OpenBullet LoliScript (LS) Guide [From XML To PDF]

General Info

What is LoliScript?

LoliScript (LS) is a custom scripting language made to give access to all existing (plus some more advanced) capabilities of OpenBullet.

Every instruction is a one-liner (but can be broken down into multiple segments for readability) and it will integrate the existing block execution capabilities with some other that cannot be reproduced with normal blocks.

This is ABSOLUTELY NOT a fully functional scripting language meant to replace the capabilities offered by JavaScript or Python.

Here's an example of a LoliScript command:

```
PRINT Hello, World!
```

And here's how you would execute a standard Function Block:

```
FUNCTION Constant "Hello, World!" -> VAR "GREETING"
```

In this documentation you will learn how to use LoliScript to its full potential and integrate it with your other blocks or even stop using blocks altogether!

When to use it?

You should use LoliScript when:

You want something faster and much easier to debug at a glance Parameter (usually identifies an enumerator value)

```
FUNCTION Hash SHA512 ...
```

Boolean (a parameter name (case sensitive!) followed by =True or =False)

```
KEYCHECK BanOn4XX=TRUE
```

Redirector (used to direct the output to a variable or a file)

```
FUNCTION Constant "Hello" -> VAR "GREETING"
FUNCTION Constant "Hello" -> CAP "Greeting"
```

Variables

You can access the saved values of variables and use them in literals by including them in angle brackets like this A.

There are 3 types of variables:

- **Single**
- **List**
- **Dictionary**

Examples (NAME is the variable name):

- Single

```
<NAME>
```

- List (access one element by index or all element using * as index)

```
<NAME[0]>
<NAME[*]>
```

- Dictionary (access by key using () and by value using {}, supports *)

```
<NAME(key)>
<NAME{value}>
<NAME(*)>
<NAME{*}>
```

Comments

Every line that starts with two #s is considered a comment

Example:

```
## THIS IS A COMMENT
```

General Commands

PRINT

This command is peculiar since it doesn't take a literal argument but just parses the rest of the line after the keyword `PRINT` and outputs it to the debugger log.

Syntax:

```
PRINT TEXT
```

Example:

```
PRINT Hello, World!
```

SET

This command is very powerful as it allows manipulation on the global data of the bot.

Syntax:

```
SET IDENTIFIER [PARAMETERS]
```

Allowed identifiers:

SOURCE – The stored response source of the last request performed

STATUS – The status of the bot

RESPONSECODE – The response code of the last request performed

COOKIE – Sets the value for a cookie of the global cookie jar given its name

ADDRESS – The address of the response of the last request performed

USEPROXY – Whether to use the proxy that is currently assigned to the bot during requests

PROXY – Sets the current proxy

PROXYTYPE – Sets the current proxy type

DATA – Sets the current data

VAR – Sets a variable

CAP – Sets a captured data

NEWGVAR – Initializes a new global variable (only if it doesn't exist)

GVAR – Sets a global variable's value

GCOOKIES – Sets cookies from the local cookie jar into the global cookie jar

Examples:

```
SET SOURCE "Example Source"
SET STATUS SUCCESS
SET STATUS CUSTOM "ABCD"
SET RESPONSECODE 200
SET COOKIE "token" "abcdef"
SET ADDRESS "https://www.google.com"
SET USEPROXY FALSE
SET PROXY "127.0.0.1:8888"
SET PROXYTYPE HTTP
SET DATA "abc:def"
```

```
SET VAR "varname" "data"
SET CAP "capname" "data"
SET NEWGVAR "sharedcookie" "abc=123"
SET GVAR "sharedcookie" "def=456"
SET GCOOKIES
```

DELETE

This command allows you to delete elements from current collections.

Syntax:

```
DELETE IDENTIFIER [CONDITION] "LITERAL"
```

Allowed identifiers:

- COOKIE
- VAR
- GVAR

The default condition is EqualTo

```
Examples:
DELETE COOKIE Contains "phpsessid"
DELETE VAR "TOKEN"
DELETE GVAR "MAX"
```

Mouse Action

This command allows you to perform advanced mouse actions in the browser.

General Syntax:

```
MOUSEACTION [ACTION [PARAMETERS]]*
```

General Syntax for an ELEMENT parameter:

```
ELEMENT LOCATOR "STRING" [INDEX]
```

https://t.me/viperzcrew

Allowed Actions:

```
## Spawn a div with a given id to a (x,y) absolute position, so you can use it as element hook for mouse movement
SPAWN "ID" X Y
CLICK [ELEMENT]
CLICKANDHOLD [ELEMENT]
RIGHTCLICK [ELEMENT]
DOUBLECLICK [ELEMENT]
DRAGANDDROP ELEMENT -> ELEMENT
DRAGANDDROPWITHOFFSET X Y ELEMENT
KEYDOWN "KEY" [ELEMENT]
KEYUP "KEY" [ELEMENT]
MOVEBY X Y
MOVETO ELEMENT
RELEASE [ELEMENT]
SENDKEYS "KEYS" [ELEMENT]

## Draws random points in a square starting from the top left corner and with the given width and height
DRAWPOINTS MAXW MAXH AMOUNT
DRAWLINE (X1 Y2 -> X2 Y2 / ELEMENT -> ELEMENT) : AMOUNT
DRAWLINEHUMAN (X1 Y1 -> X2 Y2 / ELEMENT -> ELEMENT) : AMOUNT [GRAVITY WIND]
```

Examples:

```
## Moving to an element and clicking 100 pixels right of it
MOUSEACTION
MOVETO ELEMENT CLASS "testclass" 1
MOVEBY 100 0
CLICK

## Moving to a specific set of coordinates on the page
MOUSEACTION
SPAWN "target" 100 150
MOVETO ELEMENT ID "target"

## Moving like a human from point A (0,0) to point B (700,700). The lower the AMOUNT, the faster the line is drawn
MOUSEACTION
DRAWLINEHUMAN 0 0 -> 700 700 : 150 1 1
```


https://t.me/viperzcrew

```
## Moving like a human from element with id="one" to element with
id="two", 20 points and default gravity/wind
MOUSEACTION
DRAWLINEHUMAN ELEMENT ID "one" -> ELEMENT ID "two" : 20
```

Block Commands

Info

The block commands will spawn and execute a block of the corresponding type, after setting all the required parameters.

You can give a label to any block by writing # and then the preferred label, for example:

```
#GETLEN FUNCTION Length "Test123" -> VAR "LEN"
```

Function

This will spawn and execute a Function Block.

Note: If the input has a variable of List type, the function will be executed on all the elements and return a new List variable.

Syntax:

```
FUNCTION Name [ARGUMENTS] ["INPUT STRING"] [-> VAR/CAP "NAME"]
```

Examples:

```
FUNCTION Constant "Hello" -> VAR "GREETING"
FUNCTION Constant "Name is <NAME[*]>" -> CAP "Names"
FUNCTION ClearCookies
```

Peculiar functions:

```
FUNCTION Hash SHA512 "<PASS>" -> VAR "HASHED"
FUNCTION HMAC SHA1 "key" HmacBase64=True "<PASS>" -> VAR "DIGEST"
FUNCTION Translate StopAfterFirstMatch=True
KEY "key1" VALUE "value1"
KEY "key2" VALUE "value2"
```

```
"input" -> CAP "Translated"
FUNCTION DateToUnixTime "format" "input" -> VAR "DATE"
FUNCTION Replace "what" "with" UseRegex=True "input" -> VAR
"REPLACED"
FUNCTION RegexMatch "pattern" "input" -> VAR "MATCHED"
FUNCTION RandomNum 0 100 -> VAR "RANDOM"
FUNCTION CountOccurrences "tofind" "input" -> VAR "COUNT"
FUNCTION CharAt "index" "input" -> VAR "CHAR"
FUNCTION Substring "startindex" "length" "input" -> VAR
"SUBSTRING"
```

Keycheck

This will spawn and execute a Keycheck Block.

Syntax (expanded for readability):

```
KEYCHECK [BanOn4XX?] [BanOnToCheck?]
[KEYCHAIN TYPE ["CUSTOMNAME"] MODE
[KEY "STRING" [CONDITION "STRING"]]*
]*
```

Note: As shown above, for brevity,
`KEY "<SOURCE>" Contains "abc"`
can be written as
`KEY "abc"`

Example:

```
KEYCHECK BanOnToCheck=False
  KEYCHAIN Success OR
    KEY "Logout"
  KEYCHAIN Failure OR
    KEY "<SOURCE>" EqualTo ""
  KEYCHAIN Custom "DEFAULT" OR
    KEY "abc" Contains "ab"
```

Request

This will spawn and execute a Request Block.

Syntax (expanded for readability):

```
REQUEST METHOD "URL" [AcceptEncoding?] [AutoRedirect?]
[ReadResponseSource?] [ParseQuery?] [EncodeContent?]
    [STANDARD / MULTIPART / BASICAUTH]
    [CONTENT "postdata"]
    [CONTENTTYPE "ctype"]
    [STRINGCONTENT "name: value"]*
    [FILECONTENT "name: path"]*
    [BOUNDARY "abcd"]
    [COOKIE "abc: def"]*
    [HEADER "abc: def"]*
    [-> STRING / -> FILE "path"]
```

Examples:

```
REQUEST GET "http://example.com/file.zip" -> FILE "file.zip"
```

```
REQUEST POST "http://example.com/poster" AutoRedirect=False
  CONTENT "{\"id\":1}"
  CONTENTTYPE "application/json"
  COOKIE "Hello: World"
  HEADER "Hello: World"
```

```
REQUEST POST "http://example.com/multi" Multipart
  STRINGCONTENT "Hello: World"
  STRINGCONTENT "Hi: Again"
  FILECONTENT "Image: image.jpg"
  BOUNDARY "abcd"

REQUEST POST "http://example.com/basic" BasicAuth
   USERNAME "<USER>" PASSWORD "<PASS>"
```


Recaptcha

This will spawn and execute a Recaptcha Block.

Syntax:

```
RECAPTCHA "URL" "SITEKEY" -> VAR "NAME"
```

Example:

```
RECAPTCHA "http://example.com" "ABCD" -> VAR "RECAP"
```

Captcha

This will spawn and execute a Captcha Block.

Syntax:

```
CAPTCHA "URL" [Base64?] [SendScreenshot?] -> VAR "NAME"
```

Example:

```
CAPTCHA "http://example.com/image.png" -> VAR "CAP"
```

Parse

This will spawn and execute a Parse Block.

Syntax:

```
PARSE "TARGET" LR "LEFT" "RIGHT" [Recursive?] [EncodeOutput?]
[CreateEmpty?] [UseRegexLR?] -> VAR/CAP "NAME" ["PREFIX"
"SUFFIX"]

PARSE "TARGET" CSS "SELECTOR" "ATTRIBUTE" [INDEX / Recursive?]
[EncodeOutput?] [CreateEmpty?] -> VAR/CAP "NAME" ["PREFIX"
"SUFFIX"]

PARSE "TARGET" JSON "FIELD" [JTokenParsing?] [Recursive?]
[EncodeOutput?] [CreateEmpty?] -> VAR/CAP "NAME" ["PREFIX"
"SUFFIX"]

PARSE "TARGET" REGEX "REGEX" "OUTPUT" [Recursive?]
[EncodeOutput?] [CreateEmpty?] -> VAR/CAP "NAME" ["PREFIX"
"SUFFIX"]
```


https://t.me/viperzcrew

https://t.me/viperzcrew

Examples:

```
PARSE "<SOURCE>" LR "<span>" "</span>" Recursive=True -> CAP "BALANCE" "$" ""

PARSE "<SOURCE>" CSS "[name=csrf]" "value" -> VAR "CSRF"
```

Bypass CF

This will spawn and execute a BypassCF Block.

Syntax:

```
BYPASSCF "URL" ["UA"]
```

Example:

```
BYPASSCF "http://example.com"
```

Utility

This will spawn and execute a Utility Block.

Syntax for list:

```
UTILITY LIST "List Name" ACTION [PARAMETERS] [-> VAR/CAP "NAME"]
```

Syntax for variable:

```
UTILITY VARIABLE "Variable Name" ACTION [PARAMETERS] [-> VAR/CAP "NAME"]

Syntax for conversion:
UTILITY CONVERSION FROM TO "input" [-> VAR/CAP "NAME"]
```

Syntax for file:

```
UTILITY FILE "File Name" ACTION [PARAMETERS] [-> VAR/CAP "NAME"]
```

Example:

https://t.me/viperzcrew

```
UTILITY LIST "List1" JOIN "," -> VAR "JOINED"

UTILITY VARIABLE "Var1" SPLIT "," -> VAR "SPLITTED"

UTILITY CONVERSION BASE64 HEX "0xA35F" -> VAR "BASE64"

UTILITY FILE "test.txt" WRITE "Hello" -> VAR "TEXT"
```

TCP

This will spawn and execute a TCP block

NOTE: This block is currently only available in LoliScript

Syntax:

```
TCP COMMAND [ARGUMENTS]
```

Allowed Commands:

```
CONNECT "HOST" "PORT" [UseSSL?] [WaitForHello?] [-> VAR/CAP
"NAME"]

SEND "MESSAGE" [WebSocket?] [-> VAR/CAP "NAME"]
DISCONNECT
```

Example:

```
TCP CONNECT "google.com" "443" UseSSL=True
TCP SEND "Hello" -> VAR "RESPONSE"
TCP DISCONNECT
```


Navigate

This will spawn and execute a Navigate Block.

Syntax:

```
NAVIGATE "URL" [TIMEOUT] [BanOnTimeout?]
```

Example:

```
NAVIGATE "http://example.com"

NAVIGATE "http://example.com" 60 BanOnTimeout=True
```

Browser Action

This will spawn and execute a BrowserAction Block.

Syntax:

```
BROWSERACTION ACTION ["INPUT"]
```

Examples:

```
BROWSERACTION OPEN
BROWSERACTION SENDKEYS "Hello"
```

Element Action

This will spawn and execute an ElementAction block

Syntax:

```
ELEMENTACTION LOCATOR "LOCATOR" [INDEX / Recursive?] ACTION
["INPUT"] [-> VAR/CAP "NAME"]
```

Examples:

```
ELEMENTACTION ID "testid" CLICK
ELEMENTACTION SELECTOR "div input" SENDKEYS "abcd"
ELEMENTACTION CLASS "testclass" Recursive=True GETTEXT -> CAP
"ITEMS"
```

Execute JS

This will spawn and execute an ExecuteJS block

Syntax:

```
EXECUTEJS "SCRIPT"
```

Examples:

```
EXECUTEJS "alert('hi');"
```

Flow Control

If Else

The IF statement will check the condition after it. If it's successful, it will continue to the next instruction, otherwise it will jump to the first ELSE or ENDIF statement.

The ELSE statement will be executed if the IF wasn't successful, otherwise it will jump to the first ENDIF statement.

Syntax:

```
IF "STRING1" CONDITION "STRING2"
## OTHER COMMANDS
ENDIF
```

You can add an ELSE command too like this:

```
IF "STRING1" CONDITION "STRING2"
## COMMANDS TO BE EXECUTED ON TRUE CONDITION
ELSE
## COMMANDS TO BE EXECUTED ON FALSE CONDITION
ENDIF
```

While

The WHILE statement will check the condition after it. If it's successful, it will continue to the next instruction, otherwise it will jump to the first ENDWHILE statement.

The ENDWHILE statement will always jump back to the first WHILE statement above it.

The WHILE - ENDWHILE pair allows to reproduce the FOR statement of common high level programming languages as well.

Syntax:

```
WHILE "STRING1" CONDITION "STRING2"
## OTHER COMMANDS
ENDWHILE
```

Jump

The JUMP statement will jump to the first block with the specified label (within the script!).

Syntax:

```
JUMP #LABEL
```

Begin (End) Script

The BEGIN SCRIPT statement will mark an area of the script where the commands will be executed using another scripting language.

The OUTPUTS literal can contain a comma-separated list of variable names that you want to extract from the script's scope and declare inside OB as strings or lists.

It is useful to mention that your normal variables will be automatically declared inside the script as strings / lists of strings / dictionaries (if their name is allowed by that language's standards).

Available languages:

- JavaScript
- IronPython

Syntax:

```
BEGIN SCRIPT LANGUAGE
[script here]
```

```
END SCRIPT -> VARS "OUTPUTS"
```

Example:

```
BEGIN SCRIPT JavaScript
var a = 1+2;
var b = 'hello';
END SCRIPT -> VARS "a,b"
```

THIS DOCUMENT WAS FORMATTED BY @MRBLACKX
IT TOOK 3 DAYS TO FIND CORRECT COLOURS AND FORMAT BOLD UNDERLINE ETC.

OFFICIAL SOURCE OF THIS DOCUMENT IS OPENBULLET