MikroTik Scripting Cheat Sheet: MikroTikScripting.com

Hello World

```
# file: hello.rsc
# Hello World Script
:local Message "Hello World!";
:put $Message;

> /import hello.rsc
Hello World
```

Local Variables

```
# Create local variable with
# assigned value
:local BookTitle "Jaws";

# Print variable value
:put $BookTitle;

# update variable value
:set BookTitle "War and Peace";

# Clear value
:set BookTitle;

# Note: local vars only exist for
# lifetime of a script.
```

Global Variables

```
# Create global variable with
# assigned value
:global NumPages 278;

# Print variable value
:put $NumPages;

# update variable value
:set NumPages ($NumPages + 2);

# Clear value
:set NumPages;

# Note: global vars exist until
# cleared and can be shared
# between scripts

# print out current global vars
/environment print
```

Key Topic: Variable Names

```
# suggested convention:

:local DayOfWeek "Saturday";
:local MULTIPLIER 10; # constant
:local PingFunc; # function
```

Loops

```
# "for" loop
:for LoopCount from=1 to=10 do={
    :put "Current loop count = $LoopCount";
    :delay 1;
}
```

```
# "foreach" loop
:local MyFruit {"apples"; "plums"; "pears"};

# print out each fruit
:foreach Fruit in=$MyFruit do={
    :put "I like $Fruit as a snack";
}
```

```
# "while" loop
:local TimeNow [/system clock get time];
:local EndTime ($TimeNow + 00:00:10);

:put "Starting 10 sec delay timer…";

:while ([/system clock get time] < $EndTime) do={
  :delay 1;
}

:put "Timer expired!";
```

```
# "do-while" loop
:local Temperature 15;
:do {
    :put "The temperature is: $Temperature ";
} while= ( $Temperature < 12);

# Note: loop executed once even though initial
# condition is false
```

If Statements

```
# "if" statement
:local TimeNow [/system clock get time];

# additional greeting if before noon
:if ($TimeNow < 12:00:00) do={
    :put "Good morning.";
}

:put "Have a good day!";
```

```
# "if-else" statement
:local TimeNow [/system clock get time];

:if ($TimeNow < 12:00:00) do={
    :put "Good morning.";
} else={
    :put "Good afternoon/evening."
}

:put "Have a good day!";
```

Key Topic: Code Comments

```
# single line comment (ends at end-of-line)

# comments cannot span multiple lines
# therefore multiple comments are required for
# a comment block

:put "hello"; # inline comments are useful too
```

Strings

```
# Concatenation
:local Food "Chips";
:local Meal ("Fish and " . $Food);
:put $Meal;

# Substitution
:local NiceFood "Pizza";
:put "I love $NiceFood";


# find & extract subs-string
:local InterfaceName "ether1-WAN";
:local DashLoc [:find $InterfaceName "-"];
:put [:pick $InterfaceName 0 $DashLoc];
```

Arrays

```
# simple list
:local Fruit { "apple"; "orange"; "pear" };

# key/pair array
:local Interface { name="ether"; speed="1gbps" };

# concatenate arrays
:local Array1 { 10; 20; 30; };
:local Array2 { 40; 50; 60 };
:local BigArray ( $Array1, $Array2 );

# access array elements (0 = first element)
:local FirstFruit ($Fruit->0);
:local InterfaceName ($Interface->"name");

# empty array
:local EmptyArray [:toarray ""];
```

Key Topic: Variable Scopes

- **Global Scope**: exists between start and end of every script (can be only one global scope per script).
- **Local Scope**: unique local scope exists between each pair of curly braces. Multiple local scopes may exist, with child/parent relationships formed by embedded local scopes.

Functions

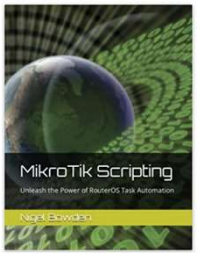
```
# define a function
:global TimeNowFunc do={
    :return [/system clock get time];
}

# print current time
:local CurrentTime [$TimeNowFunc];
:put ("The time now is: $CurrentTime");

# function to calc time
:global TimeInXMins do={
    :local TimeInterval [:totime "00:$1:00"];
    :local TimeNow [/system clock get time];

    return ($TimeNow + $TimeInterval);
}

:local Interval 10;
:put ("The time in $Interval mins is " .
[$TimeInXMins $Interval]);
```

Operators

```
# command operator []
:local Ver [/system resource get version];

# substitution $
:put "99 + 100 is $(99+100)";
:put "the time is $[/system clock get time]";

# grouping operator ()
:put ((10 + 3) * 10);

# regex ~
:put [/interface find where name ~ "^eth"];

# Addition (std math)
:put (99 + 1);

# Addition (IP math)
:put ("300th host: " . (192.168.0.0 + 300));

# Division (always integer result)
:put (5 / 2);

# In (host add in IP network)
:put (10.6.0.1 in 10.0.0.0/12);

# Bitwise and (&)
:put ("Network is : " . 192.168.7.55 & \
    255.255.254.0);
```

The MikroTik Scripting book is available on Amazon.

Find out more at: MikroTikScripting.com



V1.02 (Latest version? Check at https://MikroTikScripting.com/cheatsheet)

Global Commands

```
:beep <freq> <length>
:delay <time>
:do { <commands> } on-error={ <commands> }
:do { <commands> } while=( <conditions> )
:environment print
:error <output>
:execute <expr> [file=<name>]
:execute script=<name>
:find <str> <sub-str> <start>
:for <var> from=<int> to=<int> step=<int>
 do={ <commands> }
:foreach <var> in=<array> do={ <commands> }
:global <var> [<value>]
:if (<condition>) do={<commands>}
:if (<condition>) do={<commands>}
else={<commands>}
:len <expression>
:local <var> [<value>]
:log <topic> <message>
:parse <expression>
:pick <str> <start>[<end>]
:pick <array> <start>[<end>]
:return <value>
:resolve <arg>
:retry command=<expr> delay=[num] \
 max=[num] on-error=<expr>
:rndnum from=[num] to=[num]
:rndstr from=[str] to=[num]
:set <var> [<value>]
:terminal <operation>
:time <expression>
:timestamp
:toarray <var>
:tobool <var>
:toid <var>
:toip <var>
:toip6 <var>
:tonum <var>
:tostr <var>
:totime <var>
:typeof <var>
:while ( <condition> ) do={ <commands> }
```

Data Types

array	A sequence (list) of values, e.g. { "apple"; "orange"; 5 }
bool	A true or false value (Boolean)
id	A unique hexadecimal value assigned to each config item. Also called an internal ID. e.g. *100
ip	An IPv4 address. e.g. 192.168.1.1
ip6	An IPv6 address. e.g. FE80::260:3EFF:FE21:D370
ip6-prefix	An IPv6 prefix e.g. 2001:DB8:0:1::/64
ip-prefix	An IP prefix, e.g. 192.168.2.0/24
nil	Returned by some commands to indicate an empty result
nothing	If a variable has no value, then it returns a "nothing" value
num	A 64bit signed integer, e.g. 100, -5, 3000
str	A sequence of alphanumeric characters. e.g. "ether1-WAN1"
time	A time value. e.g. 18:48:00 (6:48PM)

Arithmetic Operators

+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulo (division remainder)
-	Negation

Note: results are always an integer

Comparison Operators

<	Less than
>	Greater than
=	Equal to
<=	Less than or equal to
>=	Greater than or equal to
!=	Not equal to

Note: result is always boolean (true / false)

Bitwise Operators

~	Invert all bits
|	Perform binary OR on all bits
^	Perform binary XOR on all bits
&	Perform binary AND on all bits
<<	Left shift binary specified num bits
>>	Right shift binary specified num bits

Logical Operators

!	not
&&, and	and
||, or	or
in	in

Concatenation Operators

"."	Joins two strings
","	Joins two arrays or adds an element to an array

Miscellaneous Operators

[]	Returns output of single command, for use in expressions or cmd substitution
()	Returns output of a grouped operation, for use in expressions or cmd substitution
$	Returns the value of a variable or expression. Used in substitution
~	A matching operator that uses POSIX extended regular expression matching
->	Retrieves an array element

Regex Metacharacters

.	Match single character
[]	Match chars in braces
[^]	Match chars NOT in braces
|	Add OR logic to pattern
^	Match beginning of string
$	Match end of string
+	Match one or more instances of char
*	Match zero or more instances of char
*	Match zero or one instances of char

Escape Sequences

\"	Literal quote character
\\	Literal backslash
\n	Newline
\r	Carriage return
\t	Horizontal tab
\$	Literal $ (prevents var substitution)
\?	Literal ? (prevents help system trigger)
_	Insert space character
\a	BEL (0x07) character
\b	Backspace (0x07) character
\f	Form feed (0xFF) character
\v	Vertical tab
\xx	Character with ascii hex value xx