The UBot Studio

SCRIPT REFERENCE

The Browser Functions
The Browser Function menu will be covered in this section. They are located in the Browser Functions menu within the Parameters main menu:

**Solve Captcha**

This function does exactly what it says. It solves captchas based on three integrated captcha solving services (DeathbyCaptcha, Decaptcher and Bypasscaptcha) as well as the manual solving option. The biggest difference in version 4 with solving captchas is it stores the user’s captcha credentials for those services. If no services are stored it uses the manual solving option by default. You begin by storing your service credentials. They can be located in the tools menu under “CAPTCHA Service Accounts”: 
Once you click on the CAPTCHA Service Accounts you will be presented with a new window where you will press the Add button to store your credentials:

Once you press this button you will then choose which service you wish to store your credentials for:

Once you enter your credentials, Ubot will connect with the service provider and display how many credits you have remaining in that account:
You can choose to add all three services, but keep in mind UBot will solve the captcha using the first checkmark it comes across in the list. So if you have Death by Captcha first and it is checked, it will always use that service. If it encounters no services, it will automatically use the manual option.

So on to solving captchas. This is done slightly different than in v3.5. In version 4 you are simply going to drag the captcha answer field into the scripting area:
From here you are going to drag the $solve captcha function into the “Text to Type” box:
Now select the Element Selector (Ξ) and then click on the captcha image in the browser:

Notice it did not provide a suitable attribute in the captcha Element box. Therefore, you can click the advanced element editor (Ξ) and choose another attribute to choose the image by:
In this case we chose src (1) and when it displayed the src we erased the entire token and inserted a wildcard (2). We then chose the wildcard from the dropdown (3) and finally clicked on “Add” (4) and “OK”:

You then want to click on the two remaining “OK” buttons to save and close the node. You are done, and your bot is ready to solve that captcha.

**Page Scrape**

This function returns text scraped from the current page between two specified pieces of text. In order to scrape your data you can either save it to a list or set it to a variable. We will be setting the variable for this example:
We will be scraping the word “Changer” in this example. The next step is to drag the $page scrape function into the New Value box:
In this example you will add the word “Password” into the text on left side box, and the word “synchronizes” in the text on right side box. By doing this it will scrape the text between those two words which in this case is Changer:

![Image of set dialog box]

Click OK and then run the script. You will be able to then check the debugger to see what your variable was set to. In this case it will be the word “Changer”:

```
#scrape | Changer
```

**Scrape Attribute**

This function returns the value of a specified attribute on the selected element. As with all other scraping, you will be scraping the data to a list or variable. Again, for this example we will be setting a variable and scraping the data to that variable:
In this example we will be scraping the total results displayed on the page. In order to do this we need to drag the $scrape attribute into the New Value box:
From here you will choose the element selector ( ) and choose the results in the browser window:

In this case you will notice you are not provided with a unique attribute that will scrape the desired data:

Therefore, we will need to click on the advanced editor ( ) and choose a different attribute:
We will be choosing innertext and replacing the numbers with wildcards:
1) Replace the numbers with wildcards (*)
2) Choose wildcard from the dropdown
3) Click Add and OK
You will then need to choose the attribute you wish to scrape. In this case we will choose `outertext` from the dropdown and click OK:

Once this is done, you can run the script and see the scraped data in the debugger:
Element Offset

This function allows you to return just one of the elements when the element selector matches more than one element (i.e. several elements share the exact same attributes). In the following page we will attempt to scrape only the username:

The problem with this page is all the information contained within Personal and Messaging areas share one attribute (<dd>):

So in this case we can set a variable to store the username (i.e. #username) and use the element offset function to scrape the data. We start by dragging the set command into
the scripting area, drag the $scrape attribute into the “New Value” box, and drag the $element offset into the $scrape attribute node:

In the element offset node you will use the element selector, choose the username on the page, and then switch to the advanced editor where you can use the tagname attribute (which is “dd” as shown above). Enter 0 into the offset box as this will choose the first occurrence of this tagname and scrape your chosen data:
**Element Parent/Child/Sibling**

These functions return the parent, child and sibling of an element. Elements are arranged in a hierarchical order within the html structure. Below you will see an excellent explanation of this structure borrowed from the w3schools website:

Look at the following HTML fragment:

```html
<html>
  <head>
    <title>DOM Tutorial</title>
  </head>
  <body>
    <h1>DOM Lesson one</h1>
    <p>Hello world!</p>
  </body>
</html>
```

From the HTML above:

- The `<html>` node has no parent node; it is the root node.
- The parent node of the `<head>` and `<body>` nodes is the `<html>` node.
- The parent node of the "Hello world!" text node is the `<p>` node.

And:

- The `<html>` node has two child nodes; `<head>` and `<body>`.
- The `<head>` node has one child node; the `<title>` node.
- The `<title>` node also has one child node; the text node "DOM Tutorial".
- The `<h1>` and `<p>` nodes are siblings, and both child nodes of `<body>`.

If you would like to read the entire lesson, you can find it at:

http://www.w3schools.com/html/dom_nodetree.asp

[EXAMPLES COMING SOON]