Constants

Overview

A constant is like a variable, except that its value **cannot change** throughout the runtime of the program.

DEFINE

In DuckyScript, a constant is initialized using the `DEFINE` command. One may consider the use of a `DEFINE` within a payload **like a find-and-replace** just before the time of compile - within what is called the preprocessor.

`DEFINE` can be used to more easily expose or abstract configuration options used throughout your payload. This means to change a constant value that is described by a `DEFINE` you only need to change it in one location no matter how many times its used throughout your payload.

Syntax

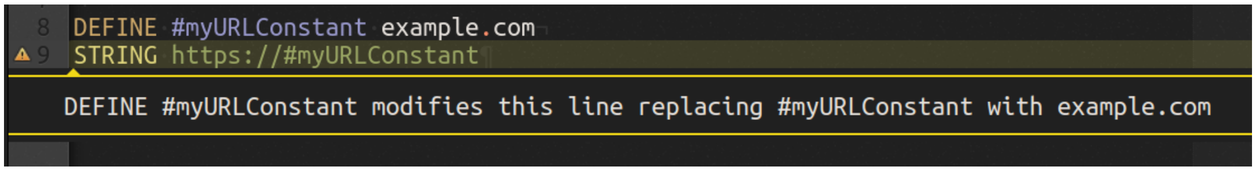
```
DEFINE LABEL VALUE
```

1. `DEFINE` denotes the start of a constant definition
2. `LABEL` is the label or key to be used by the compiler to locate usage within your payload
3. `VALUE` is the value to replace matching instances of `LABEL` throughout your payload. The `VALUE` is everything past `LABEL` to the end of the line (minus the first space).

With this in mind its best to keep your `LABEL` as descriptive as possible. Remember - **it will be replaced with the given** `VALUE` - the length of the `LABEL` will have no affect on the actual length of your compiled payload.

Within PayloadStudio

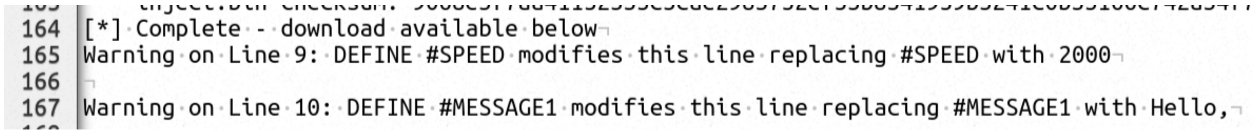
PayloadStudio takes the guess work out of what will get replaced where by automatically annotating lines that are modified by `DEFINE` statements throughout your payload.



```
DEFINE #myURLConstant example.com
STRING https://#myURLConstant

DEFINE #myURLConstant modifies this line replacing #myURLConstant with example.com
```

This also gives you the chance to spot any misconfigurations when compiling your payload as PayloadStudio will list these in the console upon generating your `inject.bin`



```
[*] Complete - download available below
Warning on Line 9: DEFINE #SPEED modifies this line replacing #SPEED with 2000

Warning on Line 10: DEFINE #MESSAGE1 modifies this line replacing #MESSAGE1 with Hello,
```

Labels

Depending on the format of your `LABEL`, `DEFINE` will behave differently in it's find-and-replace method. This is to significantly reduce the likelihood that your `DEFINE` statement has negative unintended side-affects.

With `#`

```
DEFINE #myConstant TEST
```

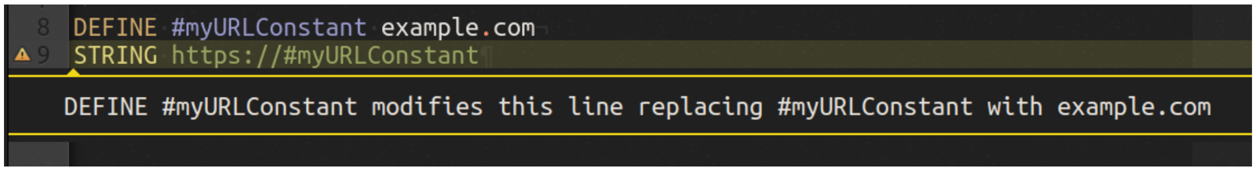
Using this syntax,

```
#myURLConstant
```

will be replaced anywhere within your payload **even if it is touching other characters.** This is because the `LABEL` starts with `#`

```
DEFINE #myURLConstant example.com
STRING https://www.#myURLConstant
```

This will result in `https://www.example.com` because `#myURLConstant` starts with a `#`


```
DEFINE #myURLConstant example.com
STRING https://#myURLConstant
```
DEFINE #myURLConstant modifies this line replacing #myURLConstant with example.com

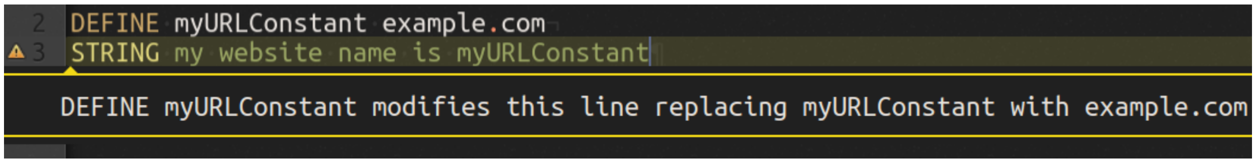
Without `#`

`DEFINE myURLConstant TEST`

Using this syntax, `myURLConstant` will be replaced **anywhere it is not touching other characters (is its own individual word)** within your payload. This is because the `LABEL` does not start with `#`

```
DEFINE myURLConstant example.com
STRING my website name is myURLConstant
```

this will result in `my website name is example.com`


```
DEFINE myURLConstant example.com
STRING my website name is myURLConstant
```
DEFINE myURLConstant modifies this line replacing myURLConstant with example.com

```
While this method is still supported, it is no longer best pract

Usage of a given LABEL becomes very hard to spot mid-payload mal

Consider the following example:
DEFINE test 123
STRING This is a test showing the ambiguity

Result:
This is a 123 showing the ambiguity

The instance of test in the above STRING will be replaced but it
```

Examples

Example as Boolean

```
REM Example Boolean
DEFINE #BLINK_ON_FINISH TRUE
```

DuckyScript developers may find it useful to include defines at the top of their payload which determine whether or not a function will run. This makes it easier for the end-user to customize a shared payload.

Example as Integer

```
REM Integer
DEFINE #DELAY_SPEED 2000
```

In this example, one may imagine the `DELAY_SPEED` constant will be used in conjunction with one or more `DELAY` commands.

Example as `STRING`

```
DEFINE #MESSAGE example.com
STRING https://
STRING #MESSAGE
```

```
DEFINE #MESSAGE example.com
STRING https://#MESSAGE
```

In both cases this will result in "`https://example.com`" being typed because the label used starts with a `#` See above

Example Payload

```
REM Example constants using DEFINE

ATTACKMODE HID STORAGE
```

```
DEFINE #SPEED 2000
DEFINE #MESSAGE1 Hello,
DEFINE #MESSAGE2 World! Written with a define!

DELAY #SPEED
STRING #MESSAGE1
DELAY #SPEED
SPACE
STRING #MESSAGE2
```

Result

The payload will begin with a 2 second delay, then type "`Hello, World! Written with a define!`" with a 2 second delay in between `#MESSAGE1` and `#MESSAGE2`.

Changing the string values of `#MESSAGE1` and `#MESSAGE2` will change the outcome of the payload.

Changing the integer value of `#SPEED` will change the delay between the first and second message.

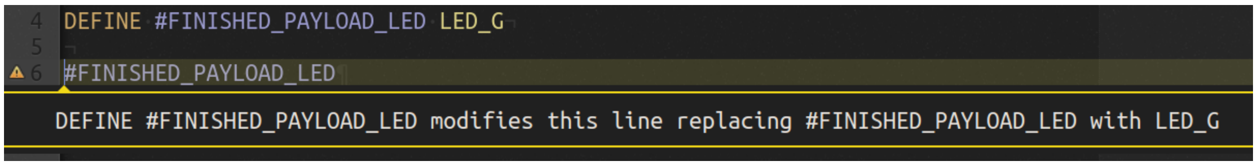
Advanced Example

Considering `DEFINE` is a effectively an automatic find-and-replace step prior to compile, the `VALUE` of a `DEFINE` is not limited to any specific datatypes. Any valid DuckyScript syntax can be the `VALUE` of a `DEFINE`

```
DEFINE #FINISHED_PAYLOAD_LED LED_G

...Payload...

#FINISHED_PAYLOAD_LED
```



```
DEFINE #FINISHED_PAYLOAD_LED LED_G

#FINISHED_PAYLOAD_LED
```

```
DEFINE #FINISHED_PAYLOAD_LED modifies this line replacing #FINISHED_PAYLOAD_LED with LED_G
```

Best Practices

Configurable payload options should be specified in variables or defines at the top of the payload.

Define labels should start with
`#` for easy identification throughout your payload.

When writing a payload that calls external resources which may vary depending on the operator, such as a website to open or address to establish a reverse shell with, it is best to use `DEFINE`.

In addition to comment blocks (like the `REM` title/author/description lines in the above example), putting your `DEFINE` commands at the top of your payload makes it easier for someone else to use your payload effectively. Even more so if the constants are commented!

Avoiding Errors

Internal variables begin with an underscore, so it is best practice to avoid this style.

Spaces cannot be used in naming a constant — however underscore makes for a suitable replacement. For example: `DEFINE #REMOTE_HOST 192.168.1.100`.

Labels should descriptive. For example, `#RHST` is better than `#R`, and `#REMOTE_HOST` is better than `#RHOST`.

Be careful when using the uppercase letter `O` or lowercase letter `l` as they may be confused with the numbers `0` and `1`.

Avoid using the names of commands or internal variables (e.g. `ATTACKMODE`, `STRING`, `WINDOWS`, `MAC`, `$_BUTTON_ENABLED`). See the full command and variable reference.

Invalid Usages

```
DEFINE myURLConstant example.com
STRING https://www.myURLConstant
```

This will result in `https://www.myURLConstant` because `myURLConstant` was **not its own word and does not start with** `#`

```
DEFINE #TEST 123
DEFINE #TEST2 #TEST
```

This **will not** replace the value of `#TEST2` with `123`