

Console Command Tool (CCT) v1.0

Introduction:

CTT is a small command line utility that allows you to do some useful tasks without the need to display any user interface, although you can also use custom windows with script code or hypertext markup language.

System requirements:

This application is a console executable for x86 processors, and it doesn't require any installation process. Just copy the executable file to any folder you want, and run it with the desired command line option, according to your needs.

The minimum version of the recommended operating system is Windows XP, and the Service Pack 3 is necessary to access some functionalities. Also it works without restrictions in the latest versions of Microsoft Windows (Vista / 7 / 8 / 10).

Output formats:

Most of the functions that return a value of numeric type will store it in the variable `%errorlevel%`. In other cases, the value will be printed on the screen.

String captures in a batch process:

The captures of strings in batch files can be done with the **for** command and the **/f** parameter as follows:

```
for /f "delims=" %%# in ('cct ...') do set "s=%%#"
echo "%s"
```

Console color codes for text (Foreground & Background):

- 0: Black
- 1: Blue
- 2: Green
- 3: Cyan
- 4: Red
- 5: Magenta
- 6: Brown
- 7: White
- 8: Dark gray
- 9: Bright blue
- 10: Bright green
- 11: Bright cyan
- 12: Bright red
- 13: Bright magenta
- 14: Bright yellow
- 15: Bright white

With strings: the color codes must be located between the dollar character (\$) and the semicolon character (;). To restore the original color of the console you can use the at character (@). Examples: \$@; or \$6; or \$40; or \$F4;

ASCII codes:

With strings: the ASCII codes must be located between the octothorpe character (#) and the semicolon character (;). Examples: #0A; or #13;

Index

Usage	4
Mixed operations	4
modeless	4
modal	4
topmost	4
more	4
Main operations	4
download	4
zip	4
unzip	4
base64	4
hash	4
screenshot	4
image2	4
speak	4
regex	4
web-browser	4
swf	4
js	4
vbs	4
script-encoder	4
js-encoded	4
vbs-encoded	4
exec	4
shell	4
rundll	4
key	4
printf	4
echo	4
type	4
color	5
random	5
imath	5
fmath	5
cursor	5
window	5
os-info	5
sleep	5
open	5
save	5
folder	5
popup	5
Parameters	6
unzip	6
base64	6
hash	6
imath	6
fmath	6
regex.....	7
speak	7
key	8
web-browser	8
swf	8
js	8
vbs	8
js-encoded	8
vbs-encoded	8

script-encoder	8
exec	9
shell	9
rundll	9
cursor	9
window	9
open	9
save	9
folder	9
os-info	9
Scripting	10
JScript & VBScript	10
WebBrowser	12
Using COM Objects	14
ActionScript	14
Command line examples	17
download	17
zip	17
unzip	17
hash	17
base64	17
regexp	17
screenshot	17
image2	17
speak	17
web-browser	17
swf	18
js	18
vbs	18
js-encoded	18
vbs-encoded	18
script-encoder	18
rundll	18
shell	18
exec	18
popup	19
open	19
save	19
folder	19
type	19
echo	19
printf	19
key	19
color	19
cursor	19
window	19
random	19
imath	19
fmath	19
sleep	19
os-info	19
Other examples	20

Usage:

cct --operation [/parameter] [value]

Mixed operations:

--modeless	Show non-modal dialog boxes.
--modal	Show a modal window.
--topmost	Show the window above the others.
--more	Print the contents of a file on the screen displayed by specified lines and customized text.

Main operations:

--download	Download files from the specified URL.
--zip	Compress the specified folder or file in ZIP format.
--unzip	Unzip files with ZIP format in the specified folder.
--base64	Encode or decode the specified text or file in base64.
--hash	Calculate the bytes of a text or file and print the result on the screen. This feature is operative from Windows XP (Service Pack 3) for the algorithms: SHA-256, SHA-384, and SHA-512.
--screenshot	Take a screenshot and save it to the clipboard or a file with the specified format (bmp, jpg, png, and gif). <u>GDI+</u> is required.
--image2	Convert from one image format to another. The supported formats are: bmp, jpg, png, and gif. <u>GDI+</u> is required.
--speak	Speak the contents of the specified text or file.
--regex	It allows to operate with texts through regular expressions.
--web-browser	Show a window with embedded <u>Internet Explorer</u> .
--swf	Show a window with embedded <u>Adobe Flash Player</u> .
--js	Run the specified <u>JScript</u> code.
--vbs	Run the specified <u>VBScript</u> code.
--script-encoder	Encode the specified <u>JScript</u> or <u>VBScript</u> code.
--js-encoded	Run the specified <u>JScript</u> code in encoded format.
--vbs-encoded	Run the specified <u>VBScript</u> code in encoded format.
--exec	Run a program with the specified parameters.
--shell	Run the Windows Shell with the specified parameters.
--rundll	Run a dynamic library procedure with the specified parameters.
--key	Receive the keys pressed on the console. It has support for printing color characters on the screen.
--printf	Print text on the screen with the specified format. It has support for printing color characters.
--echo	Print the specified text on the screen. It has support for printing color characters and ASCII codes.
--type	Print the contents of the specified file on the screen. It has support for printing color characters.

Console Command Tool (CCT) v1.0

--color	Change the text color in the current console.
--random	Generate a random number and return the result. Allows ranges from least to highest.
--imath	Calculate basic operations with integer numbers.
--fmath	Calculate mathematical operations with float numbers.
--cursor	It modifies or reports about the console cursor.
--window	It modifies or reports about the console window.
--os-info	Report about the operating system.
--sleep	Suspend the execution of the current thread until the time-out interval elapses.
--open	Show the open file dialog box.
--save	Show the save file dialog box.
--folder	Show the Shell folder picker.
--popup	Show the specified message box.

Parameters:

unzip:
/quiet Silent mode (Operational from Windows Vista).

base64:
/encode Encode indicator.
/decode Decode indicator.
/file Specify the path of a file for reading.
/output Specify the path of a file for writing.

hash:
/md5 Message-Digest Algorithm 5 (128 bits).
/sha1 Secure Hash Algorithm 1 (160 bits).
/sha256 Secure Hash Algorithm 2 (256 bits).
/sha384 Secure Hash Algorithm 2 (384 bits).
/sha512 Secure Hash Algorithm 2 (512 bits).
/file Specify the path of a file for reading.

imath:
/min Return the minimum value of two integers.
/max Return the maximum value of two integers.
/xor It performs a bitwise logical operation XOR (exclusive OR) between two integer operators and return the result.

/or It performs a bitwise logical operation between two integer operators and return the result.

/not It performs a bitwise logical operation between two integer operators and return the result.

/and It performs a bitwise logical operation between two integer operators and return the result.

/sum Add two integer operators and return the result.
/sub Subtract two integer operators and return the result.
/mul Multiply two integer operators and return the result.
/div Divide two integer operators and return the result.
/mod Divide two integer operators and return the remaining result.

fmath:
/min Print the minimum value of two floating numbers on the screen with the specified format.

/max Print the maximum value of two floating numbers on the screen with the specified format.

/sum Add two floating numbers and print the result on the screen with the specified format.

/sub Subtract two floating numbers and print the result on the screen with the specified format.

/mul Multiply two floating numbers and print the result on the screen with the specified format.

/div Divide two floating numbers and print the result on the screen with the specified format.

/mod	Divide two floating numbers and print the remaining result on the screen with the pecified format.
/sqrt	Print the square root of a floating number on screen with the specified format.
/exp	Print the base-e exponential of a floating number on screen with the specified format.
/sin	Print the sine of an angle with a floating number (radians) on screen with the specified format.
/sinh	Print the hyperbolic sine of a floating number on screen with the specified format.
/cos	Print the cosine of an angle with a floating number (radians) on screen with the specified format.
/cosh	Print the hyperbolic cosine of a floating number on screen with the specified format.
/tan	Print the tangent of an angle with a floating number (radians) on screen with the specified format.
/tanh	Print the hyperbolic tangent of a floating number on screen with the specified format.
/log	Print the natural logarithm of a floating number on screen with the specified format.
/log10	Print the common (base-10) logarithm of a floating number on screen with the specified format.
/floor	It rounds a floating number downward and print the largest integral value on screen with the specified format.
/ceil	It rounds a floating number upward and print the smallest integral value on screen with the specified format.
/pow	Print the base raised to the power exponent on screen with the specified format.
/format	Specify the printing format.

regex:

/i	Case insensitive mode.
/g	Global mode.
/test	It returns True or False variant type, indicating if the regular expression matches.
/replace	It replaces the part of the string found in a match with another string.

speak:

/file	Specify the path of a file for reading and speaking.
-------	--

key:

/noecho	It indicates that the character is not printed on the screen.
/down	It Indicates that the event is thrown when a key is pressed and not when it is released.

The supported keys are: From A to Z, numbers, [esc], [enter] or [return], [left], [down], [right], [up], [f1], [f2], [f3], [f4], [f5], [f6], [f7], [f8], [f9], [f10], [f11], [f12], [back], [insert], [delete], [home], [end], [prior], [next], and [tab].

web-browser:

/title	Specify a fixed title for the window.
/arguments	Collect the specified arguments for the application.
/unescape	It prevents the window from closing when the escape key is pressed and released.
/unscrolling	Show a window without scroll-bars.
/unselectable	Show a window with non-selectable text.
/resizable	Show a resizable window.
/dialog	Show a dialog box.
/flat	Show a flat window.
/fullscreen	Show a full-screen window.
/trusted	It indicates that the local web page is trusted to work with ActiveX objects.
/key	Specify a security key to work with ActiveX objects from external pages to the local machine. The /trusted parameter is required.

swf:

/title	Specify a title for the window.
/icon	Specify a icon file for the window.
/arguments	Collect the specified arguments for the application.
/background	Specify a background color.
/menuless	It disables the context menu.
/unescape	It prevents the window from closing when the escape key is pressed and released.
/resizable	Show a resizable window.
/dialog	Show a dialog box.
/flat	Show a flat window.
/fullscreen	Show a full-screen window.
/region	Show a region window.
/alpha	Show a alpha window. The quantity must be specified as a percentage.

js:

/file	Specify the path of a file for reading and execution.
-------	---

vbs:

/file	Specify the path of a file for reading and execution.
-------	---

js-encoded:

/file	Specify the path of a file for reading and execution.
-------	---

vbs-encoded:

/file	Specify the path of a file for reading and execution.
-------	---

script-encoder:

/file	Specify the path of a file for reading.
/output	Specify the path of a file for writing.

/type	Specify the type of script (js or vbs).
/base64	It indicates that the resulting code must be encoded in base64. If no /output is indicated it is not necessary.

exec:

/partner	The new process will be tied to the current console.
/detached	The new process will be a separate process.
/console	Create a new console for the new process.

shell:

/directory	Specify the working directory.
/max	Show the maximized main window.
/min	Show the minimized main window.
/hide	Hide the main window.
/wait	Wait for the process to finish and return the result.

The allowed verbs are: edit, open, print, printto, explore, find, and runas.

rundll:

/cdecl	It indicates a call of type C.
/float	It indicates a type of returned value.
/string	It indicates a type of returned value.
/format	Specify the printing format on the screen.

cursor:

/get	It indicates that a value must be returned.
/set	It indicates that a value must be defined.
/x	Specify the right position of console cursor.
/y	Specify the top position of console cursor.

window:

/get	It indicates that a value must be returned.
/set	It indicates that a value must be defined.
/x	Specify the right position of console window.
/y	Specify the top position of console window.
/width	Specify the width of console window.
/height	Specify the height of console window.
/center	Center the console window on the screen.

open:

/title	Specify the title of the dialog box.
/filter	Specify the group names for the allowed extensions.

save:

/title	Specify the title of the dialog box.
/filter	Specify the group names for the allowed extensions.

folder:

/title	Specify the title of the dialog box.
--------	--------------------------------------

os-info:

/type	Return the product type.
/build	Return the build number of the operating system.
/platform	Return the operating system platform.
/suite	Return the product suites available on the system.
/sp	Return the version number of the latest service pack installed on the system.
/major	Return the major version number of the operating system.
/minor	Return the minor version number of the operating system.

Without parameters, it returns a legend. For more information you can see the [RTL OSVERSIONINFOEXW structure](#).

Scripting:

This application allows you to use automation technology that provides scripting capabilities similar to batch files, but with a wider range of supported functions. You can also use script code from the command line.

Extended objects and functions have been created for the operations: `--js[-encoded]`, `--vbs[-encoded]`, `--web-browser`, and `--swf`. The supported languages are: JScript, VBScript, and ActionScript respectively.

JScript & VBScript:

console object (`--js[-encoded]`, and `--vbs[-encoded]` operations):

Depending on the scripting language you use and the task you have chosen, you might or might not need to use the **console** object. For example, JScript and VBScript does not include functions for parsing command-line arguments. Fortunately, you can still use command-line arguments; you simply make use of the argument parsing functionality provided by the **console** object, whose methods and properties are:

.arguments ([[index]]):

Command-line arguments are values that you enter at the command line when you run a script. The arguments must be separated by one or more spaces. Because of this, command line arguments that contain blank spaces must be enclosed in quotation marks (") to be treated as a single argument. Use a zero-based index to retrieve individual arguments with this method. If no index is added the property returns the total number of arguments.

Arguments:

index: Number of argument to receive. The first element starts at zero.

Example (JScript):

```
for (var i = 0, n = console.arguments; i < n; ++i)
    console.echo(console.arguments(i) + "\n");
```

.echo ([operation], string):

Output the specified text to either the console window. It has support to print colored characters.

Arguments:

operation: String that specifies an operation. At the moment the only operation available is **colored**.

string: Character set to print on screen.

Example (JScript):

```
console.echo("Hello World!\n");
console.echo("colored", "$6;Hello $5;World$@;!\n");
```

.hwnd [()]:

Return a value of numeric type that is the window identifier of the console.

Example (JScript):

```
var hwnd = console.hwnd;
```

.more ([text], string):

Output the specified text to either the console window. It has support to print colored characters.

Arguments:

text: The customized string that indicates if there is more to display.

string: Character set to print on screen.

Example (JScript):

```
var fs = new ActiveXObject("Scripting.FileSystemObject");
var tf = fs.OpenTextFile("D:\\My File.txt", 1);
var sf = tf.ReadAll();
```

```
tf.Close();
console.more(sf);
console.more(" -- My Customized More Text -- ", sf);
```

.password ([[text]]):

Return the specified character set and prints asterisks (*) on the screen.

Arguments:

text: Specify an indicative text.

Example (JScript):

```
var p = console.password;
var p = console.password("Password: ");
```

.popup (message, title, flag):

Show the specified message box and returns the response identifier.

Arguments:

message: Specify a body for the message box.

title: Specify a title for the message box.

flag: The contents and behavior of the dialog box.

For more information you can see the [MessageBox](#) function.

Example (JScript):

```
var n = console.popup("My message!", "My title", 49);
```

.quit `[[code]]`:

Forces script execution to stop at any time.

Arguments:

code: Integer value returned as the process's exit code.
If you do not include the **code** parameter, the zero value is returned.

Example (JScript):

```
console.quit(-1);
```

.sleep `(milliseconds)`:

Suspends the execution of the current thread until the time-out interval elapses.

Arguments:

milliseconds: Integer value indicating the interval you want the script process to be inactive.

Example (JScript):

```
console.sleep(1000);
```

WebBrowser:

`window.external` object (`--web-browser` operation):

Use the `window.external` object in your scripting code to access public properties and methods:

.arguments `[[index]]`:

Command-line arguments are values that you enter at the command line when you run a web-browser operation. The command-line arguments follow the **/arguments** parameter and are separated by one or more spaces. Because of this, command line arguments that contain blank spaces must be enclosed in quotation marks (") to be treated as a single argument. Use a zero-based index to retrieve individual arguments with this method. If no index is added the property returns the total number of arguments.

Arguments:

index: Number of argument to receive. The first element starts at zero.

Example (JScript):

```
for (var i = 0, n = window.external.arguments; i < n; ++i)  
    alert(window.external.arguments(i));
```

.echo (string):

Output the specified text to either the console window.

Arguments:

string: Character set to print on screen.

Example (JScript):

```
window.external.echo("Hello World!\n");
```

.hwnd [()]:

Return a value of numeric type that is the window identifier of the web-browser control.

Example (JScript):

```
var hwnd = window.external.hwnd;
```

.popup (message, title, flag):

Show the specified message box and returns the response identifier.

Arguments:

message: Specify a body for the message box.

title: Specify a title for the message box.

flag: The contents and behavior of the dialog box.

For more information you can see the [MessageBox](#) function.

Example (JScript):

```
var n = window.external.popup  
(  
    "My message!"  
    , "My title"  
    , 49  
);
```

.sleep (milliseconds):

Suspends the execution of the current thread until the time-out interval elapses.

Arguments:

milliseconds: Integer value indicating the interval you want the script process to be inactive.

Example (JScript):

```
window.external.sleep(1000);
```

.trusted (enabled, key):

Specifying a security key allows you to work with ActiveX objects from external pages to the local machine. The **/trusted** parameter is required.

Arguments:

enabled: Specify a **true** or **false** value.
title: Specify the security key.

Example (cmd):

```
cct --web-browser /trusted /key "My Custom Key..."  
http://www.mydomain.com
```

Example (JScript):

```
window.external.trusted(true, "My Custom Key...");
```

Using COM Objects (--js[-encoded], --vbs[-encoded], and --web-browser):

To use a COM object in a script run by script engine, you must first create an instance of the object. You can do this by calling the CreateObject method (in VBScript) or the ActiveXObject object (in JScript).

The following example illustrates creating an ActiveXObject object using JScript:

```
var obj = new ActiveXObject("Scripting.FileSystemObject");
```

The following example illustrates calling CreateObject using VBScript:

```
Dim obj  
Set obj = CreateObject("Scripting.FileSystemObject")
```

ActionScript (--swf operation):

To have access to all the functionalities, the scripts must be in ActionScript 3.0, especially to use the ExternalInterface class.

fscommand:

Lets the SWF file communicate with either the program hosting Flash Player and other programs that can host ActiveX controls.

To use fscommand() to send a message to Flash Player, you must use predefined commands and parameters. For more information you can see the [Adobe reference web-site \(fscommand\)](#).

echo: Print a text on the console.
exec: Execute an application.
quit: Close the window.

showmenu:

Specifying "true" enables the full set of context menu items.
Specifying "false" hides all of the context menu items.

ExternalInterface.call:

The ExternalInterface class is an application programming interface that enables straightforward communication between ActionScript and the SWF container. For this purpose you will need to use the **import** directive to access the methods of the class. For more information you can see the [Adobe reference web-site \(ExternalInterface\)](#).

arguments:

Command-line arguments are values that you enter at the command line when you run a swf operation. The command-line arguments follow the **/arguments** parameter and are separated by one or more spaces. Because of this, command line arguments that contain blank spaces must be enclosed in quotation marks (") to be treated as a single argument.

The call returns a string in JSON format that can be parsed to pick up an array of arguments.

Example (ActionScript):

```
import flash.external.*;

var sarg: String = ExternalInterface.call("arguments");
var args: Array;

JSON.parse(sarg, function (o, v)
{
    if (o == "arguments")
        args = v;

    return v;
});
```

getpath:

Return the path of the specified directory.

Arguments:

name: At the moment the only available **name** is **temp** that returns a string with the path of the system temporary directory.

Example (ActionScript):

```
import flash.external.*;

var temp: String = ExternalInterface.call
(
    "getpath"
    , "temp"
);
```

hwnd:

Return a value of numeric type that is the window identifier of the SWF control.

Example (ActionScript):

```
import flash.external.*;

var hwnd: Number = ExternalInterface.call("hwnd");
```

popup:

Show the specified message box and returns the response identifier.

Arguments:

message: Specify a body for the message box.
title: Specify a title for the message box.
flag: The contents and behavior of the dialog box.

For more information you can see the [MessageBox function](#).

Example (ActionScript):

```
import flash.external.*;

var result: Number = ExternalInterface.call
(
    "popup"
    , "My message"
    , "My title"
    , 49
);
```

sleep:

Suspends the execution of the current thread until the time-out interval elapses.

Arguments:

milliseconds: Integer value indicating the interval you want the script process to be inactive.

Example (ActionScript):

```
import flash.external.*;

ExternalInterface.call("sleep", 1000);
```


Command line examples:

download:

```
cct --download https://mydomain.net/myfile.zip
cct --download http://mydomain.net/myfile.zip "D:\My File.zip"
cct --download http://mydomain.net/myfile.zip "D:\My Folder"
```

zip:

```
cct --zip "D:\My Folder"
cct --zip "D:\My Folder" "My File.zip"
cct --zip "D:\My Folder\My File.doc"
cct --zip "My File.doc" "D:\My Folder\My File.zip"
```

unzip:

```
cct --unzip "My File.zip" "D:\My Folder"
cct --unzip /quiet "My File.zip" "D:\My Folder"
```

hash:

```
cct --hash /md5 "Hello World!"
cct --hash /sha1 /file "D:\My Folder\My File.bin"
```

base64:

```
cct --base64 /encode "Hello World!"
cct --base64 /decode "SGVsbG8gV29ybGQh"
cct --base64 /encode /file "D:\My Folder\My File.txt"
cct --base64 /encode /file "My File.bin" /output Base64.txt
cct --base64 /decode /file Base64.txt /output "My File.bin"
```

regex:

```
cct --regex "[0-9]" /test "abc 123"
cct --regex "(\d{3})-(\d{3})-(\d{4})" /g "555-123-4567"
    /replace "$1" "$2-$3"
```

screenshot:

```
cct --screenshot
cct --screenshot "D:\My Folder\My File.jpg"
```

image2:

```
cct --image2 "D:\My Folder\My File.bmp" .png
cct --image2 "D:\My File.png" "C:\My File.jpg"
```

speak:

```
cct --speak "Hello World!"
cct --speak /file "D:\My File.txt"
```

web-browser:

```
cct --web-browser http://www.mydomain.com
cct --web-browser 800 600 /resizable http://www.mydomain.com
cct --web-browser /dialog file:///d:/my-directory/my-page.html
cct --web-browser /fullscreen http://www.mydomain.com
cct --web-browser /trusted 800 600 file:///d:/my-page.html
    /arguments "argument 1" "argument 2" "argument ..."
```

swf:

```
cct --swf 800 600 http://www.my-website.com/my-file.swf
cct --swf /region /background 0 "D:\My Folder\My File.swf"
cct --swf /alpha 75 /background 0 /menuless "D:\My File.swf"
cct --swf /fullscreen "D:\My Folder\My File.swf"
cct --swf /title "My Title" /icon "D:\My Folder\My Icon.ico"
"D:\My Folder\My File.swf" /arguments "argument 1"
"argument 2" "argument ..."
```

js:

```
cct --js "new ActiveXObject('WScript.Shell').SendKeys('{F7}')"
cct --js Y29uc29sZS5lY2hvKCdIZWxsbyBxb3JsZCEnKts=
cct --js /file "D:\My Folder\My File.js"
cct --js /file "D:\My File.js" "argument 1" "argument ..."
```

vbs:

```
cct --vbs "MsgBox \"Hello World!\""
cct --vbs Y29uc29sZS5lY2hvICJIZWxsbyBxb3JsZCEi
cct --vbs /file "D:\My Folder\My File.vbs"
cct --vbs /file "D:\My File.vbs" "argument 1" "argument ..."
```

js-encoded:

```
cct --js-encoded /file "My Encoded File.js"
cct --js-encoded I0B+XkhBQUFBQT09Xkt4ZEtf1IrMXRLY0JfK3NWS1AJR01WTmV2I25Ba
0FBQT09XiN+QAA=
```

vbs-encoded:

```
cct --vbs-encoded /file "My Encoded File.vbs"
cct --vbs-encoded I0B+Xkd3QUFBQT09Xkt4ZEtf1IrMXRLfKpfK3NWS1AJR01WTmVFWVFr
QUFBPT1eI35AAA==
```

script-encoder:

```
cct --script-encoder /file "My File.js"
cct --script-encoder /file "My File.vbs" /output "My Encoded File.vbs"
cct --script-encoder "console.echo('Hello World!')" /type js
cct --script-encoder /file "My File.txt" /type vbs /base64 /output
"My Encoded File.vbs"
```

rundll:

```
cct --rundll user32.dll MessageBoxA 0 "Hello World!" Test 64
cct --rundll msvcrt.dll /cdecl printf "<%c%c%c>" 65 66 67
cct --rundll msvcrt.dll /cdecl printf "floats: %4.2f %+.0e"
3.1416 3.1416

cct --rundll msvcrt.dll /float /format %4.2f /cdecl pow
3.1416 3.1416
```

shell:

```
cct --shell "D:\My Folder\My File.pdf"
cct --shell "My File.jpg"
cct --shell http://www.mydomain.com
cct --shell chrome http://www.mydomain.com
cct --shell print "D:\My File.doc"
cct --shell printto "D:\My File.txt" "Microsoft Print to PDF"
cct --shell edit "D:\My Folder\My File.png"
cct --shell find "D:\My Folder"
cct --shell explore "D:\My Folder"
```

exec:

```
cct --exec /detached notepad "D:\My Folder\My File.txt"
cct --exec /partner mshta "D:\My Folder\My File.hta"
```

popup:

```
cct --popup "Hello World!" "My message" 64
cct --popup "Message with#0A;#0D;line break" "My title" 49
```

open:

```
cct --open /title "Open file" /filter
    "Image files|.bmp;*.jpg;*.png;*.gif|All files|*.*|"
```

save:

```
cct --save /title "Save file" /filter
    "Image files|.bmp;*.jpg;*.png;*.gif|All files|*.*|"
```

folder:

```
cct --folder /title "Select a destination folder"
```

type:

```
cct --type "D:\My Folder\My File.txt"
cct --more --type "D:\My Folder\My File.txt"
cct --more " -- My More Text -- " --type "D:\My Folder\My File.txt"
cct --more " -- My More Text -- " 10 --type "My File.txt"
```

echo:

```
cct --echo Hello World!
cct --echo $6;Hello $5;World$@;!#0A;#0D;
```

printf:

```
cct --printf "Characters: %c%c%c" a 98 c
cct --printf "Decimals: $12;%d$@; %ld %c%c" 1977 FFFF 10 13
```

key:

```
cct --key "Press any key to continue..." /noecho /down
cct --key "Press one of the following keys [$14;A$@;, or $14;B$@;]: " ab
```

color:

```
cct --color F4
cct --color 10
```

cursor:

```
cct --cursor /get /x
cct --cursor /set /y 10
```

window:

```
cct --window /get /width
cct --window /set /height 500
cct --window /center
```

random:

```
cct --random
cct --random 50 100
```

imath:

```
cct --imath /max 2 10
cct --imath /mod 15 2
```

fmath:

```
cct --fmath /sum 2.56 3.68
cct --fmath /format %.f /sqrt 3.2
```

sleep:

```
cct --sleep 5000
```

os-info:

```
cct --os-info /major
```

Other examples:

Password in a batch file:

```
@echo off
chcp 1252 >nul

setlocal
cct --js "var s = new ActiveXObject('WScript.Shell'); s =
s.Environment('VOLATILE'); s('pwd') = console.password('Password: ')"

for /f "delims=" %%# in ('cct --js "var s = new ActiveXObject('WScript.
Shell'); s = s.Environment('VOLATILE'); console.echo(s('pwd')); s('pwd') =
';"') do set "p=%%#"

echo The password is: "%p%"

pause
endlocal
```

Kiosk mode with Microsoft Edge:

> In a line:

```
cct --js "var s = new ActiveXObject('WScript.Shell'); s.Run('cmd /c start
microsoft-edge:http://www.mydomain.com'); console.sleep(1000);
s.SendKeys('{F11}')"


```

> In a batch file:

```
@echo off
start microsoft-edge:http://www.mydomain.com
cct --sleep 1000
cct --js "new ActiveXObject('WScript.Shell').SendKeys('{F11}')"


```

Kiosk mode with Google Chrome:

> In a line:

```
cct --js "var s = new ActiveXObject('WScript.Shell'); s.Run('cmd /c start
chrome -kiosk http://www.mydomain.com'); console.sleep(1000);
s.SendKeys('{F11}')"


```

> In a batch file:

```
@echo off
start chrome -kiosk http://www.mydomain.com
cct --sleep 1000
cct --js "new ActiveXObject('WScript.Shell').SendKeys('{F11}')"


```