*AutoLISP Functions*

# wcmatch

**Performs a wild-card pattern match on a string**

```
(wcmatch string pattern)
```

## Arguments

**string**

A string to be compared. The comparison is case-sensitive, so uppercase and lowercase characters must match.

**pattern**

A string containing the pattern to match against *string*. The *pattern* can contain the wild-card pattern-matching characters shown in the table Wild-card characters. You can use commas in a pattern to enter more than one pattern condition. Only the first 500 characters (approximately) of the *string* and *pattern* are compared; anything beyond that is ignored.

Both arguments can be either a quoted string or a string variable. It is valid to use variables and values returned from AutoLISP functions for *string* and *pattern* values.

## Return Values

If *string* and *pattern* match, wcmatch returns T; otherwise, wcmatch returns nil.

**Wild-card characters**

| Character | Definition |
|---|---|
| # (pound) | Matches any single numeric digit. |
| @ (at) | Matches any single alphabetic character. |
| . (period) | Matches any single nonalphanumeric character. |
| * (asterisk) | Matches any character sequence, including an empty one, and it can be used anywhere in the search pattern: at the beginning, middle, or end. |
| ? (question mark) | Matches any single character. |
| ~ (tilde) | If it is the first character in the pattern, it matches anything except the pattern. |
| [...] | Matches any one of the characters enclosed. |
| [~...] | Matches any single character not enclosed. |
| – (hyphen) | Used inside brackets to specify a range for a single character. |
| , (comma) | Separates two patterns. |
| ` (reverse quote) | Escapes special characters (reads next character literally). |

## Examples

The following command tests a string to see if it begins with the character N:

```
Command: (wcmatch "Name" "N*")
T
```

The following example performs three comparisons. If any of the three pattern conditions is met, wcmatch returns T. The tests are:

- Does the string contain three characters?
- Does the string not contain an m?
- Does the string begin with the letter “N”?

If any of the three pattern conditions is met, wcmatch returns T:

```
Command: (wcmatch "Name" "???,~*m*,N*")
T
```

In this example, the last condition was met, so wcmatch returned T.

## Using Escape Characters with wcmatch

To test for a wild-card character in a string, you can use the single reverse-quote character (`) to *escape* the character. *Escape* means that the character following the single reverse quote is not read as a wild-card character; it is compared at its face value. For example, to search for a comma anywhere in the string “Name”, enter the following:

```
Command: (wcmatch "Name" "*`,*")
nil
```

Both the C and AutoLISP programming languages use the backslash (\) as an escape character, so you need two backslashes (\\) to produce one backslash in a string. To test for a backslash character anywhere in “Name”, use the following function call:

```
Command: (wcmatch "Name" "*`\\*")
nil
```

All characters enclosed in brackets ([ . . . ]) are read literally, so there is no need to escape them, with the following exceptions: the tilde character (~) is read literally only when it is not the first bracketed character (as in "[A~BC]"); otherwise, it is read as the negation character, meaning that wcmatch should match all characters except those following the tilde (as in "[~ABC]"). The dash character (–) is read literally only when it is the first or last bracketed character (as in "[–ABC]" or "[ABC–]") or when it follows a leading tilde (as in "[~-ABC]"). Otherwise, the dash character (–) is used within brackets to specify a range of values for a specific character. The range works only for single characters, so "STR[1–38]" matches STR1, STR2, STR3, and STR8, and "[A–Z]" matches any single uppercase letter.

The closing bracket character (]) is also read literally if it is the first bracketed character or if it follows a leading tilde (as in "[ ]ABC]" or "[~]ABC]").

**Note** Because additional wild-card characters might be added in future releases of AutoLISP, it is a good idea to escape all nonalphanumeric characters in your pattern to ensure upward compatibility.