INSIDE THE ULTIMA ONLINE GOLD DEMO - THE COMMAND LIST – PART 2

GOAL

It's our goal to get a deep understanding of how the Ultima Online Gold Demo works. This demo is a representation of the rule set from the Ultima Online Second Age Era.

There is proof that some people have already reversed this demo partially or as a whole, however so far no tools or knowledge has been published. This project is to overcome does shortcomings.

URL's with some proof for this:
http://www.runuo.com/forums/general-discussion/94767-help-m-files.html
http://azaroth.org/2008/12/31/your-topic/ (posting by Faust)

If we understand the demo there is a big chance we can alter the demo and even create our own demo. By default mounting horses is not possible in the demo, but what if we can alter the demo and unlock horses; can we then see how horses behaved during T2A?

This demo is 10 years old and I do not understand no one published his/her work. Maybe that DMCA thing is in the way?

UTILITIES USED

IDA Pro, a very professional utility, definitely worth buying, Standard version is affordable.
BRAIN, every one has one, use it

ABOUT ME

I'm just a guy who loves the Ultima universe and knows a bit assembler. Why not combine the two? ☺ In my 13th year my mom won a PC, 286, in a local supermarket. It came with MS-DOS 4.01 on big floppies. One day my mom asked me to continue a disk copy she had started. You had to play disc jockey back then; the PC didn't have enough memory to hold the disk image. I got confused by the DISKCOPY program; my English wasn't good yet and didn't know the difference between source and destination. I really remember panicking and up to this day I believe I copied the destination to the source. Please, don't tell my mom.

CONTINUATION

In Part 1 I showed you how I opened up the Command List. It's a very important array used by the scripting engine and it is really the gate for seeing how the UO Demo code works. Plus, if the UO Demo code is directly based on the UO server code back then, well, 1 plus 1 is 2. In this part we continue the journey into the command list.

AFTER CLEANING UP

Now, I told you I needed to clean up the array to make it visually more attractive. I spent more than a day on this job but it has been rewarding:

```
00606EA0 GLOBAL_CommandList struct_Command <offset a__TK_IF, offset COMMAND__TK_IF, 3, \
00606EA0                                    ; DATA XREF: sub_40CF6E+22↑r
00606EA0                                    ; sub_40CF6E+7A↑r ...
00606EA0                                    offset a__vi_i__0> ; "TK_IF"
00606EA0                 struct_Command <offset a__TK_ELSE, offset COMMAND__TK_ELSE, 2, \
00606EA0                                    offset a__v_i__0>
00606EA0                 struct_Command <offset a__TK_ENDIF, offset COMMAND__TK_ENDIF, 0, \
00606EA0                                    offset a__v__0>
00606EA0                 struct_Command <offset a__TK_WHILE, offset COMMAND__TK_WHILE, 3, \
00606EA0                                    offset a__vi_i__1>
00606EA0                 struct_Command <offset a__TK_ENDWHILE, offset COMMAND__TK_ENDWHILE, 2,\
00606EA0                                    offset a__v_i__1>
00606EA0                 struct_Command <offset a__TK_FOR, offset COMMAND__TK_FOR, 3, \
00606EA0                                    offset a_vii__0>
00606EA0                 struct_Command <offset a__TK_ENDFOR, offset COMMAND__TK_ENDFOR, 2, \
00606EA0                                    offset a__v_i__2>
00606EA0                 struct_Command <offset a__TK_CONTINUE, offset COMMAND__TK_CONTINUE, 2,\
00606EA0                                    offset a__v_i__3>
00606EA0                 struct_Command <offset a__TK_BREAK, offset COMMAND__TK_BREAK, 2, \
00606EA0                                    offset a__v_i__4>
00606EA0                 struct_Command <offset a__TK_GOTO, offset COMMAND__TK_GOTO, 2, \
00606EA0                                    offset a__vi__0>
00606EA0                 struct_Command <offset a__TK_SWITCH, offset COMMAND__TK_SWITCH, 26h, \
00606EA0                                    offset a__vi_j__0>
00606EA0                 struct_Command <offset a__TK_ENDSWITCH, offset COMMAND__TK_ENDSWITCH, 0,\
00606EA0                                    offset a__v__1>
00606EA0                 struct_Command <offset a__TK_CASE, offset COMMAND__TK_CASE, 0, \
00606EA0                                    offset a__v__2>
00606EA0                 struct_Command <offset a__TK_DEFAULT, offset COMMAND__TK_DEFAULT, 0, \
00606EA0                                    offset a__v__3>
00606EA0                 struct_Command <offset a__TK_RETURN__0, offset COMMAND__TK_RETURN__0, 0,\
00606EA0                                    offset a__v__4>
00606EA0                 struct_Command <offset a__TK_RETURN__1, offset COMMAND__TK_RETURN__1, \
00606EA0                                    2, offset a__vi__1>
00606EA0                 struct_Command <offset a__TK_RETURN__2, offset COMMAND__TK_RETURN__2, \
00606EA0                                    56h, offset a__vc__0>
00606EA0                 struct_Command <offset a__TK_RETURN__3, offset COMMAND__TK_RETURN__3, \
00606EA0                                    51h, offset a__vo__0>
00606EA0                 struct_Command <offset a__TK_RETURN__4, offset COMMAND__TK_RETURN__4, \
00606EA0                                    8, offset a__vs__0>
00606EA0                 struct_Command <offset a__oprnull, offset COMMAND__oprnull, 14h, \
00606EA0                                    offset a__ii__0>
00606EA0                 struct_Command <offset a__oprplus, offset COMMAND__oprplus, 15h, \
00606EA0                                    offset a__iii__0>
00606EA0                 struct_Command <offset a__oprminus, offset COMMAND__oprminus, 15h, \
00606EA0                                    offset a  iii  1>
```

And let me show you again the structure uncovered:

```
00000000 ; ---------------------------------------------------------------------------
00000000
00000000 struct_Command  struc ; (sizeof=0x10)
00000000 Command         dd ?                    ; offset
00000004 FunctionAddress dd ?                    ; offset
00000008 UnknownValue    dd ?
0000000C ParaмaterPassing dd ?                   ; offset
00000010 struct_Command  ends
00000010
```

PARAMETER PASSING

I will talk with you about parameter passing to the functions. I did not debug any of those functions and what I will explain next is only based on what I saw while cleaning up.

Let's take a look at the following functions inside the GLOBAL_CommandList:

```
struct_Command <offset a__barkstr, offset COMMAND_barkstr, 8, \
                offset a__vs__1>
struct_Command <offset a__barkint, offset COMMAND_barkint, 2, \
                offset a__vi__3>
struct_Command <offset a__strtoi, offset COMMAND_strtoi, 61h, \
                offset a__is__0>
struct_Command <offset a__strlen, offset COMMAND_strlen, 61h, \
                offset a__is__1>
```

Let's see, the command "barkstr" is followed by "vs". The command "barkint" is followed by "vi", the command "strtoi" is followed by "is" and "strlen" is followed by "is".

Do you also see the pattern emerging?

"strlen" and "strtoi" are taken straight from the C language, "strlen" returns the length of a **s**tring and "strtoi" converts a **s**tring to **i**nteger like "Val" in Basic. The "bark" commands I'm not sure what they do, but it is obvious for me that barkstr takes a **s**tring as parameter and that barkint takes an **i**nteger as parameter.

What I deducted: strtoi -> is -> returns integer, takes string as parameter.
And: strlen -> is -> returns integer, takes string as parameter.

For barkstr: barkstr -> vs -> returns void, takes string as parameter.
For barkint: barkint -> vi -> returns void, takes integer as parameter.

Returning void is actually the same as returning nothing, C language to the rescue.

Let's look at something else:

```
struct_Command <offset a__copyList, offset COMMAND_copyList, 10h, \
                offset a__vll__3>
struct_Command <offset a__sortList, offset COMMAND_sortList, 24h, \
                offset a__vli__3>
```

There is a big chance "l" in "vll" and "vli" means list. "copyList" will copy a list to another list and return void. "sortList" will operate on a list, takes a sort direction as extra parameter and returns void.

It's making sense, right?

Let's continue:

```
struct_Command <offset a__isHuman, offset COMMAND_isHuman, 2Ch, \
                offset a__io__8>
struct_Command <offset a__isMobile, offset COMMAND_isMobile, 2Ch, \
                offset a__io__9>
struct_Command <offset a__isPlayer, offset COMMAND_isPlayer, 2Ch, \
                offset a__io__10>
```

So, both functions take an object as parameter and will return an integer. The integer will then indicate true or false.

```
struct_Command <offset a__getYear, offset COMMAND_getYear, 53h, \
                offset a__i__0>
struct_Command <offset a__getMonth, offset COMMAND_getMonth, 53h, \
                offset a__i__1>
struct_Command <offset a__getWeek, offset COMMAND_getWeek, 53h, \
                offset a__i__2>
struct_Command <offset a__getDay, offset COMMAND_getDay, 53h, \
                offset a__i__3>
```

OK, everything really seems logic; those 4 functions only return an integer and take no parameters.

And does C mean "location":

```
struct_Command <offset a__getTileAt, offset COMMAND_getTileAt, 55h, \
                offset a__ic__0>
struct_Command <offset a__isInMap, offset COMMAND_isInMap, 55h, \
                offset a__ic__1>
struct_Command <offset a__isInWorld, offset COMMAND_isInWorld, 55h, \
                offset a__ic__2>
```

Most probably: "yes".

CONCLUSION

By cleaning up and applying knowledge and logic you can understand some of the things even without understanding how everything works behind the hood.

We now know:

First character = Return value type.
The rest are parameter types and we know how many parameter each function takes.

i = integer
o = object
c = location
v = void

There is also a "u", which I think means "unsigned integer" but that has too proven by deeper analyzing! This knowledge is going to come in handy while working on a script decompiler.