MULTI-ALIST

By Ron Kaplan

This document was last edited in January 2025.

MULTI-ALIST provides a collection of macros that make it easy to store and retrieve items in a hierarchical, multi-level association list indexed by an arbitrary number of keys. The macro PUTMULTI adds a new value to a list of indexed items and GETMULTI returns the items at that index.

(PUTMULTI PLACE KEY_1...KEY_n VAL) [Macro]

PLACE is a commonLIsp place, a form that can be passed as the first argument of CL:SETF. KEY_1 through KEY_n are the indexing keys and VAL is a new value to be pushed at the bottom of the multi-alist. The value of PUTMULTI is VAL. For example, if the variable WINES is initialized to NIL and the following expressions are evaluated

```
(PUTMULTI WINES 'RED 'DRY 'FRENCH 'CABERNET)
(PUTMULTI WINES 'RED 'DRY 'FRENCH 'PINOT-NOIR)
(PUTMULTI WINES 'WHITE 'DRY 'FRENCH 'CHARDONNAY)
```

the value of WINES will be the multi-alist

```
((RED (DRY (FRENCH CABERNET PINOT-NOIR)))
 (WHITE (DRY (FRENCH CHARDONNAY)))
```

The macro GETMULTI retrieves the value stored under a sequence of keys anchored at PLACE:

(GETMULTI PLACE KEY_1...KEY_n) [Macro]

For the wine example the expression

```
(GETMULTI WINES 'RED 'DRY 'FRENCH)
```

will return the list (PINOT-NOIR CABERNET).

The expression (PUTMULTI WINES 'RED 'DRY 'FRENCH 'CABERNET) effectively expands to the fragment

```
(LET (TEMP)
   (LOCALVARS TEMP)
   (SETQ TEMP (OR (SASSOC 'RED WINES)
                  (CAR (CL:PUSH (CONS  'RED) WINES))))
   (SETQ TEMP (OR (SASSOC 'DRY (CDR TEMP))
                  (CAR (CL:PUSH (CONS 'DRY) (CDR TEMP)))))
   (SETQ TEMP (OR (SASSOC 'FRENCH (CDR TEMP))
                  (CAR (CL:PUSH (CONS 'FRENCH) (CDR TEMP)))))
   (CL:PUSH 'CABERNET (CDR TEMP))
   'CABERNET
```

and (GETMULTI WINDOW 'RED 'DRY 'FRENCH) effectively expands to

```
(LET (TEMP)
   (LOCALVARS TEMP)
   (SETQ TEMP (CDR (SASSOC 'RED WINES)))
   (SETQ TEMP (CDR (SASSOC 'DRY TEMP)))
   (CDR (SASSOC 'FRENCH TEMP))
```

PUTMULTI-D and PUTMULTI-NEW are variations on this basic scheme.

(PUTMULTI-D PLACE KEY_1...KEY_n VAL) [Macro]

Uses RPLACD instead of CL:PUSH to destructively store VAL as the only value at the bottom of the multi-alist. Any previous value(s) located by those keys are replaced by the new value. As an example, an entry could be added to the \FONTSINCORE database by

```
(PUTMULTI-D \FONTSINCORE FAMILY SIZE FACE ROTATION DEVICE FONT)
```

and retrieved by

```
(GETMULTI \FONTSINCORE FAMILY SIZE FACE ROTATION DEVICE)
```

(PUTMULTI-NEW PLACE KEY_1...KEY_n VAL) [Macro]

The value VAL is added to the multi-alist only if it is not already a MEMBER of the values at the bottom.

Fast versions of these macros use FASSOC instead of SASSOC to traverse through the structure. All indexing keys are thus matched by EQ.

(FPUTMULTI PLACE KEY_1...KEY_n VAL) [Macro]
(FPUTMULTI-D PLACE KEY_1...KEY_n) [Macro]
(FPUTMULTI-NEW PLACE KEY_1...KEY_n VAL) [Macro]

(FGETMULTI PLACE KEY_1...KEY_n VAL) [Macro]

The macro ADDTOMULTI pushes a value at the bottom of the list, like PUTMULTI, but the keys are provided as a precomputed list and not spread out as individual arguments.

(ADDTOMULTI PLACE KEYS VAL) [Macro]

There are rudimentary macros for computing histograms and cross-tabulations:

(PUTMULTI-COUNT PLACE KEY_1...KEY_n) [Macro]

The value located by KEY_1...KEY_n is the number of times that PUTMULTI-COUNT has been evaluated in that place and with those keys.

(PUTMULTI-SUM PLACE KEY_1...KEY_n NUM) [Macro]

NUM must be a number. The value located by the keys is the running sum of numbers "inserted" with those keys.

Items can be removed from a multi-alist by the macros REMOVEMULTI and REMOVMULTIALL:

(REMOVEMULTI PLACE KEY_1...KEY_n VAL) [Macro]

Removes VAL from the location indexed by the keys.

(REMOVEMULTIALL PLACE KEY_1...KEY_n) [Macro]

Removes all values at the location indexed by KEY_1...KEY_n.

The function MAPMULTI iterates through paths in a multi-alist.

(MAPMULTI MULTIALIST MAPFN) [Function]

The m-ary function MAPFN is applied to each m-length successive substructure of MULTIALIST. Thus, if F is a 4 argument function and (MAPMULTI WINES F) is evaluated, then on one application F will receive the arguments RED DRY FRENCH CABERNET. The arity m=4 in this case covers the 3 keys and the final value of each path. For m less than 3 the last argument will be one entry in the trailing sub-alist of the structure after a prefix of initial keys. Thus for m=3 the function would receive

```
RED DRY (FRENCH ZINFANDEL CABERNET
```

at one of its applications.

(COLLECTMULTI MULTIALIST COLLECTFN) [Function]

This binds the special variable $$COLLECT, calls (MAPMULTI MULTIALIST COLLECTFN), and returns the final value of $$COLLECT. The value will contain any items that the mapping function decides to push onto $$COLLECT.