Scripting examples

Introduction

This section contains some useful scripts and shows all available scripting features. Script examples used in this section were tested with the latest 3.x version.

Create a file

it is not possible to create a file directly, however, there is a workaround:

```
/file print file=myFile
/file set myFile.txt contents=""
```

Append text to a file in a new line

There is no direct way to append text to a file, however you can store the old content and append to it in a new line:

```
:local oldText [/file get test.txt contents as-string]
:local addText "test append"
:local newText ($oldText."\n".$addText)
/file set myFile.txt contents=$newText
```

Check if IP on the interface has changed

Sometimes provider gives dynamic IP addresses. This script will compare if a dynamic IP address is changed.

```
:global currentIP;

:local newIP [/ip address get [find interface="ether1"] address];

:if ($newIP != $currentIP) do={
    :put "ip address $currentIP changed to $newIP";
    :set currentIP $newIP;
}
```

Strip netmask

This script is useful if you need an IP address without a netmask (for example to use it in a firewall), but "`/ip address get [id] address`" returns the IP address and netmask.

```
:global ipaddress 10.1.101.1/24

:for i from=( [:len $ipaddress] - 1) to=0 do={
        :if ( [:pick $ipaddress $i] = "/") do={
                :put [:pick $ipaddress 0 $i]
        }
}
```

Another much more simple way:

```
:global ipaddress 10.1.101.1/24
:put [:pick $ipaddress 0 [:find $ipaddress "/"]]
```

Resolve host-name

Many users are asking features to use DNS names instead of IP addresses for radius servers, firewall rules, etc.

So here is an example of how to resolve the RADIUS server's IP.

Let's say we have the radius server configured:

```
/radius
add address=3.4.5.6 comment=myRad
```

And here is a script that will resolve the IP address, compare resolved IP with configured one, and replace it if not equal:

```
/system script add name="resolver" source= {

:local resolvedIP [:resolve "server.example.com"];
:local radiusID [/radius find comment="myRad"];
:local currentIP [/radius get $radiusID address];

:if ($resolvedIP != $currentIP) do={
   /radius set $radiusID address=$resolvedIP;
   /log info "radius ip updated";
}

}
```

Add this script to the scheduler to run for example every 5 minutes

```
/system scheduler add name=resolveRadiusIP on-event="resolver" interval=5m
```

Write simple queue stats in multiple files

Let's consider queue namings are "some text.1" so we can search queues by the last number right after the dot.

```
:local entriesPerFile 10;
:local currentQueue 0;
:local queuesInFile 0;
:local fileContent "";
#determine needed file count
:local numQueues [/queue simple print count-only] ;
:local fileCount ($numQueues / $entriesPerFile);
:if ( ($fileCount * $entriesPerFile) != $numQueues) do={
   :set fileCount ($fileCount + 1);
}

#remove old files
/file remove [find name~"stats"];

:put "fileCount=$fileCount";

:for i from=1 to=$fileCount do={
#create file
   /file print file="stats$i.txt";
#clear content
   /file set [find name="stats$i.txt"] contents="";

   :while ($queuesInFile < $entriesPerFile) do={
     :if ($currentQueue < $numQueues) do={
         :set currentQueue ($currentQueue +1);
         :put $currentQueue ;
         /queue simple
         :local internalID [find name~"\\.$currentQueue\$"];
         :put "internalID=$internalID";
         :set fileContent ($fileContent . [get $internalID target-address] . \
           " " . [get $internalID total-bytes] . "\r\n");
     }
     :set queuesInFile ($queuesInFile +1);

   }
   /file set "stats$i.txt" contents=$fileContent;
   :set fileContent "";
   :set queuesInFile 0;

}
```

Generate backup and send it by e-mail

This script generates a backup file and sends it to a specified e-mail address. The mail subject contains the router's name, current date, and time.

Note that the SMTP server must be configured before this script can be used. See /tool e-mail for configuration options.

```
/system backup save name=email_backup
/tool e-mail send file=email_backup.backup to="me@test.com" body="See attached file" \
   subject="$[/system identity get name] $[/system clock get time] $[/system clock get date] Backup"
```

The backup file contains sensitive information like passwords. So to get access to generated backup files, the script or scheduler must have a 'sensitive' policy.

Use string as a function

```
:global printA [:parse ":local A; :put \$A;" ];
$printA
```

Check bandwidth and add limitations

This script checks if the download on an interface is more than 512kbps if true then the queue is added to limit the speed to 256kbps.

```
:foreach i in=[/interface find] do={
    /interface monitor-traffic $i once do={
        :if ($"received-bits-per-second" > 0 ) do={
            :local tmpIP [/ip address get [/ip address find interface=$i] address] ;
#            :log warning $tmpIP ;
            :for j from=( [:len $tmpIP] - 1) to=0 do={
                :if ( [:pick $tmpIP $j] = "/") do={
                    /queue simple add name=$i max-limit=256000/256000 dst-address=[:pick $tmpIP 0 $j] ;
                }
            }
        }
    }
}
```

Block access to specific websites

This script is useful if you want to block certain websites but you don't want to use a web proxy.

This example looks at entries "Rapidshare" and "youtube" in the DNS cache and adds IPs to the address list named "restricted". Before you begin, you must set up a router to catch all DNS requests:

```
/ip firewall nat
add action=redirect chain=dstnat comment=DNS dst-port=53 protocol=tcp to-ports=53
add action=redirect chain=dstnat dst-port=53 protocol=udp to-ports=53
```

and add firewall

```
/ip firewall filter
add chain=forward dst-address-list=restricted action=drop
```

Now we can write a script and schedule it to run, let's say, every 30 seconds.

Script Code:

```
:foreach i in=[/ip dns cache find] do={
    :local bNew "true";
    :local cacheName [/ip dns cache all get $i name] ;
#    :put $cacheName;

    :if (([:find $cacheName "rapidshare"] >= 0) || ([:find $cacheName "youtube"] >= 0)) do={

        :local tmpAddress [/ip dns cache get $i address] ;
#        :put $tmpAddress;

# if address list is empty do not check
        :if ( [/ip firewall address-list find list="restricted" ] = "") do={
            :log info ("added entry: $[/ip dns cache get $i name] IP $tmpAddress");
            /ip firewall address-list add address=$tmpAddress list=restricted comment=$cacheName;
        } else={
            :foreach j in=[/ip firewall address-list find list="restricted"] do={
                :if ( [/ip firewall address-list get $j address] = $tmpAddress ) do={
                    :set bNew "false";
                }
            }
            :if ( $bNew = "true" ) do={
                :log info ("added entry: $[/ip dns cache get $i name] IP $tmpAddress");
                /ip firewall address-list add address=$tmpAddress list=restricted comment=$cacheName;
            }
        }
    }
}
```

Parse file to add ppp secrets

This script requires that entries inside the file are in the following format:

username,password,local_address,remote_address,profile,service

For example:

```
janis,123,1.1.1.1,2.2.2.1,ppp_profile,myService
juris,456,1.1.1.1,2.2.2.2,ppp_profile,myService
aija,678,1.1.1.1,2.2.2.3,ppp_profile,myService
```

```
:global content [/file get [/file find name=test.txt] contents] ;
:global contentLen [ :len $content ] ;

:global lineEnd 0;
:global line "";
:global lastEnd 0;


:do {
       :set lineEnd [:find $content "\r\n" $lastEnd ] ;
       :set line [:pick $content $lastEnd $lineEnd] ;
       :set lastEnd ( $lineEnd + 2 ) ;

       :local tmpArray [:toarray $line] ;
        :if ( [:pick $tmpArray 0] != "" ) do={
        :put $tmpArray;
         /ppp secret add name=[:pick $tmpArray 0] password=[:pick $tmpArray 1] \
             local-address=[:pick $tmpArray 2] remote-address=[:pick $tmpArray 3] \
             profile=[:pick $tmpArray 4] service=[:pick $tmpArray 5];
}
} while ($lineEnd < $contentLen)
```

Detect new log entry

This script is checking if a new log entry is added to a particular buffer.

In this example we will use PPPoE logs:

```
/system logging action
add name="pppoe"
/system logging
add action=pppoe topics=pppoe,info,!ppp,!debug
```

Log buffer will look similar to this one:

```
[admin@mainGW] > /log print where buffer=pppoe
13:11:08 pppoe,info PPPoE connection established from 00:0C:42:04:4C:EE
```

Now we can write a script to detect if a new entry is added.

```
:global lastTime;

:global currentBuf [ :toarray [ /log find buffer=pppoe  ] ] ;
:global currentLineCount [ :len $currentBuf ] ;
:global currentTime [ :totime [/log get [ :pick $currentBuf ($currentLineCount -1) ] time   ] ];

:global message "";

:if ( $lastTime = "" ) do={
        :set lastTime $currentTime ;
        :set message [/log get [ :pick $currentBuf ($currentLineCount-1) ] message];

} else={
        :if ( $lastTime < $currentTime ) do={
                :set lastTime $currentTime ;
                :set message [/log get [ :pick $currentBuf ($currentLineCount-1) ] message];
        }
}
```

After a new entry is detected, it is saved in the "message" variable, which you can use later to parse log messages, for example, to get the PPPoE client's mac addresses.

Allow use of ntp.org pool service for NTP

This script resolves the hostnames of two NTP servers, compares the result with the current NTP settings, and changes the addresses if they're different. This script is required as RouterOS does not allow hostnames to be used in the NTP configuration. Two scripts are used. The first defines some system variables which are used in other scripts and the second does the grunt work:

```
# System configuration script - "GlobalVars"

:put "Setting system globals";

# System name
:global SYSname [/system identity get name];

# E-mail address to send notifications to
:global SYSsendemail "mail@my.address";

# E-mail address to send notifications from
:global SYSmyemail "routeros@my.address";

# Mail server to use
:global SYSemailserver "1.2.3.4";

# NTP pools to use (check www.pool.ntp.org)
:global SYSntpa "0.uk.pool.ntp.org";
:global SYSntpb "1.uk.pool.ntp.org";
```

```
# Check and set NTP servers - "setntppool"

# We need to use the following globals which must be defined here even
# though they are also defined in the script we call to set them.
:global SYSname;
:global SYSsendemail;
:global SYSmyemail;
:global SYSmyname;
:global SYSemailserver;
:global SYSntpa;
:global SYSntpb;

# Load the global variables with the system defaults
/system script run GlobalVars

# Resolve the two ntp pool hostnames
:local ntpipa [:resolve $SYSntpa];
:local ntpipb [:resolve $SYSntpb];

# Get the current settings
:local ntpcura [/system ntp client get primary-ntp];
:local ntpcurb [/system ntp client get secondary-ntp];

# Define a variable so we know if anything's changed.
:local changea 0;
:local changeb 0;

# Debug output
:put ("Old: " . $ntpcura . " New: " . $ntpipa);
:put ("Old: " . $ntpcurb . " New: " . $ntpipb);

# Change primary if required
:if ($ntpipa != $ntpcura) do={
    :put "Changing primary NTP";
    /system ntp client set primary-ntp="$ntpipa";
    :set changea 1;
    }

# Change secondary if required
:if ($ntpipb != $ntpcurb) do={
    :put "Changing secondary NTP";
    /system ntp client set secondary-ntp="$ntpipb";
    :set changeb 1;
    }

# If we've made a change, send an e-mail to say so.
:if (($changea = 1) || ($changeb = 1)) do={
    :put "Sending e-mail.";
    /tool e-mail send \
        to=$SYSsendemail \
        subject=($SYSname . " NTP change") \
        from=$SYSmyemail \
        server=$SYSemailserver \
        body=("Your NTP servers have just been changed:\n\nPrimary:\nOld: " . $ntpcura . "\nNew: " \
          . $ntpipa . "\n\nSecondary\nOld: " . $ntpcurb . "\nNew: " . $ntpipb);
    }
```

Scheduler entry:

```
/system scheduler add \
  comment="Check and set NTP servers" \
  disabled=no \
  interval=12h \
  name=CheckNTPServers \
  on-event=setntppool \
  policy=read,write,test \
  start-date=jan/01/1970 \
  start-time=16:00:00
```

Other scripts

- Dynamic_DNS_Update_Script_for_EveryDNS
- Dynamic_DNS_Update_Script_for_ChangeIP.com
- UPS Script