Chapter 2 – Hello World

In this chapter, we'll examine the basics of running a script on your MikroTik device.

If you've studied any other programming or scripting language, you'll be familiar with the concept of a "hello world" program. It's a piece of minimal code traditionally used to show how to run a basic program that prints the phrase "hello world". This provides a simple starting point to show how to perform a basic operation using the language. We'll take that same approach here, but we'll log the output phrase to the MikroTik logging system to make things a little more interesting.

Two steps are required to run a script on your MikroTik device (e.g. router):

- Store the script code on the device
- Run the code on the device

There are several options for each of these steps. We'll take a detailed look at each in this chapter.

In summary, your code can be stored on the MikroTik device using the following methods:

- As a file on the MikroTik's file system
- In the MikroTik's script repository

Once stored on the MikroTik, the options available to execute the code are:

- Via CLI commands
- Regular and/or scheduled operation via the MikroTik scheduler
- Using the Netwatch feature in response to a network event

Hello World Script

When learning programming or scripting languages, there is a tradition of starting the exploration of that language with a simple "hello world" code example. It demonstrates a simple example of how to make the language do something for you. It shows the steps required to create and run a simple program, or in our case, a script.

There are several ways we may wish to execute a script on a MikroTik device, so we'll use the same "`hello world`" example to demonstrate the options available.

This may seem a trivial example when you're probably keen to move on to see scripts that can perform cool networking functions. However, the focus here is to understand the options available to run your scripts rather than getting down to the business of writing complex scripts.

The script code we'll be using is as follows:

```
# filename: ch2-01-hello-world.rsc
# Hello World Script
:local Message "Hello World!";
:log info $Message;
```

Let's do a quick line-by-line analysis of this code to give an overview of what's happening here. Don't worry about understanding everything that's going on in this script; we'll cover it all in more detail later:

```
# filename: ch2-01-hello-world.rsc
```

This is a comment that shows the name of this file. Any line that begins with a "#" symbol is considered a comment. Commented lines in the script are ignored when the script is executed. They are provided to document the operation of the script. You could remove them without impacting the script's operation, though this is a bad idea from a documentation perspective.

You wouldn't usually include this line indicating the filename in your production code. I've included it as a useful for reference so that you can find the file by its name in the book's GitHub repository that contains all sample code (https://github.com/wifinigel/MikrotikScripting). This convention is repeated for all other scripts in this book.

```
# Hello World Script
```

This is another comment. It briefly describes what the script does.

```
:local Message "Hello World!";
```


Next, we assign the *string* "`Hello World!`" to a *variable* called "`Message`". A string is a collection of characters enclosed in speech marks.

We can reuse the string of characters elsewhere in our script by assigning it to a variable. This is extremely useful if we need to use the same string multiple times in our script. We only use the string once in this case, so it's not particularly useful. You'll use variables many times as you progress through this book and come to understand their value in your coding.

In RouterOS, there are two types we can use when creating a variable: "global" and "local". In our example, we've created the "`Message`" variable as a local variable using the "`:local`" keyword. Don't worry too much about variable types for now. We'll cover them in detail very soon. For now, just know that we have declared a local variable.

```
:log info $Message;
```

The final line of the script uses the "`:log`" keyword to send the contents of the "`Message`" variable to the system log. Each time the script is run, the "`Hello World`" message will be shown in the system log using the "informational" logging level.

Note that when a variable has been created, and we wish to use it later to retrieve its contents, we have to prefix it with the "$" character. As you can see in our example, to get back the contents of the "`Message`" variable, we use the format: `$Message`.

Ok, enough of the code analysis. Don't worry if it doesn't make sense yet; we'll return to these topics very soon and explore them in much more detail.

Running a Script

As briefly mentioned in chapter 1, several methods are available to run a script on a MikroTik device. Once we've completed the development cycle detailed in chapter 1, we'll likely want to ensure the script runs regularly or is triggered by an event. In some instances, we may have created an ad-hoc script to manually run on the CLI.

The process and options for running a script are outlined in the chart below: