**Syslog-NG**

|  |
| --- |
| **Conf** |
| /etc/syslog-ng/syslog-ng.conf |

**Caractéristiques :**

* Collecte et forwarding de logs
* Buffer disque : Stockages des logs sur DD si le serveur central ne répond plus et réémission dans l’ordre.
* TLS pour l’échange de logs entre client et serveur.
* Filtres pour les messages entrants.
* Parsing des messages entrants, notamment JSON, rewrinting, corrélation et agrégation de plusieurs logs entrants en un seul sortant.
* Supporte HDFS/Elastic search, Apache Kafka, AMPQ, STOMP, MongoDB, MSSQL…Graphite…
* Supporte logs JSON.
* Supporte IPv6.
* Flow-control.
* Ne gère pas la high-availability : c’est à l’OS de le faire.

**Arrivé d’un message :**

* Parsing automatique du message entrant qui est au format syslog (ancien ou nouveau), erreur sinon.
* Si aucun timezone, on ajoute le timezone local ou le timezone spécifié.
* Faire attention lors du forward du message à spécifier la bonne time-zone : le timestamp sera modifié en conséquence.

**Ancien Syslog : RFC 3164 (Syslog legacy ou BSD syslog)**

1 K maxi, message contenant :

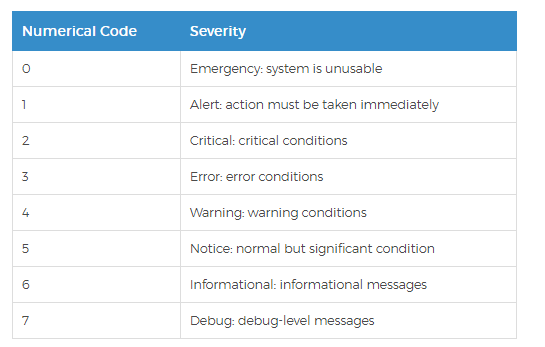
* PRI = 8 \* facility + severity
* HEADER : timestamp + hostname/IP
* MSG : Name of program/PID : msg

**New Syslog : RFC 5424 (IETF Syslog)**

Message contient :

* HEADER : Priority, version (1), ISOTIMESTAMP, HOSTNAME/IP, APP, PID, MSGID (ASCII format)
* STRUCTURED-DATA : meta-informations: [name="value"…] (UTF-8 format) Syslog-ng parse les datas structurées et on y a accès avec des macros.
* MSG (UTF-8 format)





**Option : Parsing du message entrant qui est au format Syslog : on utilise alors ensuite des macros pour accéder aux paramètres.**

Value-pairs( **!!!!Faire un tableau plutôt !!!!!**

Scope( //on Selectionne un groupe de Macro prédéfinies.

Nv\_pairs //name-value pairs : soft macros (sauf les macros .\*)

Core //=base=rfc3164

Syslog //=rfc5424

All\_macros //Toutes les hard-macros

Selected-macros //rfc 3164+ TAGS, SOURCEIP, SEQNUM

sdata //Toutes les structured-data du message nouveau-syslog

everything

)

Exlude("R\_\*") //Exclure certaines macros

Exclude("S\_\*")

Exclude(".SDATA")

Key("HOST" , "PROGRAM") //Macros spécifiques

Pair("MSGHDR" "$PROGRAM[$PID] : ") //Définition de nouvelles macros

)

**Syntaxe générale du fichier de conf :**

@version : 3.8 #Obligatoire

@define name "value" #Accès à la variable avec `name`

@define allow-config-dups 1 #Autorisation de définir deux fois un objet avec le même nom.

#Le dernier prend la valeur finale.

@include "file" #idem qu’en C, \* marche .

**#Generality**

**#Syslog-ng –syntax-only** pour checker la validité du fichier de configuration.

# \_ ou – équivalents

# virgule ou espace équivalents dans les arguments d’une fonction.

# Nombres : 12 (base 10 ) 012 (octale) 0x12 (hexa).

#Utilisé les mêmes drivers pour client et serveur.

#Source est en écoute sur un port uniquement si elle est définit dans un log statement.

* **Inline définition**: comme les classes anonymes en java qui ne fonctionnent qu’une fois : au lieu de source(s\_source) ; on utilise : source { internal() ;} ;
* **Plusieurs confs dans un même objet : fonction channel{} ;**



On pourra utiliser les résultats du parser dans les autres objets du log statement.

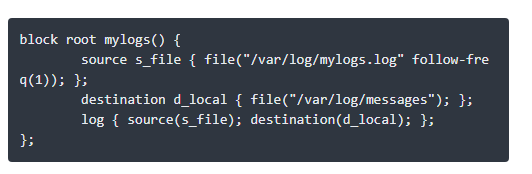
* **SCL : source configuration file = blocks = C++ templates = bloc de code réutilisable dans un fichier à part. On peut passer des arguments à un block.**

Appel avec block()

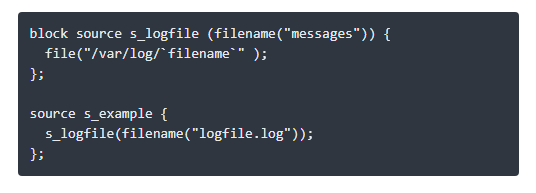




Utiliser root si plusieurs objets dans le block :



Passation d’arguments à un block avec valeur par défaut :



#${HOST} : Macro

**#Options**

Options{

Use\_dns(no) ;

Log-fifo-size(2Mb) ; #ou 2Mib

} ;

**#Sources**

Source s\_name {

source\_driver (arg1 , [arg2] , …) ;

source\_driver2(arg1 , [arg2] , …) ;

…

|  |  |  |
| --- | --- | --- |
| **Source driver** | **Args** | **Comments** |
| Internal( [args] ) ; |  | Messages internes à Syslog-ng |
|  | Host-override("new name") | Remplace ${HOST} du msg. |
|  | Tags("") | Labélisation du msg. |
|  | Use-fqdn(yes|no) | Long hostname | short |
| File("filename") ; |  |  |
|  |  |  |
|  |  |  |

Unix-stream("path to socket" max-connections(10) group(log)) ;

**#écoute sur une socket SOCK\_STREAM**

Network(ip(X.X.X.X) port(X)) ;

**#TCP via rfc3164**

Network(ip(X.X.X.X) port(X) transport("udp")) ;

**#UDP via RFC3164**

Syslog("X.X.X.X" transport("tcp") port(2010)) ;

**#TCP via RFC 5424**

**#>3.2, syslog() driver supporte les Deux versions de syslog.**

File("path" follow-freq(1) flags(no-parse , validate utf-8)) ;

**#check toutes les 1 secondes**

**#ne parse pas en format syslog**

Snmptrap() ;

System() ; **#messages systèmes**

Stdin() ;

…

} ;

**#Destination**

Destination d\_name{

Graylog2( //syslog-ng >=3.14

host ("X.X.X.X")

) ;

} ;

**#Filtres**

**#Logs paths = log statement = correspondance sources/filtres/destination**

Log{

Source(s\_name) ;

Filter(filtre) ;

Destination(d\_name) ;

Flags(flags) ;

} ;

**Modules :**

1 module = 1 ou plusieurs plugins = source/destination driver par exemple.

**Syslog-ng –version** : liste des modules available

**Syslog-ng –module-registry** : description des modules available

**Syslog-ng –defaut-modules** : Modules chargés au démarrage de syslog-ng

@module moduleName pour inclure un module.

@define autoload-compiled-modules 0 pour ne charger aucun modules automatiquement au démarrage.