    }

    # Si match pas avec le premier filtre il reçoit le tags "_grokparsefailure" et entre dans cette condition.
    if "_grokparsefailure" in [tags] {

      grok {
        # Si match avec "source"(chemin du fichier log), créé le champ : "date et platform" et applique les fonctions suivantes appartenant au plugin grok.
        match => { "source" =>  [ "%{DATE_SRC:date}" ] }
      }

      date {
        # Si "date" match avec le bon format date, applique les fonctions suivantes appartenant au plugin "date".   ! Le "time" n'est pas obligatoire par defaut "00:00:00,000" (Pour voir les différents formats dates, voir la page "pattern Logstash".)
        match => [ "date", "YYYYMMdd" ]

        # Envoie "date" dans le paramètre "@timestamp" de kibana, pour que la ligne en question ai une date correspondant a sa date de création.
        target => "@timestamp"
        locale => "en"
        timezone => "Europe/Paris"
      }
    }

    mutate {
        # Supprime les champs inutile "date", "dateeu", "time", "message1" et "logtimestamp".
        remove_field => [ "logtimestamp", "dateeu", "time", "date", "message1"]
        remove_tag => [ "beats_input_codec_plain_applied" ]
    }
  }
}
```

audit_filter

```
filter {
  if [fields][type] == "audit" {

    grok {
      # Si match avec "source"(chemin du fichier log), créé le champ : "platform".
      match => { "source" => [ "/.*/%{DATA:platform}/" ] }
    }

    # Copie "message"(une ligne du fichier log) dans un nouveau champ "message1" pour pouvoir travailler dessus sans impacter le message original.
    mutate {
      copy => {"message" => "message1"}
    }

    # Supprime les debuts de lignes de wrapper de "message1" SI PRESENTE !
    mutate {
      gsub => ["message1", "^.*\|.*\|.*\| ", ""]
    }

    grok {
      # Si match avec "message"(une ligne), créé les champs : "dateeu, time, log-level et payload" et applique les fonctions suivantes appartenant au plugin grok.
      match => [ "message1", "%{DATE_EU:dateeu}%{SPACE}%{TIME:time}%{SPACE}%{LOGLEVEL:log-level}%{SPACE}-%{SPACE}%{GREEDYDATA:payload}" ]

      # Ajoute un champs "logtimestamp" qui a la valeur des champs "dateeu" et "time" créé précédemment.    ! Ne pas oublier l'espace
      add_field => { "logtimestamp" => "%{dateeu} %{time}" }
    }

    # Plugin permettant de gérer les dates. (Entre dans le plugin si match dans grok)
    date {

        # Si "logtimestamp" match avec le bon format date, applique les fonctions suivantes appartenant au plugin "date". (Pour voir les différents formats dates, voir la page "pattern Logstash".)
        match => [ "logtimestamp", "dd/MM/YYYY HH:mm:ss,SSS" ]

        # Envoie "logtimestamp" dans le paramètre "@timestamp" de kibana, pour que la ligne en question ai une date correspondant a sa date de création.
        target => "@timestamp"
        locale => "en"
        timezone => "Europe/Paris"
    }

    # Si match pas avec le premier filtre il reçoit le tags "_grokparsefailure" et entre dans cette condition.
    if "_grokparsefailure" in [tags] {

      grok {
        # Si match avec "source"(chemin du fichier log), créé le champ : "date et platform" et applique les fonctions suivantes appartenant au plugin grok.
        match => { "source" =>  [ "%{DATE_SRC:date}" ] }
```

```
      }

      date {
        # Si "date" match avec le bon format date, applique les fonctions suivantes appartenant au plugin
"date".   ! Le "time" n'est pas obligatoire par defaut "00:00:00,000" (Pour voir les différents formats
dates, voir la page "pattern Logstash".)
        match => [ "date", "YYYYMMdd" ]

        # Envoie "date" dans le paramètre "@timestamp" de kibana, pour que la ligne en question ai une date
correspondant a sa date de création.
        target => "@timestamp"
        locale => "en"
        timezone => "Europe/Paris"
      }
    }

    mutate {
        # Supprime les champs inutile "date", "dateeu", "time", "message1" et "logtimestamp".
        remove_field => [ "logtimestamp", "dateeu", "time", "date", "message1" ]
        remove_tag => [ "beats_input_codec_plain_applied" ]
    }
  }
}
```

Voir les patterns Logstash (see page 17)

3.1.3 FTP filter

ftp__filter

```
filter {
  if [fields][type] == "ftp" {

    grok {
      # Si match avec "source"(chemin du fichier log), créé le champ : "platform".
      match => { "source" => [ "/.*/%{DATA:platform}/" ] }
    }

    # Copie "message"(une ligne du fichier log) dans un nouveau champ "message1" SI PRESENTE !
    mutate {
      copy => {"message" => "message1"}
    }

    # Supprime les debuts de lignes de wrapper de "message1".
    mutate {
      gsub => ["message1", "^.*\|.*\|.*\| ", ""]
    }

    grok {
      # Si match avec "message1" (copie d'une ligne), créé les champs : "log-level, date et payload" et
applique les fonctions suivantes appartenant au plugin grok.
```

```
      match => [ "message1", "%{DATETIME_US:datetime}%{SPACE}%{GREEDYDATA:class}%{LOGLEVEL:log-level}:%
{SPACE}%{GREEDYDATA:payload}" ]
    }

    # Plugin permettant de gérer les dates. (Entre dans le plugin si match dans grok)
    date {

      # Si "datetime" match avec le bon format date, applique les fonctions suivantes appartenant au plugin
"date".(Pour voir les différents formats dates, voir la page "pattern Logstash".)
      match => [ "datetime", "MMM dd, YYYY hh:mm:ss aa" ]

      # Envoie "datetime" dans le paramètre @timestamp de kibana, pour que la ligne en question ai une date
correspondant a sa date de création.
      target => "@timestamp"
      locale => "en"
      timezone => "Europe/Paris"
    }

    # Si match pas avec le premier filtre il reçoit le tags "_grokparsefailure" et entre dans cette
condition.
    if "_grokparsefailure" in [tags] {

      grok {
        # Si match avec "message" (une ligne), créé les champs : "datetime, log-level, class et payload" et
applique les fonctions suivantes appartenant au plugin grok.
        match => [ "message", "%{TIMESTAMP_ISO8601:datetime}%{SPACE}%{LOGLEVEL:log-level}%{SPACE}\[%
{GREEDYDATA:class}\]%{SPACE}%{GREEDYDATA:payload}" ]

        # Ajoute le tag "format20180208".
        add_tag => [ "format20180208" ]
        # Supprime les tags "_grokparsefailure", "_dateparsefailure" car message match avec le bon format.
        remove_tag => [ "_grokparsefailure", "_dateparsefailure" ]
      }

      # Plugin permettant de gérer les dates. (Entre dans le plugin si match dans grok)
      date {

        # Si "datetime" match avec le bon format date, applique les fonctions suivantes appartenant au
plugin "date".(Pour voir les différents formats dates, voir la page "pattern Logstash".)
        match => [ "datetime", "YYYY-MM-dd HH:mm:ss.SSS" ]

        # Envoie "datetime" dans le paramètre @timestamp de kibana, pour que la ligne en question ai une
date correspondant a sa date de création.
        target => "@timestamp"
        locale => "en"
        timezone => "Europe/Paris"
      }
    }

    # Si match pas avec le premier et deuxième filtre, il contient toujours le tags "_grokparsefailure" et
entre dans cette condition.
    if "_grokparsefailure" in [tags] {

      grok {
```

```
        # Si match avec "source"(chemin du fichier log), créé le champ : "date" et applique les fonctions
suivantes appartenant au plugin grok.
        match => { "source" => [ "%{DATE_SRC:date}" ] }
      }

      # Plugin permettant de gérer les dates. (Entre dans le plugin si match dans grok)
      date {

        # Si "date" match avec le format date, applique les fonctions suivantes appartenant au plugin
"date".(Pour voir les différents formats dates, voir la page "pattern Logstash".)
        match => [ "date", "YYYYMMdd" ]

        # Envoie "date" dans le parametre @timestamp de kibana, pour que la ligne en question ai une date
correspondant a sa date de création.
        target => "@timestamp"
        locale => "en"
        timezone => "Europe/Paris"
      }
    }

    mutate {
      # Supprime les champs inutile "datetime", "date" et "message1".
      remove_field => [ "date", "datetime", "message1" ]
      remove_tag => [ "beats_input_codec_plain_applied" ]
    }
  }
}
```

3.1.4 Import_data filter

import_data_filter

```
filter {
  if [fields][type] == "import_data" {

    grok {
      # Si match avec "source"(chemin du fichier log), créé le champ : "platform".
      match => { "source" => [ "/.*/%{DATA:platform}/" ] }
    }

    # Copie "message"(une ligne du fichier log) dans un nouveau champ "message1" pour pouvoir travailler
dessus sans impacter le message original.
    mutate {
      copy => {"message" => "message1"}
    }

    # Supprime les debuts de lignes de wrapper de "message1" SI PRESENTE !
    mutate {
      gsub => ["message1", "^.*\|.*\|.*\| ", ""]
    }
```

```
    grok {
      # Si match avec "message1" (copie d'une ligne du fichier log), créé les champs : "dateeu, time, log-level et payload" et applique les fonctions suivantes appartenant au plugin grok.
      match => [ "message1", "%{DATE_EU:dateeu}%{SPACE}%{TIME:time}%{SPACE}%{LOGLEVEL:log-level}%{SPACE}-%{SPACE}%{GREEDYDATA:payload}" ]

      # Ajoute un champs "logtimestamp" qui a la valeur des champs "dateeu" et "time" créé précédemment.    ! Ne pas oublier l'espace !
      add_field => { "logtimestamp" => "%{dateeu} %{time}" }
    }

    # Plugin permettant de gérer les dates. (Entre dans le plugin si match dans grok)
    date {
        # Si "logtimestamp" match avec le bon format date, applique les fonctions suivantes appartenant au plugin "date". (Pour voir les différents formats dates, voir la page "pattern Logstash".)
        match => [ "logtimestamp", "dd/MM/YYYY HH:mm:ss,SSS" ]

        # Envoie "logtimestamp" dans le paramètre "@timestamp" de kibana, pour que la ligne en question ai une date correspondant a sa date de création.
        target => "@timestamp"
        locale => "en"
        timezone => "Europe/Paris"
    }

    # Si match pas avec le premier, il reçcoit le tags "_grokparsefailure" et entre dans cette condition.
    if "_grokparsefailure" in [tags] {

      grok {
        # Si match avec "source"(chemin du fichier log), créé le champ : "date" et applique les fonctions suivantes appartenant au plugin grok.
        match => { "source" =>  [ "%{DATE_SRC:date}" ] }
      }

      # Plugin permettant de gérer les dates. (Entre dans le plugin si match dans grok)
      date {

        # Si "date" match avec le bon format date, applique les fonctions suivantes appartenant au plugin "date".(Pour voir les différents formats dates, voir page "pattern Logstash".)
        match => [ "date", "YYYYMMdd" ]

        # Envoie "date" dans le paramètre "@timestamp" de kibana, pour que la ligne en question ai une date correspondant a sa date de création.    ! time par defaut "00:00:00,000"
        target => "@timestamp"
        locale => "en"
        timezone => "Europe/Paris"
      }
    }

    mutate {
        # Supprime les champs inutile "dateeu", "time", "date, "message1" et "logtimestamp".
        remove_field => [ "logtimestamp", "dateeu", "time", "date", "message1" ]
        remove_tag => [ "beats_input_codec_plain_applied" ]
```

```
    }

  }

}
```

3.1.5 Import patient filter

import_patient_filter

```
filter {
  if [fields][type] == "import_patient" {

    # Supprime les debuts de lignes de wrapper de "message".
    mutate {
        gsub => ["message", "^.*\|.*\|.*\| ", ""]
    }

    grok {
      # Si match avec "message" (une ligne), créé les champs : "log-level, datetime et payload" et applique
les fonctions suivantes appartenant au plugin grok.
      match => [ "message", "%{DATETIME_US:datetime}%{GREEDYDATA}%{LOGLEVEL:log-level}:%{SPACE}%
{GREEDYDATA:payload}" ]
    }

    # Plugin permettant de gérer les dates. (Entre dans le plugin si match dans grok)
    date {

      # Si "datetime" match avec le bon format date, applique les fonctions suivantes appartenant au plugin
"date". (Pour voir les différents formats dates, voir la page "pattern Logstash".)
      match => [ "datetime", "MMM dd, YYYY hh:mm:ss aa" ]

      # Envoie "datetime" dans le paramètre "@timestamp" de kibana, pour que la ligne en question ai une
date correspondant a sa date de création.
      target => "@timestamp"
      locale => "en"
      timezone => "Europe/Paris"
    }

    # Si match pas avec le premier filtre il reçoit le tags "_grokparsefailure" et entre dans cette
condition.
    if "_grokparsefailure" in [tags] {

      grok {
        # Si match avec "source"(chemin du fichier log), créé le champ : "date" et applique les fonctions
suivantes appartenant au plugin grok.
        match => { "source" => [ "/*/*/*\.%{DATE_SRC:date}" ] }
      }

      date {
```

```
        # Si match avec le format de date, applique les fonctions suivantes appartenant au plugin
"date".   ! Le "time" n'est pas obligatoire par defaut "00:00:00,000" (Pour voir les différents formats
dates, voir la page "pattern Logstash".)
        match => [ "date", "YYYYMMdd" ]

        # Envoie "date" dans le parametre @timestamp de kibana, pour que la ligne en question ai une date
correspondant a sa date de création.
        target => "@timestamp"
        locale => "en"
        timezone => "Europe/Paris"
      }
    }

    mutate {
      # Supprime les champs "datetime", "date" et "logstamp".
      remove_field => [ "logtimestamp", "datetime", "date" ]
    }
  }
}
```

3.2 PATTERNS LOGSTASH

Dates patterns

```
#Date
TIME_US %{TIME} (AM|PM)
DATETIME_US %{MONTH} %{MONTHDAY}, %{YEAR} %{TIME_US}
DATETIME_WP %{YEAR}/%{MONTHNUM2}/%{MONTHDAY} %{TIME}

#Date source
DATE_SRC %{YEAR}%{MONTHNUM2}%{MONTHDAY}
DATE_SRC_EAR %{YEAR}-%{MONTHNUM2}-%{MONTHDAY}
```

Nom pattern	fonction	Exemple Match	Commentaire
DATETIME_WP	Match avec la date utilisé par le wrapper.	2018/01/25 00:06:35	Dans le plugin date, match avec : "YYYY/MM/dd HH:mm:ss"
DATE_SRC	Match avec la date dans le nom du fichier.	20180425	Dans le plugin date, match avec : "YYYYMMdd"
DATE_SRC_EAR	Match avec la date dans le nom du fichier EAR.	2018-04-25	Dans le plugin date, match avec : "YYYY-MM-dd"
DATETIME_US	Match avec la date dans les logs : "import-patient" et "ftp-patient"	Jan 25, 2018 12:06:35 (AM ou PM)	Dans le plugin date, match avec : "MMM dd, YYYY hh:mm:ss aa"

Nom pattern	fonction	Exemple Match	Commentaire
TIME_US	Match avec le temps dans les logs : "import-patient" et "ftp-patient"	12:06:35 (AM ou PM)	Dans le plugin date, match avec : "hh:mm:ss aa"

The pattern letters

```
Symbol  Meaning                      Presentation  Examples
------  -------                      ------------  -------
 G       era                          text          AD
 C       century of era (>=0)         number        20
 Y       year of era (>=0)            year          1996

 x       weekyear                     year          1996
 w       week of weekyear             number        27
 e       day of week                  number        2
 E       day of week                  text          Tuesday; Tue

 y       year                         year          1996
 D       day of year                  number        189
 M       month of year                month         July; Jul; 07
 d       day of month                 number        10

 a       halfday of day               text          PM
 K       hour of halfday (0~11)       number        0
 h       clockhour of halfday (1~12)  number        12

 H       hour of day (0~23)           number        0
 k       clockhour of day (1~24)      number        24
 m       minute of hour               number        30
 s       second of minute             number        55
 S       fraction of second           number        978

 z       time zone                    text          Pacific Standard Time; PST
 Z       time zone offset/id          zone          -0800; -08:00; America/Los_Angeles

 '       escape for text              delimiter
 ''      single quote                 literal       '
```

---

patterns_ear

```
PROTOCOL Protocol|Protocol: '%{DATA:protocol}'
PNS protocol:'%{DATA:protocol}' / user:'%{DATA:name}' / source:'%{DATA:source2}'
APP imagys.*?
PATIENTVISIT Patient: '%{DATA:patient}' / Visit: '%{DATA:visit}'
```

```
USERNAME User:'%{DATA:username}'

#Action
ACTION UPLOAD|READING
ACTION2 action:'\[%{DATA:action}\]
```

Nom	Fonction	Exemple Match	Valeur retour
PROTOCOL	Récupère le protocole	Protocol Protocol: 'blabla'	**protocol:** blabla
PNS protocol:'%{DATA:protocol}' / user:'%{DATA:name}' / source:'%{DATA:source2}'	Récupère le protocole, name et source	protocol:'no_protocol' / user:'unUser' / source:'Web site'	**protocol:** no_protocol **name:** unUser **source2:** Web site
APP	Récupère l'application	imagys-test-ear	**app:** imagys-test-ear
PATIENTVISIT	Récupère patient et visite	Patient: 'blabla' / Visit: 'bloblo'	**patient:** blabla **visit:** bloblo
USERNAME	Récupère l'username	User:'unUsername'	**username:** unUsername
ACTION	Récupère l'action	UPLOAD OU READING	
ACTION2	Récupère l'action	action:'[Usermanagement]	**action:** usermanagement

---

J'utilise pas les patterns wrapper (peuvent etre utiles)

Wrapper pattern

```
#Wrapper

WRAPPER .*\|.*\|.*\|%{SPACE}
WRAPPERD .*\|.*\| %{DATEWP:date} \|
```

Nom	Fonction	Match	Commentaire	Valeur retour
WRAPPER	Match avec les débuts de lignes de wrapper.	ERROR | wrapper | 2018/04/09 11:36:30 | INFO | jvm 1 | 2018/04/09 11:36:31 |	A utiliser pour passé les débuts de lignes wrapper.	
WRAPPERD	Match avec les débuts de lignes de wrapper, retourne un champs de nom "date" qui a pour pattern DATEWP	ERROR | wrapper | 2018/04/09 11:36:30 | INFO | jvm 1 | 2018/04/09 11:36:31 |	A utiliser pour passé les débuts de lignes wrapper et récupérer la date.	Créer un champs par défaut : date : 2018/04/09 11:36:31

Default patterns Logstash

```
USERNAME [a-zA-Z0-9._-]+
USER %{USERNAME}
INT (?:[+-]?(?:[0-9]+))
BASE10NUM (?<![0-9.+-])(?>[+-]?(?:(?:[0-9]+(?:\.[0-9]+)?)|(?:\.[0-9]+)))
NUMBER (?:%{BASE10NUM})
BASE16NUM (?<![0-9A-Fa-f])(?:[+-]?(?:0x)?(?:[0-9A-Fa-f]+))
BASE16FLOAT \b(?<![0-9A-Fa-f.])(?:[+-]?(?:0x)?(?:(?:[0-9A-Fa-f]+(?:\.[0-9A-Fa-f]*)?)|(?:\.[0-9A-Fa-f]+)))\b

POSINT \b(?:[1-9][0-9]*)\b
NONNEGINT \b(?:[0-9]+)\b
WORD \b\w+\b
NOTSPACE \S+
SPACE \s*
DATA .*?
GREEDYDATA .*
QUOTEDSTRING (?>(?<!\\)(?>"(?>\\.|[^\\"]+)+"|""|(?>'(?>\\.|[^\\']+)+')|''|(?>`(?>\\.|[^\\`]+)+`)|``))
UUID [A-Fa-f0-9]{8}-(?:[A-Fa-f0-9]{4}-){3}[A-Fa-f0-9]{12}

# Networking
MAC (?:%{CISCOMAC}|%{WINDOWSMAC}|%{COMMONMAC})
CISCOMAC (?:(?:[A-Fa-f0-9]{4}\.){2}[A-Fa-f0-9]{4})
WINDOWSMAC (?:(?:[A-Fa-f0-9]{2}-){5}[A-Fa-f0-9]{2})
```

```
COMMONMAC (?:(?:[A-Fa-f0-9]{2}:){5}[A-Fa-f0-9]{2})
IPV6 ((([0-9A-Fa-f]{1,4}:){7}([0-9A-Fa-f]{1,4}|:))|(([0-9A-Fa-f]{1,4}:){6}(:[0-9A-Fa-f]{1,4}|((25[0-5]|2[0
-4]\d|1\d\d|[1-9]?\d)(\.(25[0-5]|2[0-4]\d|1\d\d|[1-9]?\d)){3})|:))|(([0-9A-Fa-f]{1,4}:){5}(((:[0-9A-Fa-f]
{1,4}){1,2})|:((25[0-5]|2[0-4]\d|1\d\d|[1-9]?\d)(\.(25[0-5]|2[0-4]\d|1\d\d|[1-9]?\d)){3})|:))|(([0-9A-Fa-f]
{1,4}:){4}(((:[0-9A-Fa-f]{1,4}){1,4})|((:[0-9A-Fa-f]{1,4})?:((25[0-5]|2[0-4]\d|1\d\d|[1-9]?\d)(\.(25[0-5]|
2[0-4]\d|1\d\d|[1-9]?\d)){3}))|:))|(([0-9A-Fa-f]{1,4}:){3}(((:[0-9A-Fa-f]{1,4}){1,4})|((:[0-9A-Fa-f]{1,4})
{0,2}:((25[0-5]|2[0-4]\d|1\d\d|[1-9]?\d)(\.(25[0-5]|2[0-4]\d|1\d\d|[1-9]?\d)){3}))|:))|(([0-9A-Fa-f]{1,4}:)
{2}(((:[0-9A-Fa-f]{1,4}){1,4})|((:[0-9A-Fa-f]{1,4}){0,3}:((25[0-5]|2[0-4]\d|1\d\d|[1-9]?\d)(\.(25[0-5]|2[0
-4]\d|1\d\d|[1-9]?\d)){3}))|:))|(([0-9A-Fa-f]{1,4}:){1}(((:[0-9A-Fa-f]{1,4}){1,6})|((:[0-9A-Fa-f]{1,4}){0,
4}:((25[0-5]|2[0-4]\d|1\d\d|[1-9]?\d)(\.(25[0-5]|2[0-4]\d|1\d\d|[1-9]?\d)){3}))|:))|(:(((:[0-9A-Fa-f]{1,4})
{1,7})|((:[0-9A-Fa-f]{1,4}){0,5}:((25[0-5]|2[0-4]\d|1\d\d|[1-9]?\d)(\.(25[0-5]|2[0-4]\d|1\d\d|[1-9]?\d))
{3}))|:)))(%.+)?
IPV4 (?<![0-9])(?:(?:25[0-5]|2[0-4][0-9]|[0-1]?[0-9]{1,2})[.](?:25[0-5]|2[0-4][0-9]|[0-1]?[0-9]{1,2})[.](?:
25[0-5]|2[0-4][0-9]|[0-1]?[0-9]{1,2})[.](?:25[0-5]|2[0-4][0-9]|[0-1]?[0-9]{1,2}))(?![0-9])
IP (?:%{IPV6}|%{IPV4})
HOSTNAME \b(?:[0-9A-Za-z][0-9A-Za-z-]{0,62})(?:\.(?:[0-9A-Za-z][0-9A-Za-z-]{0,62}))*(\.?|\b)
HOST %{HOSTNAME}
IPORHOST (?:%{HOSTNAME}|%{IP})
HOSTPORT %{IPORHOST}:%{POSINT}

# paths
PATH (?:%{UNIXPATH}|%{WINPATH})
UNIXPATH (?>/(?>[\w_%!$@:.,-]+|\\.)*)+
TTY (?:/dev/(pts|tty([pq])?)(\w+)?/?(?:[0-9]+))
WINPATH (?>[A-Za-z]+:|\\)(?:\\[^\\?*]*)+
URIPROTO [A-Za-z]+(\+[A-Za-z+]+)?
URIHOST %{IPORHOST}(?::%{POSINT:port})?
# uripath comes loosely from RFC1738, but mostly from what Firefox
# doesn't turn into %XX
URIPATH (?:/[A-Za-z0-9$.+!*'(){},~:;=@#%_\-]*)+
#URIPARAM \?(?:[A-Za-z0-9]+(?:=(?:[^&]*))?(?:&(?:[A-Za-z0-9]+(?:=(?:[^&]*))?)?)*)?
URIPARAM \?[A-Za-z0-9$.+!*'|(){},~@#%&/=:;_?\-\[\]]*
URIPATHPARAM %{URIPATH}(?:%{URIPARAM})?
URI %{URIPROTO}://(?:%{USER}(?::[^@]*)?@)?(?:%{URIHOST})?(?:%{URIPATHPARAM})?

# Months: January, Feb, 3, 03, 12, December
MONTH \b(?:Jan(?:uary)?|Feb(?:ruary)?|Mar(?:ch)?|Apr(?:il)?|May|Jun(?:e)?|Jul(?:y)?|Aug(?:ust)?|
Sep(?:tember)?|Oct(?:ober)?|Nov(?:ember)?|Dec(?:ember)?)\b
MONTHNUM (?:0?[1-9]|1[0-2])
MONTHNUM2 (?:0[1-9]|1[0-2])
MONTHDAY (?:(?:0[1-9])|(?:[12][0-9])|(?:3[01])|[1-9])

# Days: Monday, Tue, Thu, etc...
DAY (?:Mon(?:day)?|Tue(?:sday)?|Wed(?:nesday)?|Thu(?:rsday)?|Fri(?:day)?|Sat(?:urday)?|Sun(?:day)?)

# Years?
YEAR (?>\d\d){1,2}
HOUR (?:2[0123]|[01]?[0-9])
MINUTE (?:[0-5][0-9])
# '60' is a leap second in most time standards and thus is valid.
SECOND (?:(?:[0-5]?[0-9]|60)(?:[:.,][0-9]+)?)
TIME (?!<[0-9])%{HOUR}:%{MINUTE}(?::%{SECOND})(?![0-9])
# datestamp is YYYY/MM/DD-HH:MM:SS.UUUU (or something like it)
DATE_US %{MONTHNUM}[/-]%{MONTHDAY}[/-]%{YEAR}
```

```
DATE_EU %{MONTHDAY}[./-]%{MONTHNUM}[./-]%{YEAR}
ISO8601_TIMEZONE (?:Z|[+-]%{HOUR}(?::?%{MINUTE}))
ISO8601_SECOND (?:%{SECOND}|60)
TIMESTAMP_ISO8601 %{YEAR}-%{MONTHNUM}-%{MONTHDAY}[T ]%{HOUR}:?%{MINUTE}(?::?%{SECOND})?%{ISO8601_TIMEZONE}?
DATE %{DATE_US}|%{DATE_EU}
DATESTAMP %{DATE}[- ]%{TIME}
TZ (?:[PMCE][SD]T|UTC)
DATESTAMP_RFC822 %{DAY} %{MONTH} %{MONTHDAY} %{YEAR} %{TIME} %{TZ}
DATESTAMP_RFC2822 %{DAY}, %{MONTHDAY} %{MONTH} %{YEAR} %{TIME} %{ISO8601_TIMEZONE}
DATESTAMP_OTHER %{DAY} %{MONTH} %{MONTHDAY} %{TIME} %{TZ} %{YEAR}
DATESTAMP_EVENTLOG %{YEAR}%{MONTHNUM2}%{MONTHDAY}%{HOUR}%{MINUTE}%{SECOND}

# Syslog Dates: Month Day HH:MM:SS
SYSLOGTIMESTAMP %{MONTH} +%{MONTHDAY} %{TIME}
PROG (?:[\w._/%-]+)
SYSLOGPROG %{PROG:program}(?:\[%{POSINT:pid}\])?
SYSLOGHOST %{IPORHOST}
SYSLOGFACILITY <%{NONNEGINT:facility}.%{NONNEGINT:priority}>
HTTPDATE %{MONTHDAY}/%{MONTH}/%{YEAR}:%{TIME} %{INT}

# Shortcuts
QS %{QUOTEDSTRING}

# Log formats
SYSLOGBASE %{SYSLOGTIMESTAMP:timestamp} (?:%{SYSLOGFACILITY} )?%{SYSLOGHOST:logsource} %{SYSLOGPROG}:
COMMONAPACHELOG %{IPORHOST:clientip} %{USER:ident} %{USER:auth} \[%{HTTPDATE:timestamp}\] "(?:%{WORD:verb}
%{NOTSPACE:request}(?: HTTP/%{NUMBER:httpversion})?|%{DATA:rawrequest})" %{NUMBER:response} (?:%
{NUMBER:bytes}|-)
COMBINEDAPACHELOG %{COMMONAPACHELOG} %{QS:referrer} %{QS:agent}

# Log Levels
LOGLEVEL ([Aa]lert|ALERT|[Tt]race|TRACE|[Dd]ebug|DEBUG|[Nn]otice|NOTICE|[Ii]nfo|INFO|[Ww]arn?(?:ing)?|WARN?
(?:ING)?|[Ee]rr?(?:or)?|ERR?(?:OR)?|[Cc]rit?(?:ical)?|CRIT?(?:ICAL)?|[Ff]atal|FATAL|[Ss]evere|SEVERE|
EMERG(?:ENCY)?|[Ee]merg(?:ency)?)
```

NOTE :

Différence entre %{GREEDYDATA} et %{DATA}

`GREEDYDATA .*` Il essai de faire correspondre autant de répétitions que possible

`DATA .*?` Il essai de faire correspondre le moins de répétitions possible

Exemple :

`Entrer : 101000000000100`

Pattern	Match
1.*1	1010000000001
1.*?1	101

Chemin petterns personnaliser

/usr/share/logstash/vendor/bundle/jruby/2.3.0/gems/logstash-patterns-core-4.1.2/patterns

3.3 FILEBEAT

filebeat.yml

```
filebeat:
  prospectors:
    - paths:
      - /log/*/*import_data*.log
      close_inactive: 20s
      fields: {type: import_data}
      multiline:
        pattern: '^.{42}([[:space:]](at|\.{3})|Caused by:|com.|org.|The last|\[Fatal Error\]|java.|
Exception in thread|$)'
        negate: false
        match: after

    - paths:
      - /log/*/*export*.log
      close_inactive: 20s
      fields: {type: export}
      multiline:
        pattern: '^.{42}([[:space:]](at|\.{3})|Caused by:|com.|org.|The last|\[Fatal Error\]|java.|
Exception in thread|$)'
        negate: false
        match: after

    - paths:
      - /log/*/*ear*.log.*
      close_inactive: 20s
      fields: {type: ear}
      multiline:
        pattern: '^\d{2}:\d{2}:\d{2},\d{3}'
        negate: true
        match: after

    - paths:
```

```
      - /log/*/*mail*.log
      close_inactive: 20s
      fields: {type: mail}
      multiline:
        pattern: '^.{42}([[:space:]]*(at|java.|nested|\.{3})|Caused by:|com.|org.|The last|\[Fatal Error\]|
java.|javax.|$)'
        negate: false
        match: after

    - paths:
      - /log/*/*audit*.log
      close_inactive: 20s
      fields: {type: audit}
      multiline:
        pattern: '^.{42}([[:space:]]*(at|java.|nested|\.{3})|Caused by:|com.|org.|The last|\[Fatal Error\]|
java.|javax.|$)'
        negate: false
        match: after

    - paths:
      - /log/*/*import-patient*.log
      close_inactive: 20s
      fields: {type: import_patient}
      multiline:
        pattern: '^.{42}(INFO|WARN|SEVERE|[[:space:]](at|\.{3})|Caused by|com.|java.|Exception in|The last
packet|$)'
        negate: false
        match: after

    - paths:
      - /log/*/*ftp*.log
      close_inactive: 20s
      fields: {type: ftp}
      multiline:
        pattern: '^.{42}(INFO|WARN|SEVERE|[[:space:]](at|\.{3})|Caused by|com.|java.|Exception in|The last
packet|$)|^([[:space:]](at|\.{3})|Caused by|com.|java.|Exception in|The last packet|$)'
        negate: false
        match: after

    - paths:
      - /viewer_logs/*.log
      close_inactive: 20s
      fields: {type: viewer}
      multiline:
        pattern: '^\d{4}/\d{2}/\d{2}'
        negate: true
        match: after

output:
  logstash:
    hosts: ["127.0.0.1:5044"]

logging:
  to_syslog: false
```

```
  to_files: true
  files:
    path: /var/log/filebeat
    name: filebeat-plain.log
    rotateeverybytes: 10485760 # 10MB
    level: info
```

3.4 Les formats logs

EXPORT	
Nom du fichier	Imagys_export_data.20180101.log
Format	`INFO   | jvm 1    | 2018/01/01 23:25:08 | 01/01/2018 23:25:08,947 WARN  - PAYLOAD`

MAIL	
Nom du fichier	imagys_mail_app.20180228.log
Format	`INFO   | jvm 1    | 2018/02/28 01:40:05 | 28/02/2018 01:40:05,737 WARN  - PAYLOAD`

AUDIT	
Nom du fichier	imagys_audit_app.20180322.log
Format	`INFO   | jvm 1    | 2018/03/22 01:37:22 | 22/03/2018 01:37:22,695 WARN  - PAYLOAD`

Filtre correspondant

```
%{DATE_EU:dateeu}%{SPACE}%{TIME:time}%{SPACE}%{LOGLEVEL:log-level}%{SPACE}-%{SPACE}%{GREEDYDATA:payload}
```

IMPORT_DATA	
Nom du fichier	imagys_import_data.20180508.log
Format	`INFO   | jvm 1    | 2018/05/08 22:02:10 | 08/05/2018 22:02:10,510 INFO  - PAYLOAD`
Nouveau format sur "test" à partir du 3 mai 2018	`03/05/2018 14:31:58,934 INFO  - PAYLOAD`

Filtre correspondant

```
%{DATE_EU:dateeu}%{SPACE}%{TIME:time}%{SPACE}%{LOGLEVEL:log-level}%{SPACE}-%{SPACE}%{GREEDYDATA:payload}
```

IMPORT_PATIENT	
Nom du fichier	imagys-import-patient.20180302.log
Format (sur 2lignes)	`INFO   | jvm 1    | 2018/03/02 00:09:38 | Mar 02, 2018 12:09:38 AM com.keosys.imagys.processing.ProcessingAgent processImport` `INFO   | jvm 1    | 2018/03/02 00:09:38 | INFO: PAYLOAD`

Filtre correspondant

```
%{CUSTOMDATE1:datetime}%{GREEDYDATA}%{LOGLEVEL:log-level}:%{SPACE}%{GREEDYDATA:payload}
```

FTP	
Nom du fichier	imagys-ftp-patient.20180515.log

FTP	
Format	`INFO   | jvm 1    | 2018/05/15 00:00:04 | May 15, 2018 12:00:04 AM com.keosys.imagys.ftp.Configuration load` `INFO   | jvm 1    | 2018/05/15 00:00:04 | INFO: PAYLOAD`
Nouveau format sur "test" a partir du 8 février 2018	`2018-02-08 10:28:30.603 INFO  [ImportProcessingThread] PAYLOAD`

Filtre correspondant

```
%{CUSTOMDATE1:datetime}%{SPACE}%{GREEDYDATA}%{LOGLEVEL:log-level}:%{SPACE}%{GREEDYDATA:payload}

# Nouveau format
%{TIMESTAMP_ISO8601:datetime}%{SPACE}%{LOGLEVEL:log-level}%{SPACE}\[%{GREEDYDATA:info}\]%{SPACE}%
{GREEDYDATA:payload}
```

EAR	
Nom du fichier	imagys-test-ear.log.2018-04-04

Format "TrailService"	`10:00:42,749 INFO  [com.keosys.apps.sos.util.TrailUtil$TrailStacker] (ajp-/172.16.60.3:8109-15)[imagys-test-ear] [TrailService] [protocol:'no_protocol' / user:'unName' / source:'Web site'] action:'[Usermanagement] [User:'unUsername'] PAYLOAD`
	OU
	`02:34:27,380 INFO  [com.keosys.apps.sos.util.TrailUtil$TrailStacker] (ajp-/172.16.60.2:8059-22)[imagys-test-ear] [TrailService] [protocol:'unProtocol' / user:'unName' / source:'Web site'] action:'[reading] [Patient: 'unPatient' / Visit: 'uneVisite'] PAYLOAD`

EAR	
Format 1	`02:49:11,618 INFO  [com.keosys.apps.sos.metier.BOAction] (ajp-/172.16.60.3:8109-129) [imagys-test-ear] [Protocol] [Patient: 'unPatient' / Visit: 'uneVisite'][READING] PAYLOAD`

Format 2	`02:14:45,425 WARN  [com.keosys.apps.webservice.business.plugin.impl.BOKVisitPlugin] (ajp-/172.16.60.3:8109-114) [imagys-test-ear] PAYLOAD`
	OU
	`02:48:43,070 WARN  [com.keosys.apps.sos.metier.BOVisitReview] (ajp-/172.16.60.3:8109-44) [imagys-test-ear] PAYLOAD`

Filtre correspondant

```
# TrailService
^%{TIME}%{SPACE}%{LOGLEVEL:log-level}%{SPACE}\[%{DATA:class}\]%{SPACE}\(%{DATA:thread}\)%{SPACE}\[%{APP:app}\]%{SPACE}\[TrailService\]%{SPACE}\[%{PNS}\]%{SPACE}%{ACTION2}%{SPACE}(\[%{PATIENTVISIT}\]|\[%{USERNAME}\])%{SPACE}%{GREEDYDATA:payload}

# Format 1
^%{TIME}%{SPACE}%{LOGLEVEL:log-level}%{SPACE}\[%{DATA:class}\]%{SPACE}\(%{DATA:thread}\)%{SPACE}\[%{APP:app}\]%{SPACE}\[%{PROTOCOL}\]%{SPACE}\[%{PATIENTVISIT}\]%{SPACE}\[%{DATA:action}\]%{GREEDYDATA:payload}

# Format 2
^%{TIME}%{SPACE}%{LOGLEVEL:log-level}%{SPACE}\[%{DATA:class}\]%{SPACE}\(%{DATA:thread}\)%{SPACE}(\[%{APP:app}\])?%{SPACE}%{GREEDYDATA:payload}
```

4 Ma première page

HELLO WORLD

```
PROTOCOL \[Protocol\]|\[Protocol: '%{DATA:protocol}'\]
PNS \[protocol:'%{DATA:protocol}' / user:'%{DATA:name}' / source:'%{DATA:source2}'\]
APP \[(?<app>imagys.*?)\]
PATIENTVISIT \[Patient: '%{DATA:patient}' / Visit: '%{DATA:visit}'\]
USERNAME \[User:'%{DATA:username}'\]

#Action :
ACTION \[(?<action>UPLOAD|READING)\]
ACTION2 action:'\[%{DATA:action}\]

#Wrapper :
WRAPPER .*\|.*\|.*\|%{SPACE}
WRAPPERD .*\|.*\| %{DATEWP:date} \|

#Date :
DATEWP %{YEAR}/%{MONTHNUM}/%{MONTHDAY} %{TIME}
CUSTUMDATE1 %{MONTH} %{MONTHDAY}, %{YEAR} %{TIME} (AM|PM)
DATESRC %{YEAR}%{MONTHNUM2}%{MONTHDAY}
```