



JP Software

Take Command 9
Tutorial 2 -- Language Basics

Tutorial 2

Take Command 9 Language Basics

Take Command 9 provides a rich command language that is highly upwardly compatible with the Microsoft CMD.EXE command language. It is suitable for creating both simple and highly sophisticated batch programs. The language can also be used at the command prompt to create very powerful real-time manipulation of your computer.

Overview

The Take Command 9 (TC9) Language has a huge set of capabilities. These capabilities are grouped into three categories:

- **Internal Commands** - These are the primary language constructs. Common commands include, DIR, COPY, MOVE, etc. The Take Command Language (4NT) gives you instant access to more than 150 internal commands. (By contrast, Microsoft's CMD.EXE has fewer than 40 internal commands.).

Internal vs External Commands

When we talk about an internal command, we mean the command is built into the Take Command program. With CMD, some commands, like XCOPY are actually separate programs. In Powershell, the commands are generally external programs. Much of Powershell requires a separate .NET Framework to be installed on the computer for the commands to work.

This is important, because it means that you can install Take Command, you know that all of the pieces are there for you (and its very compact). You can install it on a USB stick and have it work on any computer you plug the USB stick into. Try that with Powershell

- **Internal Variable** - Internal variables are special variables built into Take Command to provide information about your system. They are not stored in the environment, but can be accessed as if they were environment variables in interactive commands, aliases, and batch files. Take Command provides more than 130 internal variables that can tell you a great deal about your computer and how it is operating. These include installed hardware, hardware status, operating system and software status, etc
- **Variable Functions** - Variable functions are very similar to internal variables, but they take one or more parameters (which can be environment variables or even other variable functions). Variable functions are useful at the command prompt as well as in aliases and batch files to check on available system resources, manipulate strings and numbers, and work with files and filenames. There are more than 230 variable functions built into Take Command.

We are not going to talk about all of the features of the Take Command Language in this tutorial. The manual is 1,200 pages long. We are going to assume you know the basics of CMD and point you at a few of things that Take Command does better with less work than CMD .

Internal Commands

There are several aspects of Take Command's internal command set that are definitely worth looking at:

- Switches
- Aliases
- HTTP and FTP
- Flow of Control Commands

- Keystack Command
- Event Monitoring Commands (Triggers) -- We made this into a whole separate Tutorial 3

Each of these is covered below:

1. **Switches**

Switches modify commands by giving them special instructions. In general Take Command has a superset of the CMD switches and is generally compatible. We say generally, because unfortunately, CMD has not been consistent from version to version.

For example, in CMD, the COPY command has 7 switches (XCOPY has more). The TC9 COPY command has 25 switches. Examples of some of the switches that the CMD COPY command does not have include:

- **/N** Executes the copy command and shows you what the output would be, but does not actually execute the command
- **/O** Copy the source file, only if the target does not exist
- **/S** Copy the subdirectory tree starting with the files in the source directory plus each subdirectory below that
- **/H** Copy all matching files including those with a hidden or system attribute set

These switches allow you to custom tailor the language in ways that you cannot do with CMD. They perform very powerful operations with only two or three keystrokes.

2. **Aliases**

So, we have given you all these powerful switches, but maybe you don't always want to have to type them. Perhaps you have some common ways of doing things and you want Take Command to adapt things to the way you want to do them.

We have a solution...aliases. The TC9 aliasing capability is a massive superset of the CMD DOSKEY command (which is an external command).

Much of the power of 4NT and TC comes together in aliases, which give you the ability to create your own commands. An alias is a name that you select for a command or group of commands.

Simple aliases substitute a new name for an existing command. More complex aliases can redefine the default settings of internal or external commands, operate as very fast in-memory batch files, and perform commands based on the results of other commands. 4NT and TC also support Directory Aliases, a shorthand way of specifying pathnames.

Aliases can also create customized versions of commands. For example, the DIR command can sort a directory in various ways. You can create an alias called DE that means "sort the directory by filename

```
alias de=dir /oe /p
```

extension, and pause after each page while displaying it" like this:

So, you don't actually have to remember all those switches. You can set up versions of the language that meet your needs. For example, you can create aliases that match common Linux shell commands and operate the same way if that's more comfortable for you.

TC9 aliasing differs from CMD in a couple of key ways.

- You can use aliases in batch files. DOSKEY macros cannot be used in your batch files.

- TC9 aliases can include variable expansion, which is not available in DOSKEY. This example creates a simple command line calculator.

```
alias calc=`echo The answer is: %@eval[%$]
```

Once you have entered the example, you can type `CALC 4*19`, for example, and you will see the answer. The variable function `%@eval[%$]` will be evaluated by the parser, The text "4*19" will replace the "\$" placeholder and the variable function `@eval` which performs math functions will calculate the result.

- You can assign a commonly used alias to a keystroke. For example:

```
alias @Shift-F5=*dir /2/p
```

After you enter this example, you will see a 2-column directory with paging whenever you press Shift-F5 followed by Enter.

- TC9 also allows Directory Aliases. Directory Aliases are a shorthand way of specifying pathnames. For example, if you define an alias:

```
alias pf:=c:\program files
```

You can then reference the files in `c:\program files\jpssoft` by entering `pf:\jpssoft`.

3. **Flow of Control Commands**

One of the weakest areas of CMD is flow of control. These are the constructs like `IF..THEN..ELSE` or `DO LOOPS` that allow you to develop sophisticated batch programs. If you are creating data center batch processes, CMD keeps you from doing anything sophisticated.

TC9 provides a very rich set of constructs that allow you to duplicate most of the capabilities of the old IBM Mainframe Job Control Language (JCL) or typical Linux shells.

The following box shows some of the types of DO Loops you can create

Do Loops

```
DO count
DO FOREVER
DO varname = start TO end [BY step]
DO WHILE condition
DO UNTIL condition
DO UNTIL DATETIME date time
DO FOR n [SECONDS | MINUTES | HOURS]
DO varname IN [range...] [/I:"text" /S[n] /A:[- | +]hsad] fileset
DO varname IN [/T"delimiters" /L stringset
DO varname IN /C stringset
DO varname IN @file
```

TC9 also provides a very powerful IF..THEN..ELSE structure through the IFF command.

```
If...Then...Else Constructs
```

```
IFF condition1 THEN  
    commandset1  
[ELSEIFF condition2 THEN  
    commandset2 ]  
    ...  
[ELSE  
    commandset3 ]  
ENDIFF
```

The alias in this IFF example checks to see if the parameter is a subdirectory. If so, the alias deletes the subdirectory's files and removes it (enter this on one line):

```
alias prune `iff isdir %1 then & del /s /x /z %1 & else & echo %1 is not a directory! & endiff`
```

This example shows how a SWITCH construct works. The batch file fragment below displays one message if the user presses A, another if the user presses B or C, and a third one if the user presses any other key:

```
Switch Constructs
```

```
inkey Enter a keystroke: %%key  
switch %key  
case A  
    echo It's an A  
case B .or. C  
    echo It's either B or C  
default  
    echo It's none of A, B, or C
```

4. Keystack

KEYSTACK takes a series of keystrokes and feeds them to a program or command as if they were typed at the keyboard. It has no equivalent in CMD. KEYSTACK is most often used for programs started from batch files. For example, to start Word and open the last document you worked on, you could use the command :

```
start word & keystack /w54 alt-f "1"
```

This causes the following:

- Starts Word,
- The /w switch causes a delay of about three seconds (54 clock ticks at about 1/18 second each) for Word to get started,
- Places the keystrokes for alt-F (File pulldown menu), and 1 (open the most recently used file) into the buffer.

Word receives these keystrokes and performs the appropriate actions. Notice that the two commands, START and KEYSTACK are issued on a single command line. This ensures that the keystrokes are sent to Word's window, not back to Take Command.

5. **FTP and HTTP -- Additional Coverage in Tutorial 4**

TC9's FTP and HTTP commands allow you to treat http and ftp sites as if they were local disk drives. This is a huge advantage over CMD. In Tutorial 4, we show you how to use these commands to create practical remote monitoring applications.

In simplest form, you can act as if an FTP or HTTP site is a local disk. For example, to get a directory of the JP Software FTP site, you could use this command:

```
Dir ftp://jpsoft.com/*
```

The following example shows how to include an ftp user name and password:

```
Dir ftp://username:password@ftp.abc.com/mydir/*
```

You can reference internet sites for DIR, COPY, MOVE, DEL and other commands. These commands also work with secure versions of FTP and HTTP.

6. **Event Monitoring Commands (Triggers) -- See Tutorial 3**

One of the most powerful features in TC9 are the 7 new event monitoring commands. They allow you to watch a wide variety of activities on you computer and "trigger" processes into action to deal with or report on issues.

This is described fully in Tutorial 3. Its well worth the read.

Internal Variables

Internal variables are special variables built into Take Command to provide information about your system. They are not stored in the environment, but can be accessed as if they were environment variables in interactive commands, aliases, and batch files.

There are more than 120 of them. Key types of variables include:

- Hardware status
- Operating system and software status
- Dates and times
- Drives and directories
- Error codes
- Screen, color, and cursor
- Take Command status
- Compatibility

Here is a simple example of how to use a common variable called `_DOW` (Day Of Week):

```
if "%_DOW" == "Mon" call c:\cleanup\weekly.bat
```

This example calls another batch file if today is Monday.

Before we go on...

A Quick Note:

One of the great mysteries of the command line is the % sign. What does it do? When you see a % sign in front of a variable or function, it means that the parser should evaluate the function and replace the variable or function with its text value. So, in the last example, %_DOW is replace with the result, which in this case is MON, TUE or whatever.

How about something more realtime that you can run in the background:

```
DO FOREVER
if "%_BATTERYPERCENT" LT 25" MSGBOX Battery is low
ENDDO
```

This command will loop forever checking the battery status and popup a messagebox if the battery charge is getting low. Messagebox is actually a very powerful command in TC9. Check it out in the help file.

Here is an example that checks to see if there are enough resources free before running an application.

```
iff %_GDIFREE lt 40 then
  echo Not enough GDI resources!
  quit
else
  d:\mydir\myapp
endiff
```

Take a look at the list of internal variables by category in the help file. Its a pretty comprehensive status information available at your fingertips.

Variable Functions

Variable functions are one of the most powerful features of TC9. Variable functions are very similar to internal variables, but they take one or more parameters (which can be environment variables or even other variable functions).

Variable functions are useful at the command prompt as well as in aliases and batch files to check on available system resources, manipulate strings and numbers, and work with files and filenames.

There are more than 240 Variable Functions grouped into 8 categories. They allow you to gather and manipulate system information in very powerful ways. CMD has a small fraction of these functions available.

Remember...they are all built-in.

- Dates and times
- Network properties
- Drives and devices
- Numbers and arithmetic
- File content
- Strings and characters
- File names
- System status
- File properties
- Utility
- Input dialog boxes

Using functions, TC9 can read and write text files, as well as some specialty files, such as the Windows Registry or .ini files. In the example below, we are going to read a .csv file called names.csv (which is a text file with fields separated by commas). The file looks as follows:

```
Joe,100,joe@company.com
Jane,200,jane@company.com
Peter,400,peter@company.com
```

Our example will read this file a line at a time, and select the email address in each line. It will then echo them to the console.

```
set filename=%@expand[names*.csv]
do record in %@filename
    set email=%@field["",3,%record]
    echo %email
enddo
```

This code does the following:

- The first line of the example creates a variable with the full file and pathname of the csv file using the `@expand` function.
- The second line uses a special case of the `DO` command to:
 - Opens the filename we set in the first line with an `@filename` function
 - Creates a line counter that it sets to one
 - Sets up a new variable called "record"
 - Reads the first line of text up to the CR and returns it to "record"
- The `@field` function picks the third field in the line (which contains the email address) using a ";" as the field delimiter and places it in a variable called "email". The delimiter could be anything you wanted.
- The `echo` command outputs the email address to the console
- The `enddo` returns the loop to the `do` statement, which increments the line counter to the next line. If its the end of the file, it terminates the loop.

This particular example seems sort of limited, but we use a variant of it to process our orders, construct registration keys and email them to our users with a remarkably small amount of code.