**Multivalue eval functions**

The following list contains the functions that you can use on multivalue fields or to return multivalue fields.

For information about using string and numeric fields in functions, and nesting functions, see Evaluation functions.

**commands(X)**

*Description*

This function takes a search string, or field that contains a search string, X and returns a multivalued field containing a list of the commands used in X.

*Usage*

This function is generally not recommended for use except for analysis of `audit.log` events.

You can use this function with the `eval`, `fieldformat`, and `where` commands, and as part of eval expressions.

*Basic example*

The following example returns a multivalued field X, that contains 'search', 'stats', and 'sort'.

```plaintext
... | eval x=commands("search foo | stats count | sort count")
```

**mvappend(X,...)**

*Description*

This function takes an arbitrary number of arguments and returns a multivalue result of all the values. The arguments can be strings, multivalue fields or single value fields.

*Usage*

You can use this function with the `eval`, `fieldformat`, and `where` commands, and as part of eval expressions.

*Basic examples*

This example shows how to append two values, `localhost` is a literal string value and `srcip` is a field name.

```plaintext
... | eval fullName=mvappend("localhost", srcip)
```

This example shows how to use nested `mvappend` functions.

- The inner `mvappend` function contains two values: `localhost` is a literal string value and `srcip` is a field name.
- The outer `mvappend` function contains three values: the inner `mvappend` function, `destip` is a field name, and `192.168.1.1` which is a literal IP address.
... | eval ipaddresses=mvappend(mvappend("localhost", srcip), destip, "192.168.1.1")

The results are placed in a new field called ipaddresses which contains the array ["localhost", <values_in_scrip>, <values_in_destip>, "192.168.1.1"].

mvcount(MVFIELD)

Description

This function takes a field and returns a count of the values in that field for each result. If the field is a multivalue field, returns the number of values in that field. If the field contains a single value, this function returns 1. If the field has no values, this function returns NULL.

Usage

You can use this function with the eval, fieldformat, and where commands, and as part of eval expressions.

Basic example

... | eval n=mvcount(multifield)

Extended example

In the following example, the mvcount() function returns the number of email addresses in the To, From, and Cc fields and saves the addresses in the specified "_count" fields.

```
eventtype="sendmail" | eval To_count=mvcount(split(To,"@"))-1 | eval From_count=mvcount(From) | eval Cc_count= mvcount(split(Cc,"@"))-1
```

This search takes the values in the To field and uses the split function to separate the email address on the @ symbol. The split function is also used on the Cc field for the same purpose.

If only a single email address exists in the From field, as you would expect, mvcount(From) returns 1. If there is no Cc address, the Cc field might not exist for the event. In that situation mvcount(cc) returns NULL.

mvdedup(X)

Description

This function takes a multivalue field X and returns a multivalue field with its duplicate values removed.

Usage

You can use this function with the eval, fieldformat, and where commands, and as part of eval expressions.

Basic example

... | eval s=mvdedup(mvfield)
mvfilter(X)

Description

This function filters a multivalue field based on an arbitrary Boolean expression X. The Boolean expression X can reference ONLY ONE field at a time.

Usage

This function will return NULL values of the field x as well. If you do not want the NULL values, use one of the following expressions:

- mvfilter(!isnull(x))
- mvfilter(isnotnull(x))

You can use this function with the eval, fieldformat, and where commands, and as part of eval expressions.

Basic examples

The following example returns all of the values in field email that end in .net or .org.

```
... | eval n=mvfilter(match(email, "\..net$") OR match(email, "\..org$"))
```

mvfind(MVFIELD,"REGEX")

Description

This function tries to find a value in the multivalue field MVFIELD that matches the regular expression in "REGEX". If a match exists, the index of the first matching value is returned (beginning with zero). If no values match, NULL is returned.

Usage

You can use this function with the eval, fieldformat, and where commands, and as part of eval expressions.

Basic example

```
... | eval n=mvfind(mymvfield, "err\d+")
```

mvindex(MVFIELD,STARTINDEX, ENDINDEX)

Description

This function takes two or three arguments and returns a subset of the multivalue field using the index values provided. The field MVFIELD and the number STARTINDEX are required. The number ENDINDEX is inclusive and optional.

Usage

You can use this function with the eval, fieldformat, and where commands, and as part of eval expressions.

Indexes start at zero. If you have 5 values in the multivalue field, the first value has an index of 0. The second values has an index of 1.
Both the STARTINDEX and ENDINDEX arguments can be negative, where -1 is the last element.

If ENDINDEX is not specified, the function returns only the value at STARTINDEX.

If the indexes are out of range or invalid, the result is NULL.

**Basic examples**

Because indexes start at zero, the following example returns the third value in "multifield", if the value exists.

```
... | eval n=mvindex(multifield, 2)
```

**Extended example**

The following search displays at most the last 10 values in the <field>.

The STARTINDEX is a range, that starts with the last value, -1. The range is the last 10 values, -1-10. The ENDINDEX is -1, which returns the last value in the field.

- If the multivalue field has 20 values, only the last 10 values are returned.
- If the multivalue field has 3 values, only 3 values are returned.

```
... | eval keep=mvindex(<field>,-1-10,-1)
```

**mvjoin(MVFIELD,STR)**

**Description**

This function takes two arguments, a multivalue field (MVFIELD) and a string delimiter (STR). The function concatenates the individual values within MVFIELD using the value of STR as a separator.

**Usage**

You can use this function with the `eval`, `fieldformat`, and `where` commands, and as part of eval expressions.

**Basic examples**

You have a multivalue field called "base" that contains the values "1" "2" "3" "4" "5". The values are separated by a space. You want to create a single value field instead, with OR as the delimiter. For example "1 OR 2 OR 3 OR 4 OR 5".

The following search creates the `base` field with the values. The search then creates the `joined` field by using the result of the `mvjoin` function.

```
... | eval base=mvrange(1,6), joined=mvjoin('base',' OR ')
```

The following example joins together the individual values of "foo" using a semicolon as the delimiter:

```
... | eval n=mvjoin(foo, ";")
```
mvmap(X,Y)

_Description_

This function iterates over the values of a multi-value field (X), performs an operation (Y) on each value, and returns a multi-value field with the list of results.

- X is a multi-value expression that references a single field.
- Y is a result expression.

_Useage_

You can use this function with the `eval`, `fieldformat`, and `where` commands, and as part of eval expressions.

**Basic examples**

The following example multiplies each value in foo by 10.

```
... | eval foo = mvmap(foo, foo*10)
```

The following example multiplies each value of foo by bar, where bar is a single-valued field.

```
... | eval foo = mvmap(foo, foo*bar)
```

The following example multiplies the 2nd and 3rd values of foo by bar, where bar is a single-value field.

```
... | eval foo = mvmap(mvindex(foo,1,2), foo*bar)
```

mvrangex,Y,Z)

_Description_

This function creates a multivalue field for a range of numbers. This function can contain up to three arguments: a starting number X, an ending number Y (which is excluded from the field), and an optional step increment Z. If the increment is a timespan such as `7d`, the starting and ending numbers are treated as UNIX time.

_Useage_

You can use this function with the `eval`, `fieldformat`, and `where` commands, and as part of eval expressions.

**Basic examples**

The following example returns a multivalue field with the values 1, 3, 5, 7, 9.

```
... | eval mv=mvrangex,11,2
```

The following example takes the UNIX timestamp for 1/1/2018 as the start date and the UNIX timestamp for 4/19/2018 as
an end date and uses the increment of 7 days.

| makeresults | eval mv=mvrange(1514834731,1524134919,"7d")

This example returns a multivalue field with the UNIX timestamps. The results appear on the Statistics tab and look something like this:

<table>
<thead>
<tr>
<th>_time</th>
<th>mv</th>
</tr>
</thead>
<tbody>
<tr>
<td>2018-04-10 12:31:03</td>
<td>1514834731, 1515439531, 1516044331, 1516649131, 1517253931, 1517858731, 1518463531, 1519068331, 1519673131, 1520277931, 1520879131, 1521483931, 1522088731, 1522693531, 1523298331, 1523903131</td>
</tr>
</tbody>
</table>

**mvsort(X)**

**Description**

This function uses a multivalue field X and returns a multivalue field with the values sorted lexicographically.

**Usage**

You can use this function with the `eval`, `fieldformat`, and `where` commands, and as part of eval expressions.

Lexicographical order sorts items based on the values used to encode the items in computer memory. In Splunk software, this is almost always UTF-8 encoding, which is a superset of ASCII.

- Numbers are sorted before letters. Numbers are sorted based on the first digit. For example, the numbers 10, 9, 70, 100 are sorted lexicographically as 10, 100, 70, 9.
- Uppercase letters are sorted before lowercase letters.
- Symbols are not standard. Some symbols are sorted before numeric values. Other symbols are sorted before or after letters.

**Basic example**

... | eval s=mvsort(mvfield)
mvzip(X,Y,"Z")

Description
This function takes two multivalue fields, X and Y, and combines them by stitching together the first value of X with the first value of field Y, then the second with the second, and so on. The third argument, Z, is optional and is used to specify a delimiting character to join the two values. The default delimiter is a comma.

Usage
This is similar to the Python zip command.

You can use this function with the eval, fieldformat, and where commands, and as part of eval expressions.

Basic example
... | eval nserver=mvzip(hosts,ports)

Extended example
You can nest several mvzip functions together to create a single multivalued field three_fields from three separate fields. The pipe (|) character is used as the separator between the field values.

...| eval three_fields=mvzip(mvzip(field1,field2,"|"),field3,"|")

(Thanks to Splunk user cmerriman for this example.)

split(X,"Y")

Description
This function takes two arguments, field X and delimiting character Y. It splits the values of X on the delimiter Y and returns X as a multivalue field.

Usage
You can use this function with the eval, fieldformat, and where commands, and as part of eval expressions.

The Splunk software includes a set of multivalue functions. See Multivalue eval functions and Multivalue stats and chart functions.

Basic example
... | eval n=split(foo, ";")

See also
See the following multivalue commands:
makemv, mvcombine, mvexpand, nomv