

Introduction

Welcome to "MikroTik Scripting"!

In this book, we'll look at how you can use the power of MikroTik's RouterOS built-in scripting language to extend the functionality and ease the administration of your MikroTik device. We'll look at how we can use scripting to:

- Automate repetitive configuration tasks
- Execute administrative tasks at timed intervals
- Create new functionality customised for your environment
- Interrogate and log performance and status data

In this book, I'll start with the basics of scripting and progress through an extensive range of topics that will allow us to build, debug and run scripts on our MikroTik device. We'll explore questions such as:

- What is scripting, and how can it help me?
- How do I create and run scripts on a MikroTik device?
- What script creation workflow should I use?
- How can I debug my scripts?
- What commands and syntax can I use to create scripts?
- What other resources are available to help me with my scripts?

As we move through each topic, we'll create a series of scripts that increase in sophistication to reflect your growing scripting knowledge as you move through this book. All scripts in this book are available on GitHub. You'll be able to download a copy of each script and run it for yourself to see first-hand how it works.

Creating scripts involves authoring small chunks of code using the MikroTik scripting language. Although this book isn't intended to be a guide to teach you how to become a software developer, I'll try to point out some of the basic concepts you may need if you're not familiar with developing code. I won't be able to turn you into a full-blown software developer, but we'll look at the coding basics required to create your own scripts.

Who is this book for?

Anyone who wishes to learn how to create scripts for the Mikroik platform should find this book helpful. I anticipate that this will include the following:

- Network engineers who are familiar with deploying MikroTik hardware and wish to create their own custom features for their network environment.

- Network and software engineers who are involved with network automation and would like to automate tasks in their environment.
- Service providers with significant deployments of MikroTik equipment who need to provide additional functionality or perhaps gather performance data.
- Network operations departments who would like to automate alerts using self-monitoring scripts on MikroTik devices.

I'm sure there are many other use cases besides those listed above, but these examples hopefully give you some ideas of how scripts could be useful for you.

What do I need to know to use this book?

Although I intend to provide you with a solid introduction to scripting for MikroTik devices, there are some areas I do not have time to explore.

To get the most out of this book, you'll need at least some familiarity with the configuration and operation of MikroTik devices. It's not too important which device type (or types) you've used, but hopefully, you'll know a little about RouterOS and Winbox and have entered at least a few commands on a MikroTik CLI (Command Line Interface).

I'll also assume that you know networking basics and understand the fundamentals of IP addressing. You're hopefully familiar with terms such as SSH, HTTPS, Ping and SFTP.

What is a script?

I've talked about creating "scripts" without providing a solid definition of what I mean by a "script". I assume that many people picking up this book probably have some notion of what scripting is, but for the avoidance of doubt, let's define it here.

A script is a sequence of commands (usually stored in a file) executed to perform one or more desired operations on a piece of MikroTik hardware. They're written using commands that are defined in the MikroTik RouterOS operating system.

Each script is a simple text file. If the correct command syntax is used (i.e. correctly structured commands and statements), a script may be created using any simple text editor such as Notepad on Windows, Nano on Linux or TextEdit on macOS. Note that these are just a few examples of editors that may be used to create a script. Many other, more sophisticated editors are available and can provide advanced features such as syntax highlighting that may highlight potential syntactical errors in a script. We'll explore editors later in this book.

A script may be a simple sequence of just a few commands or maybe many hundreds of lines long to perform sophisticated operations.

A script may be scheduled to run regularly (e.g. every hour) and perform a repetitive action. For example, a script may be created to check and report the response time to a web server every 5 minutes.

Scripts may also be run on an ad-hoc basis to perform a one-time operation. A good example is a script that creates a custom device configuration from a pre-defined template of RouterOS commands.

Scripts can be extremely sophisticated, allowing them to make conditional decisions based on detected triggers (e.g. send an email alert if a WAN link failure is detected).

Each script is a self-contained software program. Creating scripts can be thought of as a small-scale software development process. You don't need to be a software developer to create scripts, but you will need to learn some of the base principles that they employ when developing your own code.

Scripts are *generally* quick, fun and easy to develop. They have few of the complexities associated with large software development projects, so are far more accessible for those of us from a more traditional network engineering and administration background.

Can I write MikroTik scripts if I'm not a developer?

Yes! If you've ever copied and pasted, or typed a sequence of commands on the CLI of a MikroTik, you've already effectively executed a script. The only difference is that you typed the commands line-by-line instead of saving them to a file for execution as a code block.


In my opinion, the best approach to learning to create scripts is to follow the examples and concepts presented in this book sequentially: we will start with simple scripts and add more features and functionality as we progress. As you learn more about the scripting language and grow in confidence, you can gradually increase the size and complexity of your scripts to create the solutions you need.

Before moving on, take time to understand each script and the concepts we explore in detail. This will ensure you will be far more effective in creating your own scripts when you're ready to strike out on your own. Although you will find plenty of script examples online, taking the time to gain an understanding of how the scripting language works, as opposed to just copying and pasting examples you find, will pay great dividends. There is nothing wrong with using open-source examples that others have shared, but taking the time to gain a solid understanding of the scripting language will ensure you can easily use and modify them for your own purposes.

Conventions Used in This Book

Before we get into the fun of creating our MikroTik scripts, here are a few conventions to watch out for in this book.


You'll sometimes see an information panel like the example shown below. This will contain particularly useful or important information about the subject being discussed. These will be used for time-saving tips or to emphasise core concepts that need your full attention.



Tip:

This is essential information that you'll be very interested in reading!

Sometimes, there will be instances of specific "gotchas" or particularly critical pieces of information that need your attention. These will be highlighted in warning dialogue panels, as shown below.



Warning:

This is a warning that could save you quite a bit of trouble, so it is well worth reading carefully :-)

We'll often be looking at scripting examples throughout this book. These will range from short snippets of example code to more fully featured scripts that we develop as we progress through this book. All code will be presented in the format shown below. Code blocks will be delimited using horizontal rules above and below the code. The code itself will be displayed in a terminal-type font:

```
# Extract MAC address of first interface
:local firstEtherInterface ([/interface find type=ether]->0)
:local macAddress [/interface get $firstEtherInterface mac-address]

# system identity (hostname)
:local hostname [/system identity get name];
```

Code Examples

Throughout this book, I'll provide examples of code that will demonstrate various aspects of the RouterOS scripting language. All code is available for you to download from the following GitHub site:

<https://github.com/wifinigel/MikrotikScripting>

At the time of writing, RouterOS v6 probably accounts for a large proportion of the population of current MikroTik devices out in the "real world". RouterOS v7 is being supplied on many new devices and will likely be the version of choice going forwards. The examples in this book are based on RouterOS v6. These *should* run without issue on a v7 device, as most commands appear backwards compatible.

Most of the code used in this book focuses on global commands, which change very little between versions 6 and 7. (See *chapter 5 for a discussion of different command types*.) Most of the significant changes between the two RouterOS versions are found in the configuration commands. Very few configuration commands are used in the examples provided, so there will be few differences to worry about.

On the GitHub site shown at the start of this sub-section, I have provided both v6 and v7 versions of the script examples shown in this book. In reality, there is very little difference between the two sets of scripts. You can download the version that suits your environment.

Screenshots

The book includes a number of screenshot images to demonstrate the expected output for various software packages. These may include highlighted areas to bring attention to import segments of the screenshot. I have referenced "highlighted areas" in the text that accompanies each screenshot. As this book may be produced in several formats, the highlighted areas may not always be obvious, particularly in black and white print formats. Therefore, I have included all screenshot images in the same GitHub code repository used for the code examples from this book. Please take a look at these if you have difficulty understanding which areas are highlighted or have difficulty viewing the detail of any of the screenshots. The images may be viewed at:

<https://github.com/wifinigel/MikrotikScripting>

Website

I have created a website to accompany this book. It contains links to all resources related to this book, together with any errata items that are reported to me. It will also contain information about any additional content I make available to supplement the book content (e.g. blog posts, videos, etc.). The website can be found at:

<https://mikrotikscripting.com>

MikroTik Scripting Language

It's worth spending a few moments talking about the MikroTik scripting language. If you've used the command line interface of a MikroTik device, you'll have seen a variety of commands. These can be used to configure a MikroTik device to provide the features needed on your network or retrieve network data.

For example, maybe we'd like to check if we have a connection to the Internet across the WAN port of our MikroTik router. We can run the command shown below and view its output. If the ping responses look OK, we can be confident that our users are probably getting an Internet service.

```
[admin@mikrotik > /ping 8.8.8.8 count=3 interface=ether1-WAN
```

We can take CLI commands like the one shown above and include them in our scripts. However, we can also perform additional, sophisticated operations by adding a few of the many RouterOS scripting commands available.

For example, suppose we wanted to regularly run the CLI command shown above from the router scheduler. In that case, we could create a script and add additional scripting logic to place an error message in the MikroTik error log whenever the ping fails. This indicates when our Internet service is probably down:

```
# (comment) check how many ping responses we get
:local pingCount [/ping 8.8.8.8 count=3 interface=ether1-WAN]

# If all ping attempts fail, let's log an error message
if ( $pingCount = 0 ) do={
    :log error "WAN link is down!!!"
}

```

This is a simple example, but hopefully demonstrates how we can re-use CLI commands we already know to craft useful new functions once we mix in some scripting commands and logic.

Do not worry about understanding what each of the lines of code above actually does. This is just a taster of what you can achieve once you immerse yourself in the world of MikroTik scripting.