Scripting on RouterOS :for fun and $profit

ANDREW COX

Who Am I?

Andrew Cox

Based in Brisbane, Australia

Omega-00 on the MikroTik Forums

A moderator for the MikroTik Facebook Group

A moderator for the MikroTik Reddit Group

Member of The Brothers WISP podcast

Owner of the MikroTik-RouterOS.com blog

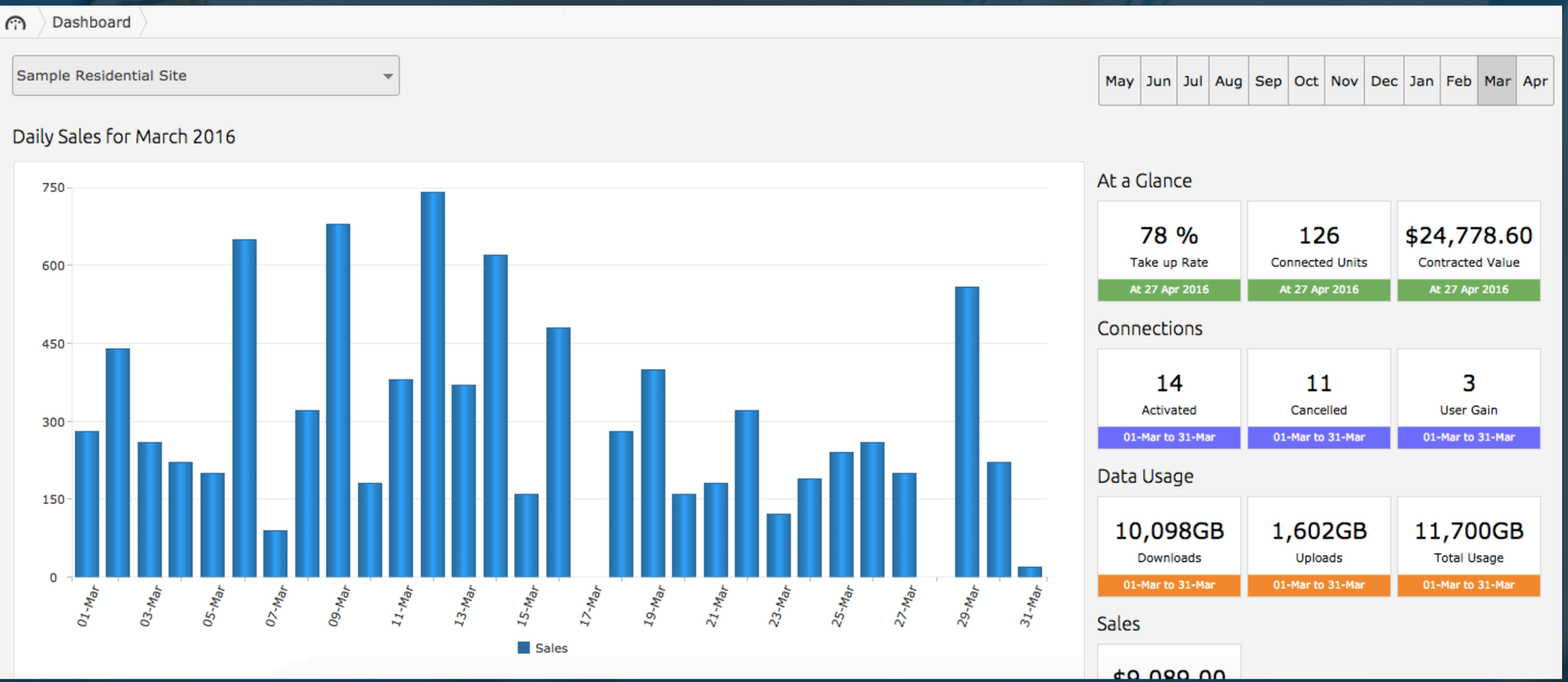
General Manager at Bright WiFi

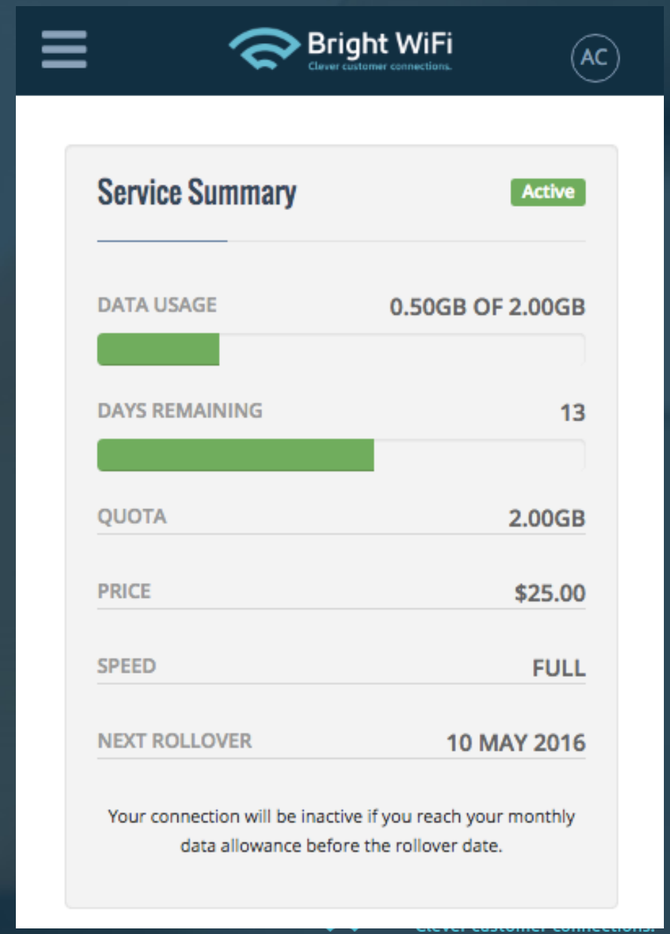
Working with MikroTik since 2005 – v2.9

Bright WiFi
Clever customer connections.

What is Bright WiFi?

Bright WiFi is to traditional hotspot solutions as NASA is to model rocketry…

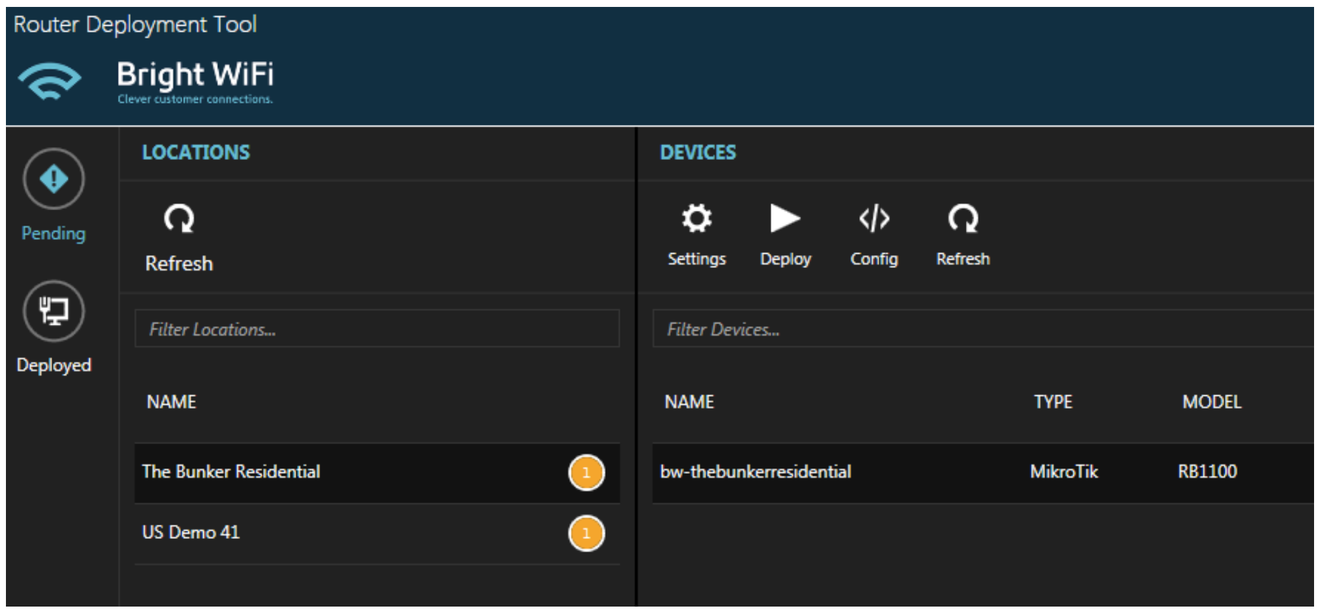




Bright WiFi

ISP AAA, analytics, deployment, and holistic management

- Automated deployment tool
- Centralised management and observation portal
- Easy to use service creation: login page, user access control, billing, reports
- Modular framework allowing integration of 3rd party services



Bright WiFi
Clever customer connections.

RouterOS Scripting

But it’s a router?!

- On-router scripting language
- Time based & on-event scripts
- Access to terminal commands & outputs
- Limited access to read & create files
- Repeatable function declarations
- Incredibly powerful tool!

RouterOS Scripting - WhoWhatWhy

But why?!

- Creating or modifying items en masse – address lists, filter rules, simple queues
- Automatic events that would otherwise require manual intervention
- Extending existing functionality to support new features
- Complex failover options where a solution might otherwise be impossible
- Setup & Backup solutions onboard the router (remote backup via ftp/email)
- Regex searching and updating records – proxy or walled garden rule changes

Bright WiFi
Clever customer connections.

RouterOS Scripting - WhoWhatWhy

Who's doing this?

- MikroTik uses scripting to configure your router ..

  /system default-configuration print
- MikroTik Wiki – http://wiki.mikrotik.com/wiki/scripts >100 scripts
- MikroTik Forums - http://forum.mikrotik.com/viewforum.php?f=9 >5000 threads
- A Google search for "MikroTik Script" .. half a million results!
- Some scripting sources:

  GregSowell.com – Mr Sowell

  Wirelessconnect.eu – Mr Smyth

  MikroTik-RouterOS.com – Me!

  MTHelper – MikroTik configuration and management tool that allows script deployment!

Getting Started

Basic Scripting Commands

:local <variable name> <value>	Defines a local variable with the optional assigned value
:global <variable name> <value>	Defines a global variable with the optional assigned value
:set <variable name> <value>	Updates / Sets the variable to the assigned value
:put <value>	Prints the command result to terminal
:log info/warning/error “<value>”	Creates a log entry using the provided value
$variablesubstitution	Returns variable data - :put $variable1
:if (<condition>) do={ } else={ }	If this then that, else this – standard conditional if statement
[<comand line substitution>]	Allows interpreted value – [/ip hotspot active print count-only]
(<grouping operator>)	Useful for math and concatenation functions – (($x * $y) + 50)
# comment	Add script comment – must be placed at beginning of line

More Advanced

Complex Functions

:resolve example.com	Return the IP address resolution for a DNS lookup
:pick <str> <start> <end>	Return a specific section of the provided value
:len <str>	Return the value length or number of items in an array
:find <str> <tofind>	Find the first occurrence of the search term
:for <str> from=0 to=10 do={ }	Perform an action for the required number of iterations
:foreach <str> in=[/ip arp find] do={$cmd }	Perform an action against each matching instance
:do :while OR :while :do	Perform an action against a conditional check
:set arraytest {“a”;”b”;”c”;”d”}	Create an array of values instead of a basic string
:put ($arraytest->3)	Returning a specific value from an array
[/ip arp find address~"^192"]	Searching for matches using a regular expression
Defined Functions!	Functions/subroutines that can be called inside scripts

Recommendations for starting out

- Use Notepad++ with syntax highlighting!
- Use :put to debug your variables
- Comment out sections when you aren’t sure what broke or use the :do { } on-error={ }
- Be aware of the difference :global :local and variable declarations within scopes

Practical Uses

Example 1:

Collecting useful information about router health and adding it to the system note which is displayed each time a new terminal is opened.

Things of note:

:set term ($term . “append new value”)

/sys reso – shortening /system resource

:pick command – select portion of a string to compare against (could also use regex)

:len command used on array output

```
:local term

:set term "Uptime:  $[/sys reso get uptime]"
:set term ($term . "| CPU: $[/system resource get cpu-load]% ")

#append/move to newline
:set term ($term . "\n")

#voltage / temp readout not available on x86 or x86_64 systems
:if ([:pick [/sys reso get arch] 0 3]="x86") do={
  :set term ($term .  "Voltage: NA |Temp: NA \n")
} else={
  :set term ($term . "Voltage: $[:pick [/sys health get voltage] 0 2] v")
:set term ($term . "Temp: $[/system health get temperature]c | ")
}

#get current hotspot and ppp active sessions counts
:set term ($term . "HS active: $[:len [/ip hotspot active find]]")
:set term ($term . " | PPP Active: $[:len [/ppp active find]]")

#update note
/system note set note="$term"
```

```
:local fetchsuccess true
:local content
:local contentLen
:local lineEnd 0
:local line
:local lastEnd 0
#Start file download using fetch
:do {
  /tool fetch url=http://example.com/address.txt
} on-error={
  :set fetchsuccess false
}
#Continue import if fetch succeedd
:if ($fetchsuccess) do={
  /ip firewall address-list remove [find list=IP-LIST]
  :set content [/file get [find name=address.txt] contents]
  :set contentLen [ :len $content ]
  :do {
    :set lineEnd [:find $content "\n" $lastEnd ]
    :set line [:pick $content $lastEnd $lineEnd]
    :set lastEnd ( $lineEnd + 1 )
#Dont process lines with comments
    :if ( [:pick $line 0 1] != "#" ) do={
      :local entry [:pick $line 0 ($lineEnd -1) ]
      :if ( [:len $entry ] > 0 ) do={
        /ip fire address-list add list=IP-LIST addr=$entry
       }
    }
  } while ($lineEnd < $contentLen)
} else={:log info “Address list update failed - unable to
download list”}
```

Practical Uses

Example 2:

Dynamically importing and updating spam filter / adware address lists for blocking.

Components:

1. Fetch file from predefined URL
2. Clear all entries from old address list
3. Load line of the file as a new address list entry
4. Continue until reaching end of file

Bright WiFi
Clever customer connections.

Practical Uses

Example 3:

Updating time server daily

Script is run by a scheduler entry, will reoccur each day and log an entry if the process fails.

Note the “on-error” catch in case the DNS resolution fails.

Also shown is the importable version you could paste right into terminal to add the scheduler entry and script.

```
:local timeserver
:do {
:set timeserver [:resolve time.windows.com]
} on-error={:set timeserver "failed"}

:if ($timeserver = "failed") do={
:log warning "Warning - unable to check or update time server"
} else={
:log info "Successfully updated time server"
/system ntp client set enabled=yes primary-ntp=$timeserver
}

Importable version:
/system scheduler
add disabled=no interval=1d name=UpdateTimeServer on-event=UpdateTime
start-date=jan/01/1970 start-time=00:00:00
/system script
add name=UpdateTime source=":local timeserver\r\
    \n:do {\r\
    \n:set timeserver [:resolve time.windows.com]\r\
    \n} on-error={:set timeserver \"failed\"}\r\
    \n\r\
    \n:if (\$timeserver = \"failed\") do={\r\
    \n:log warning \"Warning - unable to check or update time server\"\r\
    \n} else={\r\
    \n:log info \"Successfully updated time server\"\r\
    \n/system ntp client set enabled=yes primary-ntp=\$timeserver\r\
    \n}”
```

```
:local counter
:local datadown
:local username
:local macaddr
/ip hotspot
:foreach counter in=[active find ] do={
:set datadown [active get $counter bytes-out]
:if ($datadown>50000000) do={
:set username [active get $counter user]
:set macaddr [active get $counter mac-address]
user remove [find where name=$username]
user add name=$username limit-bytes-out=50000000 mac-add=$macaddr
active remove $counter
:log info "Logged out $username - Reached 50MB download quota"
}}

Scheduled to run every 24 hours:
/ip hotspot
:foreach counter in=[user find ] do={user remove $counter}
```

Practical Uses

Example 4:

Blocks trial hotspot users after they reach a data limit.

This feature does not exist in the standard hotspot trial user options!

Schedule to run every minute, and clear out all user accounts at midnight or every 24 hours.

Bright WiFi
Clever customer connections.

Troubleshooting

Errors and Gotchas

Running configs for interfaces or items that don’t exist (wireless interfaces)

Handling failed DNS resolutions
Handling duplicate entries (no duplicate IP’s allowed in address-list)

Make liberal use of :global or :put commands for testing new scripts to review variables in play

Comment out things you’re not sure of, minimise the testing area.

Ask the room

1. What problems have you solved with scripts?

2. What's something you'd like to solve?

3. What's something you're workng on?

Raise your hand if you'd like to contribute to one of these questions

Questions?

Bright WiFi
Clever customer connections.

Bright WiFi

Clever customer connections.