Surveillance de Scripts LUA et de réception d'EVENT

avec LoriotPro Extended & Broadcast Edition

L'objectif de ce document est de présenter une solution de surveillance de processus LUA au sein de la solution LoriotPro. Les scripts présentés peuvent aussi être utilisé comme outil de « KeepAlive » pour vérifier la réception régulière d'EVENT dans des intervalles de temps contrôlés. Cela peut aussi s'appliquer au Trap SNMP ou au syslog après conversion en EVENT. Cette option apporte une réponse à la surveillance d'équipements SNMP capables d'émettre des TRAP de KEEP ALIVE par exemple.

Par ailleurs le cas de figure est assez courant ou des processus de surveillance basés sur des scripts LUA doivent remplir à intervalle régulier des fonctions de surveillance. Ces scripts peuvent être non opérationnels pour des raisons techniques, défaut de programmation ou bug. Il est donc utile de pouvoir contrôler leur bonne exécution régulièrement.

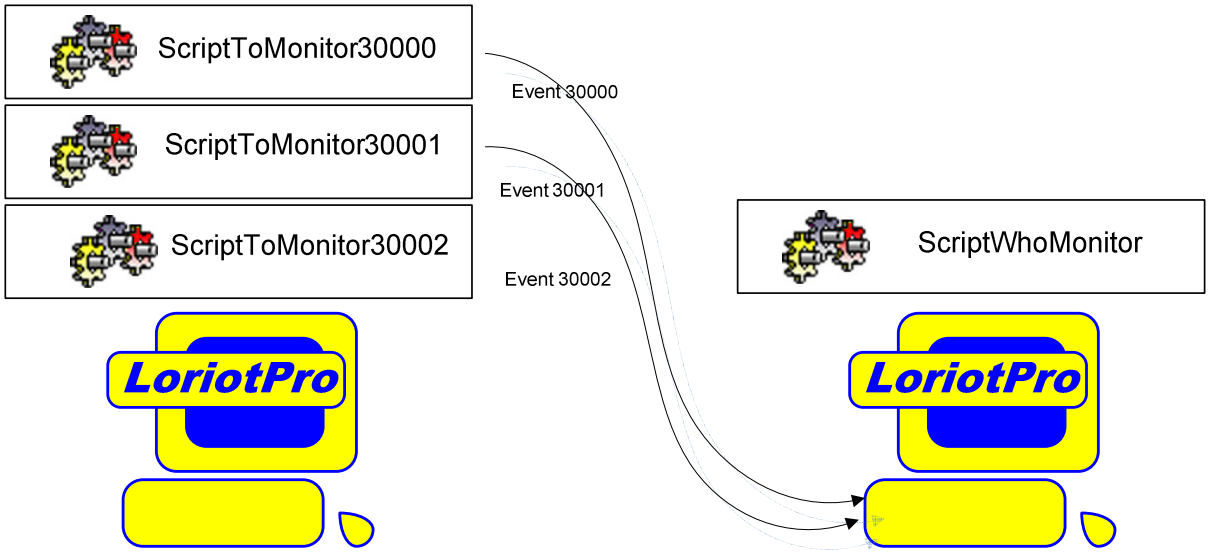
Le principe consiste donc à faire surveiller ceux-ci par un troisième processus basé sur un script LUA qui s'exécutera localement sur le LoriotPro ou sur un LoriotPro distant pour avoir une meilleure garantie de contrôle.

Pour démontrer le concept un jeu de quatre scripts a été créé.

- Script WhoMonitor.lua (Script de surveillance de scripts)
- ScriptToMonitor30000.lua (Script de traitement, génère l'Event 30000 avant de terminer)
- ScriptToMonitor30001.lua (Script de traitement, génère l'Event 30001 avant de terminer)
- ScriptToMonitor30002.lua (Script de traitement, génère l'Event 30002 avant de terminer)

Télécharger ces scripts

Les exemples ci-dessus sont écrits pour fonctionner sur un seul LoriotPro ou des LoriotPro distincts.



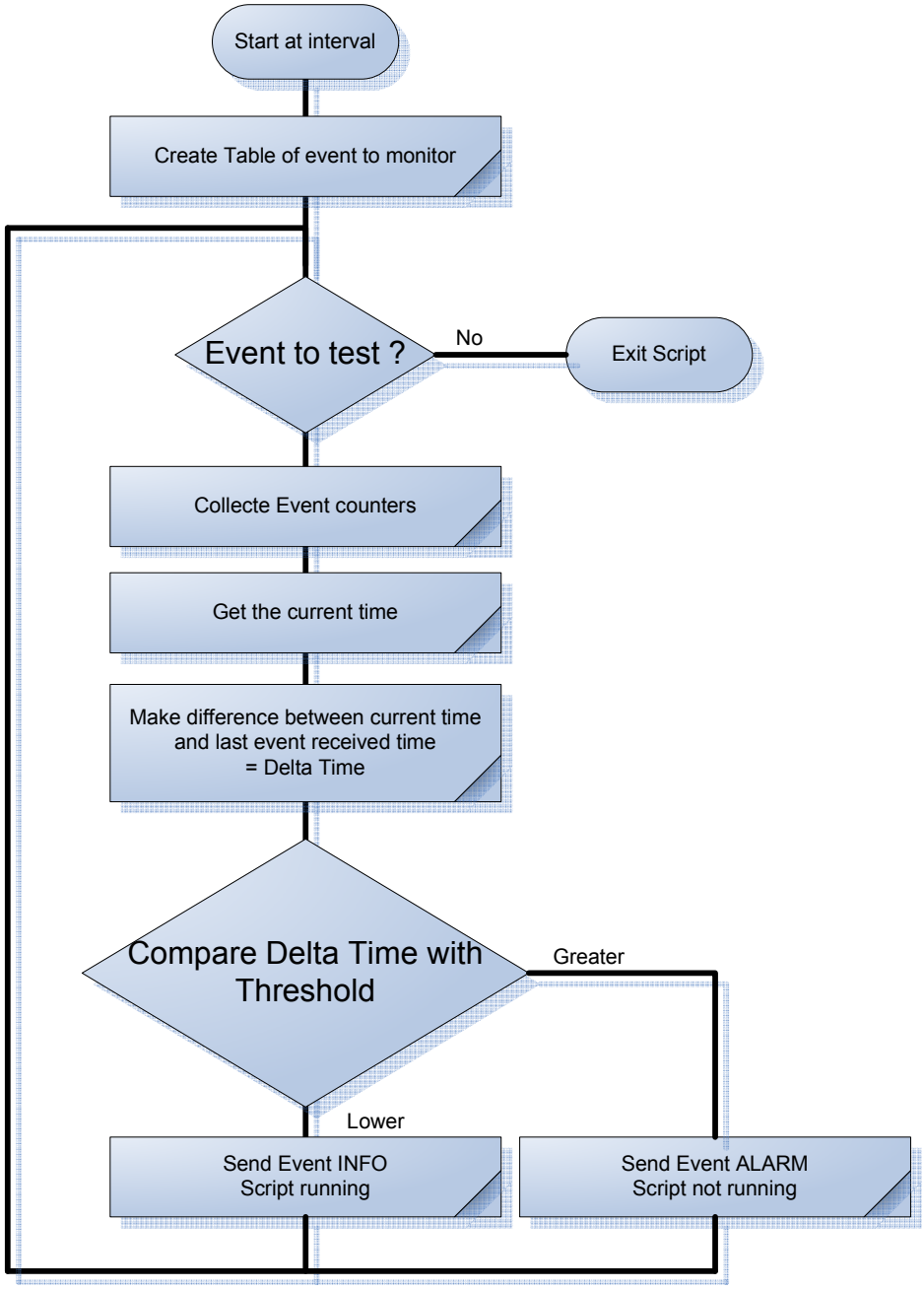
Le script de monitoring des autres scripts est basé sur l'algorithme suivant : En début de script une table est initialisée avec les numéros des événements (EVENT) associés aux scripts à surveiller et les seuils de temps maximums tolérable entre chaque exécution.

Ensuite pour chacun des événements de la table on collecte les informations courantes de l'événement. Un paramètre plus particulier nous intéresse dans cette table, c'est la date de réception du dernier événement reçu (variable Array.last).

Nous collectons ensuite la variable de temps (horloge du système) pour faire une différence entre ces deux variables. Cette valeur correspond au temps en seconde écoulé depuis la dernière fois que le script s'est terminé.

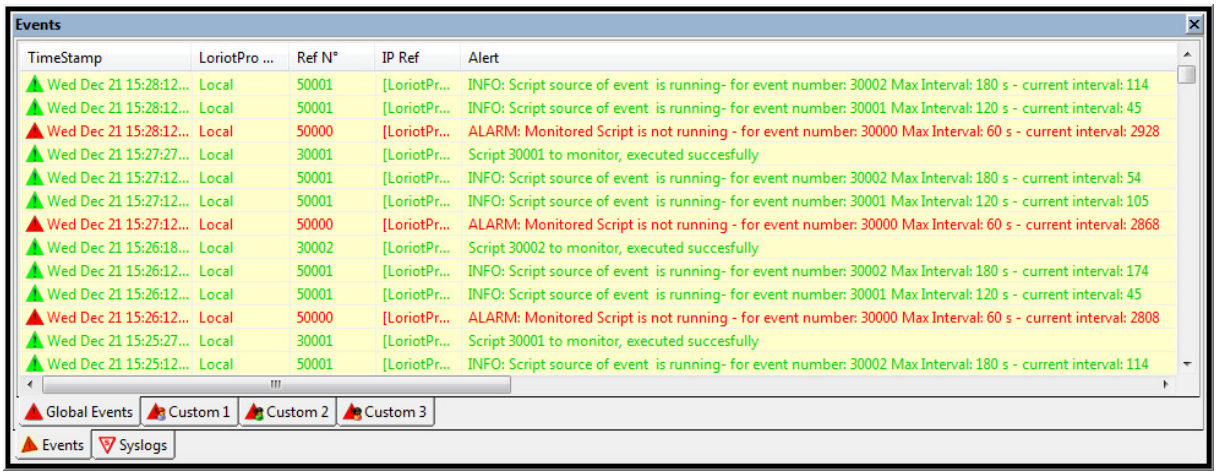
Une comparaison est ensuite faite avec leur seuil tolérable défini aussi dans la table des EVENT initialisé en debut de script.

En fonction du résultat de cette comparaison un événement est envoyé à LoritoPro pour informer ou alerter l'administrateur d'un défaut. L'événement 50000 informe d'une anomalie (En rouge) tandis que le 5001 est une notification de bon résultat (En vert dans les log)



Exemple de fonctionnement permanent

On constate que les scripts « **ScriptToMonitor30001 »** et « **ScriptToMonitor30002 »** fonctionnent bien par la présence des EVENT 30001 et 30002 par contre l'absence de 30000 est détecté par le monitor de script « ScriptWhoMonitor.lua » qui génère alors un EVENT 50000 en rouge.



Mise œuvre

La mise en œuvre consiste à mettre en place le script de surveillance « ScriptWhoMonitor.lua » et le faire s'exécuter à intervalle régulier.

Il faut copier le script « ScriptWhoMonitor.lua » dans le répertoire /config/script de LoriotPro

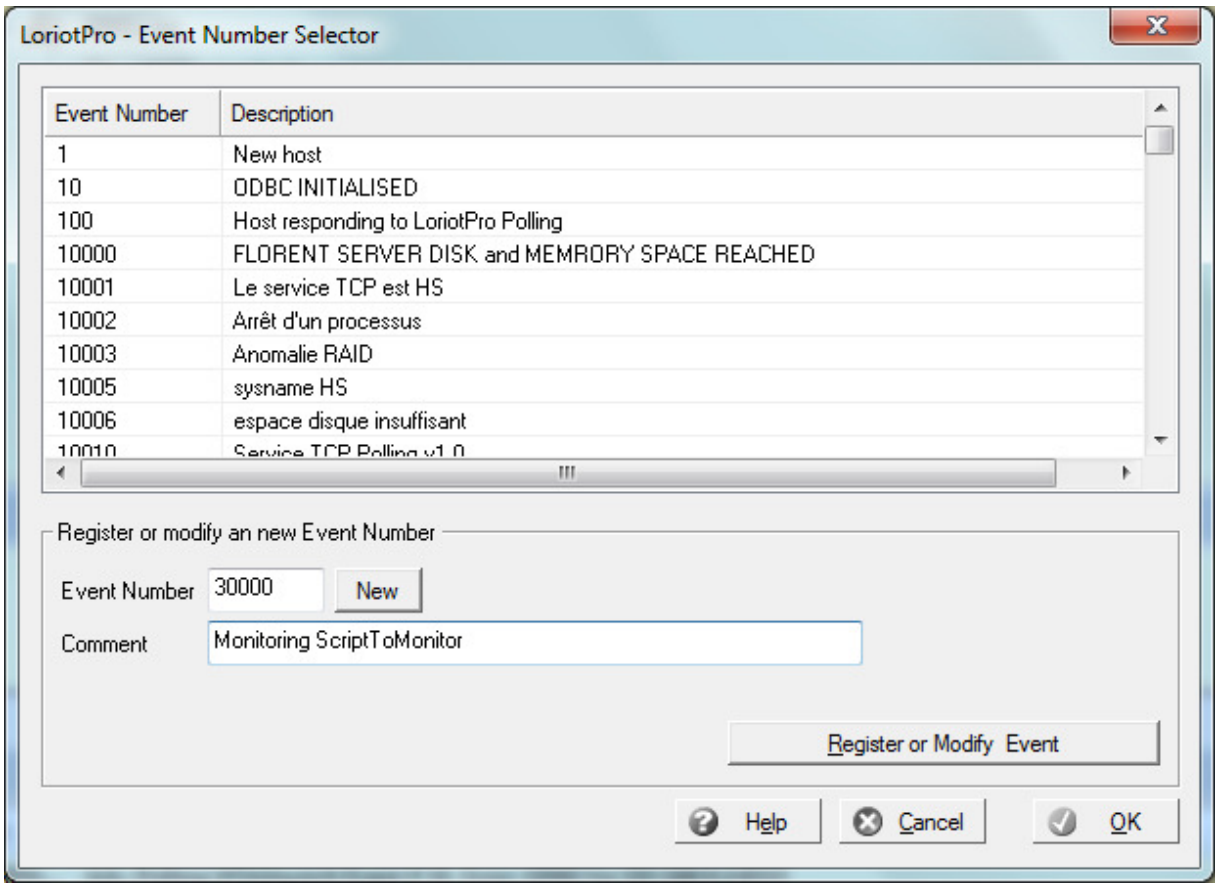
Il faut ensuite modifier la table ScriptList des événements en fonction des événements générés par les scripts à surveiller.

Exemple de table dans notre cas :

```
ScriptList = {
        {
        event_number = 30000, -- The event number that uniqueley identify this message
        MaxInterval = 60 -- Interval de temp maximum en secondes supporté pour la réception de cet event
        },
        {
        event_number = 30001,
        MaxInterval = 120 -- 2 mn
        },
        {
        event_number = 30002,
        MaxInterval = 180 -- 3 mn
        }
} -- Un tableau contenant la liste des events surveillées
```

ATTENTION à bien respecter la présence des virgules.

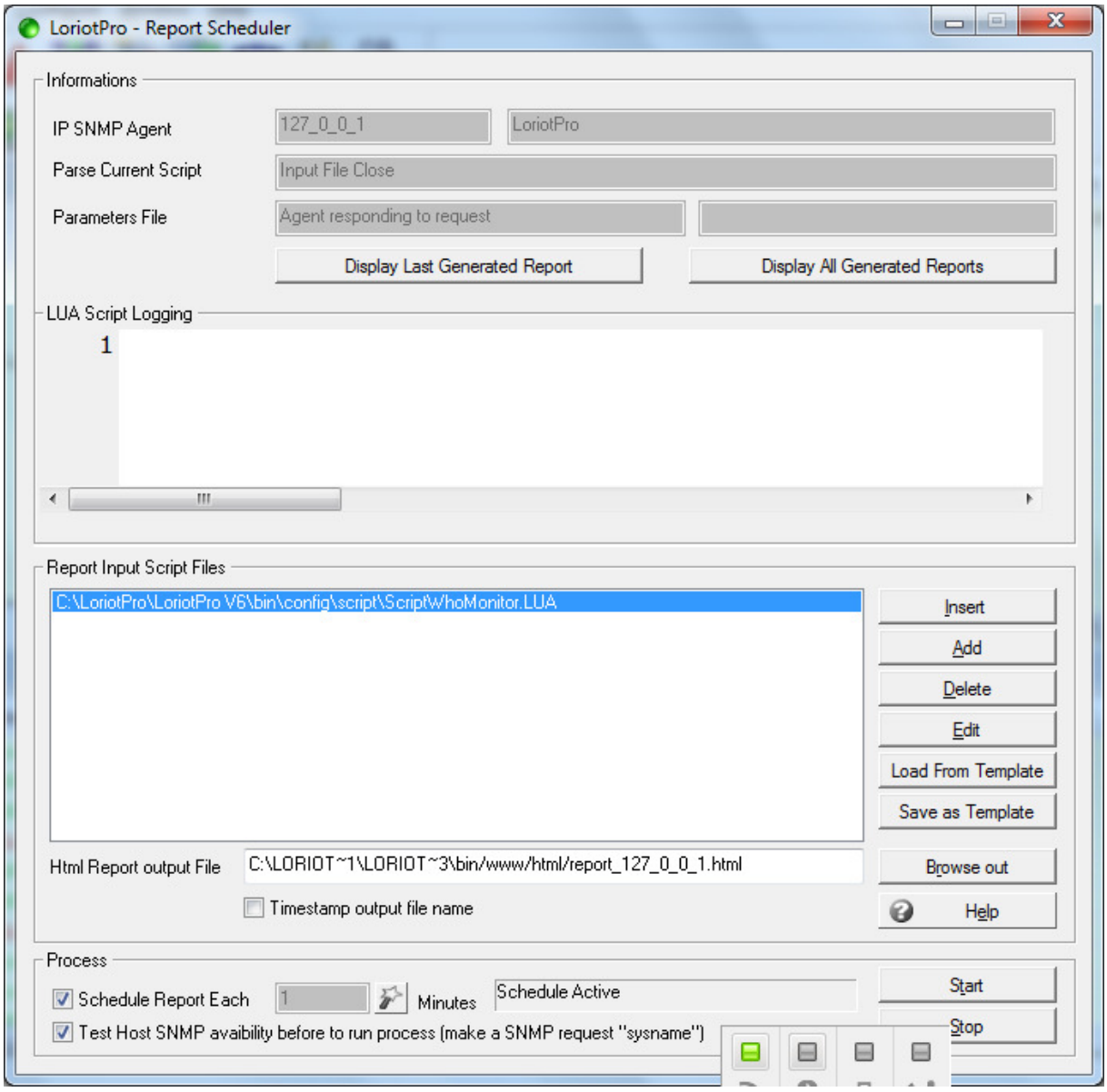
Avant d'utiliser un EVENT il faut impérativement le déclarer. Cela peut se faire en sélectionnant à partir du menu de LoriotPro l'option : ***Configure -> Register New Event Number***



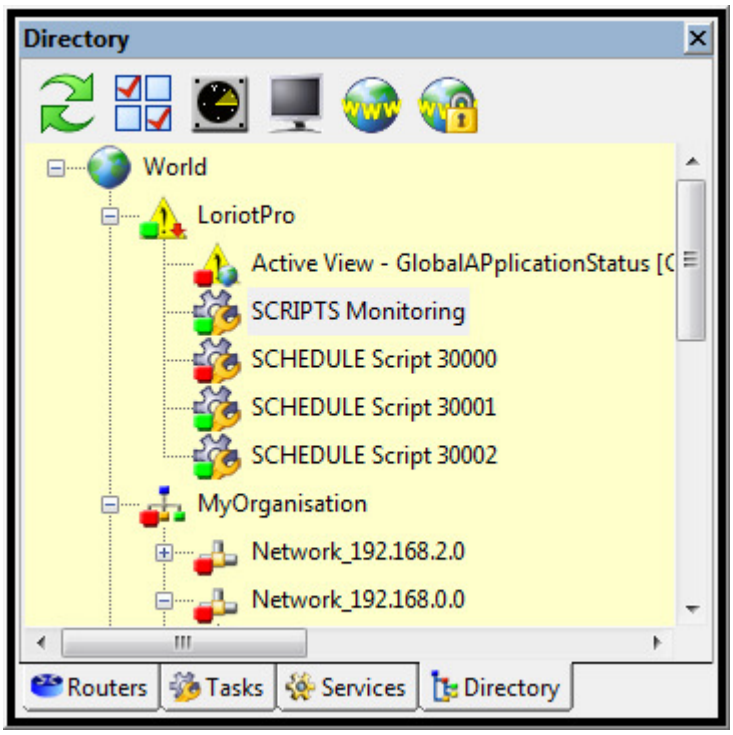
Le « ScriptWhoMonitor.lua » peut être placé dans un plugin ***« ReportSCheduler »*** pour être exécuté à intervalle régulier.

ATTENTION il faut que la fréquence d'exécution soit inférieure au seuil maximum choisi (MaxInterval) pour l'envoi des événements.

Exemple avec le plugin Report Scheduler configuré à 1 mn. Pour ajouter un plugin Report SCheduler choisir un host de l'annuaire puis par click droit sélectionner ***« Insert Task (plugin) »*** puis sélectionner ***« ReportSCheduler ».*** Pour insérer un script LUA dans ***« ReportSCheduler ».***, il faut naviguer vers le répertoire des scripts LUA dans /bin/config/script et sélectionner le type .lua.



Exemple de configuration de test avec les quatre ***« ReportSCheduler »*** exécutant les quatre scripts à intervalle programmé.



Il serait idéal de fonctionner sur deux LoriotPro différent, les scripts de traitement sur une premier système et le script de monitoring des EVENT sur une deuxième système.

Dans ce cas il faut modifier dans les scripts l'adresse IP destination utilisé dans notre exemple dans les scripts de Traitement « ScriptToMonitor3000x »

Script de monitoring type KeepALive « Script WhoMonitor.lua »

```
-- LoriotPro V6.00
-- To run correctly this file is located to bin/config/script
-- Input values
-- lp_index index for this script ".1"
-- lp_oid SNMP OID for this script "ifnumber"
-- lp_host default ip address for this script "127.0.0.1"
-- Output Values
  lp_value = 0;
  lp_buffer ="error";
-- dofile(lp.GetPath().."/config/script/loriotinit.lua"); -- to support define
-- lp.InitLuaPath(); -- to support luaforwindows librairies and hook print function

CR="\n"

ScriptList = {
            {
            event_number = 30000, -- The event number that uniqueley identify this message
            MaxInterval = 60 -- Interval de temp maximum en secondes supporté pour la réception de cet event
            },
            {
            event_number = 30001,
            MaxInterval = 120 -- 2 mn
            },
            {
            event_number = 30002,
            MaxInterval = 180 -- 3 mn
            }
} -- Un tableau contenant la liste des events surveillées
```

```
for k,v in ipairs(ScriptList) do
	event_number = ScriptList[k].event_number
	MaxInterval = ScriptList[k].MaxInterval
	lp.Trace("The event Number: "..event_number..CR)
	OkNok = lp.GetEventInformation(event_number,"Array")

if OkNok then
	lp.Trace("The event description: "..Array.description..CR) -- The event description
	lp.Trace("The timestamp of the last event received: "..Array.last..CR) -- The timestamp of the last event received
	lp.Trace("The timestamp of the first event received: "..Array.first..CR) -- The timestamp of the first event received
	lp.Trace("The timestamp of the first event received since the last global acknowledgment: "..Array.first_since..CR) -- The
timestamp of the first event received since the last global acknowledgment.
	lp.Trace("The total number of events received: "..Array.number_total..CR) -- The total number of events received
	lp.Trace("The number of events acknowledged:  "..Array.number_ack..CR) -- The number of events acknowledged
	lp.Trace("The number of events cleared:  "..Array.number_delete..CR) -- The number of events cleared
	lp.Trace("The number of events that was automatically cleared when the listbox is full:  "..Array.number_auto_delete..CR) -- The
number of events that was automatically cleared when the listbox is full
	lp.Trace("The number of events received since a global acknowledgment: "..Array.number_since_clear..CR) -- The number of
events received since a global acknowledgment.
	lp.Trace("The number of event not display in the workspace event viewer: "..Array.number_not_displayed..CR) -- The number of
event not display in the workspace event viewer.
	lp.Trace("The total number of events displayed in the workspace event viewer: "..Array.number_displayed..CR) -- The total
number of events displayed in the workspace event viewer.
	lp.Trace("The maximum number of event to display in the event viewer listbox: "..Array.max_display..CR) -- The maximum
number of event to displkay in the event viwer listbox (-1 equal no limit)
	lp.Trace(Array.number_between_action..CR) -- Reserved
	lp.Trace("State of the filter: "..Array.disable..CR) --  State of the filter  1 - The filter is enable (actions also)0 – The filter is disable
	else
	lp.Trace("Error in lp.GetEventInformation for event: "..event_number..CR)
	lp.Trace("Event probably not defined in LoriotPro"..CR)
end

CurrentTime = os.time()

lp.Trace("The current timestamp "..CurrentTime..CR)
lp.Trace("The timestamp of the last event received: "..Array.last..CR)
ElapsedTime = os.difftime(CurrentTime, Array.last)
lp.Trace("Elapsed time since last event: "..ElapsedTime..CR)

if ElapsedTime > MaxInterval then -- If no event received during this interval of time
	ipdest = "127.0.0.1" -- The ip adresse of the LoriotPro where to send the event
	port = 5001 -- The server port of the LoriotPro (by default alway 5001)
	eventnumber = 50000 -- The event number that uniqueley identify this message
	level = 4 -- The severity of the event, 2 is green, 4 is red
	ipref = "127.0.0.1" -- the IP adresse of the host that is concerned by this event
	ipmask = "255.255.255.255" -- the mask that select the useful bits of the previous address
	buffer = "ALARM: Monitored Script is not running - for event number: "..event_number.." Max Interval: "..MaxInterval.." s -
current interval: "..ElapsedTime-- the event message text

	lp.SendExternEvent(ipdest,port,eventnumber,level,ipref,ipmask,buffer)
else
	ipdest = "127.0.0.1" -- The ip adresse of the LoriotPro where to send the event
	port = 5001 -- The server port of the LoriotPro (by default alway 5001)
	eventnumber = 50001 -- The event number that uniqueley identify this message
	level = 2 -- The severity of the event, 2 is green, 4 is red
	ipref = "127.0.0.1" -- the IP adresse of the host that is concerned by this event
	ipmask = "255.255.255.255" -- the mask that select the useful bits of the previous address
	buffer = "INFO: Script source of event  is running- for event number: "..event_number.." Max Interval: "..MaxInterval.." s -
current interval: "..ElapsedTime -- the event message text

	lp.SendExternEvent(ipdest,port,eventnumber,level,ipref,ipmask,buffer)
end

lp.Trace("-------------------------------------------------------"..CR)
end
```

Ce script nécessite des modifications en fonction des numéros d'événement choisis sur les scripts surveillés. Les valeurs doivent être saisies dans la table en début « ScriptList »

Script de traitement dans notre exemple

```
-- This si the script that should be Monitored
-- Ceci est le script qui doit être surveillé

--Le code du script (pour l'exemple, il ne fait rien sauf attendre 1 minute avant de quitter
--The script code, it deos nothing for this exemple except waiting n seconde
delai= 10  --en secondes
for i=1,delai do
lp.Trace(i)
lp.Sleep(1000)
lp.Trace("-")
end

--En fin de script on envoi un Event
--We send an event at the end

ipdest = "127.0.0.1" -- The ip adresse of the LoriotPro where to send the event
port = 5001 -- The server port of the LoriotPro (by default alway 5001)
eventnumber = 30002 -- The event number that uniqueley identify this message
level = 2 -- The severity of the event, 2 is green, 4 is red
ipref = "127.0.0.1" -- the IP adresse of the host that is concerned by this event
ipmask = "255.255.255.255" -- the mask that select the useful bits of the previous address
buffer = "Script "..eventnumber.." to monitor, executed succesfully" -- the event message text

lp.SendExternEvent(ipdest,port,eventnumber,level,ipref,ipmask,buffer)
```

En fin de script on trouve un exemple de code à insérer dans les scripts LUA à surveiller. L'adresse « ipdest » doit être modifiée en fonction du LoriotPro de réception des EVENT. Si c'est en local conserver l'adresse loopback. Chaque script à surveiller doit générer une événement EVENT avec un numéro distinct (Variable eventnumber). Ipref et ipmask sont facultatif et ne change pas le fonctionnement de l'ensemble.

Nous terminons ainsi notre How To de surveillance de Scripts LUA et de réception d'EVENT au sein de LoriotPro. Un exemple très utile pour vérifier que des événements réguliers ont bien lieu dans les contraintes attendues.

Bon courage pour votre intégration et n'hésitez pas ou nous faire part de vos remarques et améliorations, merci.

Auteur : F.Brisson Société LUTEUS