



Splunk® Enterprise 8.2.0

搜索参考

生成时间: 2021 年 5 月 24 日, 14:40

Table of Contents

简介	6
欢迎使用搜索参考	6
了解 SPL 语法	6
如何使用本手册	9
快速参考	11
Splunk 快速参考指南	11
命令快速参考	11
命令分类	15
命令类型	22
面向 SQL 用户的 Splunk SPL	25
SPL 数据类型和子句	28
评估函数	32
评估函数	32
对比和条件函数	38
转换函数	46
加密函数	50
日期和时间函数	51
信息函数	55
JSON 函数	58
数学函数	70
多值 eval 函数	74
统计 eval 函数	79
文本函数	81
三角函数和双曲函数	84
统计和图表函数	89
统计和图表函数	89
聚合函数	92
事件顺序函数	109
多值统计和 chart 函数	113
时间函数	115
时间格式变量和修饰符	123
日期和时间格式变量	123
时间调节器	125
搜索命令	129
abstract	129
accum	129
addcoltotals	130
addinfo	132
addtotals	133
analyzefields	136
anomalies	137
anomalousvalue	140
anomalydetection	143
append	145
appendcols	148
appendpipe	149
arules	150
associate	151
audit	153
autoregress	154
awssnsalert	155
bin	155
bucket	157
bucketdir	157
cefout	158
chart	158
cluster	169
cofilter	171
collect	173
concurrency	176
contingency	179

convert	182
correlate	185
ctable	186
datamodel	186
datamodelsimple	189
dbinspect	189
dbxquery	192
dedup	192
delete	194
delta	196
diff	199
entitymerge	200
erex	200
Eval	203
eventcount	209
eventstats	211
extract	216
fieldformat	218
字段	221
fieldsummary	223
filldown	224
fillnull	225
findtypes	227
folderize	228
foreach	230
format	232
from	233
gauge	235
gentimes	237
geom	239
geomfilter	242
geostats	243
head	247
highlight	249
history	250
iconify	252
inputcsv	253
inputintelligence	255
inputlookup	255
iplocation	258
join	261
kmeans	263
kvform	265
loadjob	266
localize	268
localop	269
lookup	269
makecontinuous	273
makemv	274
makeresults	275
map	280
mcollect	281
metadata	284
metasearch	287
meventcollect	288
mpreview	291
msearch	293
mstats	293
multikv	302
multisearch	303
mvcombine	304
mvexpand	307
nomv	309
outlier	310
outputcsv	311
outputlookup	312
outputtext	315
overlap	316
pivot	317
predict	319
rangemap	324

rare	326
redistribute	327
regex	331
relevancy	332
reltime	333
rename	333
replace	334
require	336
rest	336
return	338
reverse	340
rex	340
rtorder	344
run	344
savedsearch	344
script	345
scrub	346
搜索	347
searchtxn	352
selfjoin	353
sendemail	357
设置	360
setfields	362
sichart	362
sirare	363
sistats	364
sitimechart	365
sitop	366
snowincident	367
snowincidentstream	367
snowevent	368
snoweventstream	368
sort	368
spath	371
stats	375
strcat	383
streamstats	384
table	391
tags	393
tail	396
timechart	396
timewrap	407
tojson	410
top	413
transaction	416
transpose	424
trendline	427
tscollect	428
tstats	429
typeahead	435
typelearner	436
typer	437
union	438
uniq	441
untable	442
walklex	445
where	447
x11	449
xmlkv	450
xmlunescape	451
xpath	452
xsDisplayConcept	453
xsDisplayContext	454
xsFindBestConcept	454
xsListConcepts	454
xsListContexts	454
xsUpdateDDContext	454
xsWhere	454
xyseries	454
第三方自定义命令	458

内部命令	459
关于内部命令	459
collapse	459
dump	459
findkeywords	460
makejson	461
mcatalog	464
noop	467
prjob	470
runshellscript	472
sendalert	473
在 CLI 中搜索	475
关于在 CLI 中搜索	475
CLI 中的搜索语法	475

简介

欢迎使用搜索参考

本手册是“搜索处理语言”(SPL)的参考指导。在此手册中，您将找到带有完整语法、描述和示例的搜索命令目录。此外，本手册还包含有关命令类别的快速参考信息、可与命令一起使用的函数，以及 SPL 与 SQL 之间的关系。

入门

如果您之前未使用过 Splunk 软件和搜索，可以从《搜索教程》开始。此教程将介绍“搜索和报表”应用程序。教程将通过上传数据至 Splunk 部署、搜索数据以及建立简单图表、报表和仪表板进行逐步指导。

在您完成搜索教程后，开始对自己的数据使用 Splunk 软件之前，您应该：

- 将数据添加到 Splunk 实例。请参阅[数据导入](#)。
- 了解建立索引的过程，以及数据的处理方式。请参阅[《管理索引器和索引器上的群集》](#)。
- 了解字段和知识对象，如主机、来源类型和事件类型。请参阅[《知识管理器手册》](#)。

搜索手册

《搜索手册》是《搜索参考》的配套手册。《搜索手册》包含有关新建和优化搜索的详细信息。

- 搜索类型
- 检索事件
- 指定时间范围
- 优化搜索
- 使用子搜索
- 新建统计表格和图表
- 事件分组和相关性
- 预测将来事件
- 管理任务

快速参考信息

快速参考指南包含：

- Splunk 功能的解释说明
- 常用搜索命令
- 搜索优化提示
- eval 和 stats 命令的函数
- 搜索示例
- 正则表达式
- 将字符串转化为时间戳的格式

SPL 命令

搜索处理语言(SPL)包括一系列的命令。

这些命令有两个快速参考指南：

- “命令快速参考”主题包含按字母顺序排列的每个命令的列表，以及命令功能的简要描述及指向命令特定文档的链接。
- “命令分类”主题按命令所执行操作的类型组织命令。本主题包含命令功能的简要描述及指向命令特定文档的链接。

命令语法

在继续之前，请先参阅“了解 SPL 语法”，了解本手册中采用的约定和规则。

了解 SPL 语法

以下各节介绍用于 Splunk SPL 命令的语法。有关使用关键字、短语、通配符和正则表达式的其他信息，请参阅“[搜索命令入门](#)”。

必要参数和可选参数

SPL 命令由必要参数和可选参数组成。

- 必要参数显示在尖括号 < > 中。
- 可选参数由方括号 [] 括起。

以此命令语法为例：

```
bin [<bins-options>...] <field> [AS <newfield>]
```

必要参数为 <field>。要使用此命令，您必须至少指定 bin <field>。

可选参数为 [<bins-options>...] 和 [AS <newfield>]。

用户输入参数

以此命令语法为例：

```
replace (<wc-string> WITH <wc-string>)... [IN <field-list>]
```

用户输入参数为：<wc-string> 和 <field-list>。

重复的参数

某些参数可以指定多次。语法显示省略号 ... 以指定参数中哪个部分可以重复。省略号始终紧跟在语法中可以重复的部分之后。

以此命令为例：

```
convert [timeformat=string] (<convert-function> [AS <field>])...
```

必要参数为 <convert-function>，附带一个可以使用 [AS <field>] 子句指定字段的选项。

请留意语法句末的省略号，就在圆括号后面。本示例中，圆括号内的语法是可以重复的 <convert-function> [AS <field>]。

以下语法中，您可以重复 <bins-options>...。

```
bin [<bins-options>...] <field> [AS <newfield>]
```

参数组

有时，语法必须以组的形式显示参数，以说明该组参数需一起使用。圆括号 () 用于参数分组。

以此语法为例：

```
replace (<wc-string> WITH <wc-string>)... [IN <field-list>]
```

参数组为 (<wc-string> WITH <wc-string>)...。这是一个必要的参数组，可以重复多次。

Keywords

很多命令在部分参数或选项中使用关键字。关键字示例包括：

- AS
- BY
- OVER
- WHERE

您在搜索时可以指定这些关键字的大小写。不过，出于可读性考量，Splunk 文档中的语法若含关键字，全部采用大写形式。

加引号的元素

如果元素带引号，您必须将该元素包含在搜索中。最常见的引用元素是括号。

想一想 chart 命令的语法：

```
chart [<chart-options>] [agg=<stats-agg-term>]
( <stats-agg-term> | <sparkline-agg-term> | "<eval-expression>" )...
[ BY <row-split><column-split> ] | [ OVER <row-split> ] [BY <column-split>] ]
```

<eval-expression> 两边的括号上有引号。这表示您必须将搜索中的 <eval-expression> 括上括号。

在以下搜索示例中，<eval-expression> 是 avg(size)/max(delay)，且包含在括号中。

```
... | chart eval(avg(size)/max(delay)) AS ratio BY host user
```

参数顺序

命令语法中的命令参数按其使用顺序排列。

在参数描述中的必要参数和可选参数部分，参数是按字母顺序排列的。每个参数都有语法和描述。此外，“可选参数”可能还有一个默认值。

数据类型

SPL 语法中数据类型所用的命名规则在下表中有说明。

语法	数据类型	注释
<bool>	布尔值	使用 true 或 false。亦可使用其他变量。例如，true 也可以改用 't'、'T'、'TRUE'、'yes' 或数字 '1'。您可将 false 指定为 'no'、数字零（0）以及单词 false 变形，类似单词 true 的变形。
<field>	字段名称。无法为字段名称指定通配符。	请参阅 <wc-field>。
<int> 或 <integer>	整数既可以是正整数也可以是负整数。	有时指“有符号的”整数。请参阅 <unsigned int>。
<string>	字符串	请参阅 <wc-string>。
<unsigned int>	无符号的整数	无符号的整数必须是正值。无符号的整数可大于有符号的整数。
<wc-field>	字段名称或带有通配符的部分名称，用于指定多个名称类似的字段。	使用星号（*）字符作为通配符。
<wc-string>	字符串值或带有通配符的部分字符串值。	使用星号（*）字符作为通配符。

布尔运算符

若一命令的语法中包含布尔运算符，您必须始终以大写形式指定该运算符。布尔运算符包括：

- 且
- 或
- NOT

要了解有关评估布尔表达式的顺序的详细信息以及一些示例，请参阅《搜索手册》中的“布尔表达式”。

要了解有关 NOT 运算符的更多信息，请参阅《搜索手册》中的“NOT 和 != 间的差异”。

BY 子句

<by-clause> 和 <split-by-clause> 是不同的参数。

当您使用 <by-clause> 时，将为 <by-clause> 字段中的各不同值返回一行。<by-clause> 以单独成行的方式显示每个唯一项目。请将 <by-clause> 视为分组。

<split-by-clause> 则以单独成列的方式显示每个唯一项目。请将 <split-by-clause> 视为拆分或分割。

在 BY 子句中不接受通配符（*）。

字段和通配符字段

当语法包含 `<field>` 时，您从事件中指定字段名称。

以此语法为例：

```
bin [<bins-options>...] <field> [AS <newfield>]
```

`<field>` 参数是必需的。您可以使用 `[AS <newfield>]` 参数指定该字段在搜索结果中显示不同的名称。这是可选参数。

例如，如果字段为 `categoryId`，并且您希望该字段在输出中命名为 `CategoryID`，那么您将指定：

```
categoryId AS CategoryID
```

`<wc-field>` 参数指示，您可以在指定字段名称的时候使用通配符字符。例如，若有一组字段以 "log" 结尾，您可以指定 `*log` 以返回所有的此类字段。

如果在值的中间使用通配符字符，特别是作为通配符用于标点符号，结果可能不可预测。

另请参阅

在搜索手册中：

- 搜索剖析
- 通配符
- 字段表达式
- 引号和转义字符

如何使用本手册

本手册可用作面向需要搜索命令目录以及完整语法、描述和用法示例的 Splunk 用户的参考指南。

快速参考信息

这些命令有两个快速参考指南：

- “命令快速参考” 主题包含按字母顺序排列的每个命令的列表，以及命令功能的简要描述及指向命令特定文档的链接。
- “命令分类” 主题按命令所执行操作的类型组织命令。本主题包含命令功能的简要描述及指向命令特定文档的链接。

函数

命令主题

每个搜索命令主题都包含以下各部分：描述、语法、示例及另请参阅。很多命令主题还有一个“使用”部分。

描述

描述命令的用途。此部分可能包括命令用法的详细信息。对于复杂命令，可能会有单独的“用法”部分。

语法

语法部分包括每个搜索命令的完整语法以及每个参数的描述。部分命令的参数包含一组待指定的选项。每组选项均遵循参数描述。

必要参数

显示语法并描述必要参数。

可选参数

显示语法并描述可选参数。如适用，还会列出默认值。

用法

包含使用该命令的额外信息。

示例

此部分包括命令用法的示例。

另请参阅

此部分中包含所有相关或相似命令的链接。

命令语法约定

语法中的命令参数将按其使用顺序列出。

参数分为“必要”参数和“可选”参数，在相应子标题下按字母顺序列出。每个参数都有语法和描述部分。另外，也可能会有其他部分，如给出参数相关信息的默认部分。

请参阅“了解 SPL 语法”。

格式约定

斜体

当提及 Splunk 文档集中的另一本手册时，该手册名以斜体显示。

快速参考

Splunk 快速参考指南

Splunk 快速参考指南是一个六页的参考卡，其中包含了基本搜索概念、命令、函数和示例。可在线阅读此指南的 PDF 版文件。

注意：在快速参考指南的示例中，前导省略号 (...) 表示管道符之前有一个搜索。前导管道指示该搜索命令是一个生成命令，会阻止命令行界面和 Splunk Web 在您的搜索前面加上 search 命令。

另请参阅

- 搜索命令分类

Splunk Answers

如果在此搜索语言参考中未找到要查找的内容，请查看 Splunk Answers 并查看其他 Splunk 用户使用搜索语言时的问题和回答。

命令快速参考

下方的表格将以字母顺序列出所有的搜索命令。表格内有命令的简短描述和相关命令的链接。如需查看完整语法、使用和详细示例，单击命令名称即可显示该命令的特定主题。

部分命令共用相同的函数。如需查看函数列表及相应的描述和示例，请参阅“评估函数”和“统计和图表函数”。

如果在表格中没有找到某一命令，该命令可能是第三方应用程序或加载项的一部分。有关应用程序和加载项提供的命令的信息，请参阅 Splunkbase 文档。

命令	描述	相关命令
abstract	生成每个搜索结果的汇总。	highlight
accum	保留指定数值型字段的累计值。	autoregress, delta, trendline, streamstats
addcoltotals	计算包含先前事件的所有数值字段值总和的事件。	addtotals, stats
addinfo	添加包含当前搜索的常用信息的字段。	search
addtotals	计算每个结果的所有数值字段值的总和。	addcoltotals、 stats
analyzefields	分析数值字段，以确定其预知另一个离散字段的能力。	anomalousvalue
anomalies	计算事件的 “unexpectedness” 分数。	anomalousvalue、 cluster、 kmeans、 outlier
anomalousvalue	查找并汇总不规则或不常见的搜索结果。	analyzefields、 anomalies、 cluster、 kmeans outlier
anomalydetection	先计算每个事件的概率然后再检测异常低的概率值，由此来识别异常事件。	analyzefields、 anomalies、 anomalousvalue、 cluster、 kmeans outlier
append	将子搜索结果附加到当前结果。	appendcols、 appendcsv、 appendlookup、 join set
appendcols	将子搜索结果的字段附加到当前结果，第一组子搜索结果对应第一个结果，第二组对应第二个，依此类推。	append、 appendcsv、 join、 set
appendpipe	将应用于当前结果集的子管道的结果附加到结果。	append、 appendcols、 join、 set
arules	查找字段值之间的关联规则。	associate、 correlate
associate	标识字段间的相关性。	correlate、 contingency
audit	返回本地审计索引中存储的审计线索信息。	
autoregress	设置数据，以计算移动平均值。	accum、 autoregress、 delta、 trendline streamstats
bin (bucket)	将连续的数字值放入离散集中。	chart、 timechart
bucketdir	用更高级别的分组替换字段值，例如，用目录替换文件名。	cluster、 dedup

chart	以表的形式返回结果以绘制图表。还可以直接使用内置图表函数。	bin、sichart、timechart
cluster	将相似事件组成群集。	anomalies、anomalousvalue、cluster、kmeans outlier
cofilter	查找 field1 和 field2 值一起出现的次数。	associate、correlate
collect	将搜索结果放入摘要索引。	overlap
concurrency	使用 duration 字段来查找每个事件的并发事件的数量。	timechart
contingency	构建两个字段的应变表。	associate、correlate
convert	将字段值转换为数字值。	eval
correlate	计算不同字段之间的相关性。	associate、contingency
datamodel	检查数据模型或数据模型数据集，并搜索数据模型数据集。	pivot
dbinspect	返回关于指定索引的信息。	
dedup	删除与指定条件相匹配的后续结果。	uniq
delete	删除特定事件或搜索结果。	
delta	计算邻近结果之间字段值之差。	accum、autoregress、trendline、streamstats
diff	返回两个搜索结果之间的差。	
erex	允许您指定示例或计数器示例值，以自动提取具有相似值的字段。	extract、kvform、multikv、regex、rex xmlkv
eval	计算表达式并将生成的值放入字段中。还可以参阅“评估函数”。	where
eventcount	返回索引中的事件数。	dbinspect
eventstats	向所有搜索结果添加摘要统计信息。	stats
extract (kv)	从搜索结果提取字段-值对。	kvform、multikv、xmlkv、rex
fieldformat	表示在不更改底层值的情况下如何在输出时显示某个字段。	eval、where
fields	删除搜索结果中的字段。	
fieldsummary	为所有字段或部分字段生成摘要信息。	analyzefields、anomalies、anomalousvalue、stats
filldown	使用最后的非空值替换空值。	fillnull
fillnull	使用指定值替换空值。	
findtypes	生成建议的事件类型列表。	typer
folderize	新建更高级别的分组，例如，用目录替换文件名。	
foreach	为通配符字段列表中的每个字段运行模板化流子搜索。	eval
format	获取子搜索的结果，并将结果格式化为单个结果。	
from	从数据集（例如数据模型数据集、CSV 查找、KV 存储查找、保存的搜索或表数据集）检索数据。	
gauge	将结果转换为适合由仪表图类型显示的格式。	
gentimes	生成时间范围结果。	
geom	为每个事件添加一个名为 geom 的字段。此字段包含 JSON 中多边形几何体的地理数据结构，用于地区分布图可视化。	geomfilter
geomfilter	接受两个点，这两个点可以指定剪切地区分布图的边框。落在边框以外的点会被过滤掉。	geom
geostats	生成将在世界地图上呈现且群集化成地理数据箱的统计信息。	stats, xyseries
head	返回前 n 个指定结果。	reverse, tail

highlight	突出显示指定的术语。	iconify
history	以事件列表或表格形式返回搜索历史。	search
iconify	针对您所指定字段列表中的所有不同值，显示唯一的图标。	highlight
inputcsv	从指定 CSV 文件加载搜索结果。	loadjob, outputcsv
inputlookup	从指定的静态查找表加载搜索结果。	inputcsv、join、lookup、outputlookup
iplocation	从 IP 地址提取位置信息。	
join	将子搜索的结果与主搜索的结果合并。	appendcols, lookup, selfjoin
kmeans	对所选字段执行 K 均值群集化。	anomalies, anomalousvalue, cluster, outlier
kvform	使用表单模板从搜索结果中提取值。	extract, kvform, multikv, xmlkv, rex
loadjob	加载先前完成的搜索任务的事件或结果。	inputcsv
localize	返回找到的搜索结果所属时间范围的列表。	map, transaction
localop	在本地而非远程对等节点上运行后续命令（即此命令之后的所有命令）。	
lookup	显式调用字段值查找。	
makecontinuous	生成具有 X 轴连续性的字段（由 chart/timechart 调用）	chart, timechart
makemv	在搜索期间将指定字段更改为多值字段。	mvcombine, mvexpand, nomv
makeresults	新建指定数量的空搜索结果。	
map	循环运算符，对每个搜索结果执行搜索。	
mcollect	将搜索结果转换为指标数据，将数据插入搜索头上的指标索引。	collect, meventcollect
metadata	从指定的索引或分布式搜索节点返回数据来源、来源类型或主机的列表。	dbinspect
metasearch	根据逻辑表达式中的条件从索引检索事件元数据。	metadata, search
meventcollect	将搜索结果转换为指标数据，将数据插入到索引器上的指标索引。	collect, mcollect
mpreview	返回指定指标索引中与提供的筛选器匹配的原始指标数据点的预览。	mcatalog, mstats, msearch
msearch	mpreview 命令的别名。	mcatalog, mstats, mpreview
mstats	计算指标索引中的测量、metric_name 和维度字段的统计数据。	stats, tstats
multikv	从表格形式的事件中提取字段-值。	
multisearch	同时运行多个流搜索。	append, join
mvcombine	将搜索结果中有单个不同字段值的事件合并到一个结果中，该结果具有包含不同字段的多值字段。	mvexpand, makemv, nomv
mvexpand	将多值字段的值扩展到多值字段每个值的不同事件中。	mvcombine, makemv, nomv
nomv	在搜索时间将指定的多值字段更改为单值字段。	makemv, mvcombine, mvexpand
outlier	移除无关的数字值。	anomalies, anomalousvalue, cluster, kmeans
outputcsv	将搜索结果输出到指定的 CSV 文件中。	inputcsv, outputtext
outputlookup	将搜索结果保存到指定的静态查找表中。	inputlookup, lookup, outputcsv
outputtext	将结果的原始文本字段 (_raw) 输出到 _xml 字段中。	outputcsv
overlap	在摘要索引中查找事件，该索引有时间重叠，或具有缺失的事件。	collect

pivot	对特定数据模型数据集运行数据透视表搜索。	datamodel
predict	允许您使用时间系列算法预测字段的未来值。	x11
rangemap	将 RANGE 字段设置为匹配的范围的名称。	
rare	显示最不常用的字段值。	sirare, stats, top
redistribute	执行并行化简搜索处理，以缩短高基数数据集搜索的搜索时间。	
regex	删除与指定正则表达式不匹配的结果。	rex, search
relevancy	计算事件与查询的匹配程度。	
reltime	将 'now' 和 '_time' 之差转换为可读的值，并将该值添加到搜索结果的 'reltime' 字段中。	convert
rename	重命名指定字段；可以使用通配符指定多个字段。	
replace	将指定字段的值替换为指定的新值。	
require	如果搜索字符串中位于搜索字符串前面的查询和命令返回零个事件或结果，则会导致搜索失败。	
rest	访问 REST 端点，并显示返回的实体作为搜索结果。	
return	指定从子搜索返回的值。	format, search
reverse	颠倒结果的顺序。	head, sort, tail
rex	指定 Perl 正则表达式命名的组，以在搜索时提取字段。	extract, kvform, multikv, xmlkv, regex
rtorder	对来自实时搜索的事件进行缓冲，以尽可能按时间顺序的升序发出事件。	
savedsearch	返回保存的搜索的搜索结果。	
script (运行)	在搜索过程中运行外部 Perl 或 Python 脚本。	
scrub	使搜索结果匿名。	
search	搜索匹配事件的索引。	
searchtxm	根据指定的搜索约束查找交易事件。	transaction
selfjoin	将结果与自身连接。	join
sendemail	将搜索结果通过电子邮件发送到指定的电子邮件地址。	
set	对子搜索执行 set 操作 (union, diff, intersect)。	append, appendcols, join, diff
setfields	将所有结果的字段值设置为常用值。	eval, fillnull, rename
sichart	chart 命令的摘要索引版本。	chart, sitimechart, timechart
sirare	rare 命令的摘要索引版本。	rare
sistats	stats 命令的摘要索引版本。	stats
sitimechart	timechart 命令的摘要索引版本。	chart, sichart, timechart
sitop	top 命令的摘要索引版本。	top
sort	按照指定的字段对搜索结果进行排序。	reverse
spath	提供一种直接的方式从结构化数据格式 (XML 和 JSON) 中提取字段。	xpath
stats	提供统计信息，可以选择按字段进行分组。还可以查看统计和图表函数。	eventstats, top, rare
strcat	连接字符串值。	
streamstats	以流化方式向所有搜索结果中添加摘要统计信息。	eventstats, stats

table	使用指定字段新建表。	fields
tags	用标记批注搜索结果中的指定字段。	eval
tail	返回后 n 个指定结果。	head, reverse
timechart	新建时间系列图和相应的统计信息表。还可以查看统计和图表函数。	chart, bucket
timewrap	显示或将 timechart 命令的输出换行，这样时间的每个 timewrap-span 都是不同系列。	timechart
top	显示字段最常见的值。	rare, stats
transaction	将搜索结果分组为交易。	
transpose	将搜索结果的行重新格式化为列。	
trendline	计算字段的移动平均值。	timechart
tscollect	将结果写入 tsidx 文件，供 tstats 命令以后使用。	collect, stats, tstats
tstats	通过 tscollect 命令新建的 tsidx 文件计算统计信息。	stats, tscollect
typeahead	返回指定前缀上的键盘缓冲信息。	
typelearner	已弃用。改用 findtypes。生成建议的 eventtype。	typer
typer	计算搜索结果的 eventtype。	findtypes
union	将两个或两个以上数据集中的结果合并到一个数据集中。	
uniq	删除与之前的结果完全重复的任何搜索。	dedup
untable	把结果从表格形式转换为 stats 输出相似的格式。与 xyseries 和 maketable 颠倒。	
walklex	从每个事件索引数据桶中生成术语或索引字段的列表。	metadata, tstats
where	对数据执行任意筛选。还可以查看“评估函数”。	eval
x11	允许您通过删除季节模式来确定数据中的趋势。	predict
xmlkv	提取 XML 键值对。	extract, kvform, multikv, rex
xmlunescape	取消对 XML 的转义。	
xpath	重新定义 XML 路径。	
xyseries	将结果转换为适用于绘图的格式。	

命令分类

下表列出了所有按使用情况分类的搜索命令。某些命令基于您指定的选项分为多类。

相关性

以下命令可用于构建相关性搜索。

命令	描述
append	将子搜索结果附加到当前结果。
appendcols	将子搜索结果的字段附加到当前结果，第一组子搜索结果对应第一个结果，第二组对应第二个，依此类推。
appendpipe	将应用于当前结果集的子管道的结果附加到结果。
arules	查找字段值之间的关联规则。
associate	标识字段间的相关性。
contingency, counttable, ctable	构建两个字段的应变表。

<code>correlate</code>	计算不同字段之间的相关性。
<code>diff</code>	返回两个搜索结果之间的差。
<code>join</code>	将来自主结果管道的结果与来自子搜索的结果合并。
<code>lookup</code>	显式调用字段值查找。
<code>selfjoin</code>	将结果与自身连接。
<code>set</code>	对子搜索执行 <code>set</code> 操作 (<code>union</code> , <code>diff</code> , <code>intersect</code>)。
<code>stats</code>	提供统计信息，可以选择按字段进行分组。请参阅“统计和图表函数”。
<code>transaction</code>	将搜索结果分组成交易。

数据和索引

可使用以下命令更多地了解数据、添加和删除数据来源或管理摘要索引中的数据。

查看数据

以下命令返回有关索引中数据的信息。它们不会以任何方式修改您的数据或索引。

命令	描述
<code>audit</code>	返回本地审计索引中存储的审计线索信息。
<code>datamodel</code>	返回有关数据模型或数据模型对象的信息。
<code>dbinspect</code>	返回关于指定索引的信息。
<code>eventcount</code>	返回索引中的事件数。
<code>metadata</code>	从指定的索引或分布式搜索节点返回数据来源、来源类型或主机的列表。
<code>typeahead</code>	返回指定前缀上的键盘缓冲信息。

管理数据

以下是一些可用于向您的索引添加数据来源或从索引中删除特定数据的命令。

命令	描述
<code>delete</code>	删除特定事件或搜索结果。

管理摘要索引

以下命令用于新建和管理您的摘要索引。

命令	描述
<code>collect, stash</code>	将搜索结果放入摘要索引。
<code>overlap</code>	在摘要索引中查找事件，该索引有时间重叠，或具有缺失的事件。
<code>sichart</code>	<code>chart</code> 的摘要索引版本。计算稍后对摘要索引运行 <code>chart</code> 搜索所需的信息。
<code>sirare</code>	<code>rare</code> 的摘要索引版本。计算稍后对摘要索引运行 <code>rare</code> 搜索所需的信息。
<code>sistats</code>	<code>stats</code> 的摘要索引版本。计算稍后对摘要索引运行 <code>stats</code> 搜索所需的信息。
<code>sitimechart</code>	<code>timechart</code> 的摘要索引版本。计算稍后对摘要索引运行 <code>timechart</code> 搜索所需的信息。
<code>sitop</code>	<code>top</code> 的摘要索引版本。计算稍后对摘要索引运行 <code>top</code> 搜索所需的信息。

字段

以下是可用于添加、提取和修改字段或字段值的命令。操作字段最有效的命令是 `eval` 及其统计和图表函数。

添加字段

使用以下命令添加新字段。

命令	描述
accum	保留指定数值型字段的累计值。
addinfo	添加包含当前搜索的常用信息的字段。
addtotals	计算每个结果的所有数值字段值的总和。
delta	计算邻近结果之间字段值之差。
eval	计算表达式并将生成的值放入字段中。还可以参阅“评估函数”。
iplocation	添加位置信息，如城市、国家/地区、纬度、经度等（基于 IP 地址）。
lookup	对于已配置的查找表，显式调用字段值查找并从查找表将字段添加到事件。
multikv	从表格形式的事件中提取字段-值。
rangemap	将 RANGE 字段设置为匹配的范围的名称。
relevancy	添加指示事件与查询的匹配程度的 relevancy 字段。
strcat	连接字符串值并将结果保存到指定的字段。

提取字段

以下命令提供从搜索结果提取新字段的不同方式。

命令	描述
erex	允许您指定示例或计数器示例值，以自动提取具有相似值的字段。
extract, kv	从搜索结果提取字段-值对。
kvform	使用表单模板从搜索结果中提取值。
rex	指定 Perl 正则表达式命名的组，以在搜索时提取字段。
spath	提供一种直接的方式从结构化数据格式（XML 和 JSON）中提取字段。
xmlkv	提取 XML 键值对。

修改字段和字段值

使用以下命令修改字段或字段值。

命令	描述
convert	将字段值转换为数字值。
filldown	使用最后的非空值替换空值。
fillnull	使用指定值替换空值。
makemv	在搜索期间将指定字段更改为多值字段。
nomv	在搜索时间将指定的多值字段更改为单值字段。
reltime	将 'now' 和 '_time' 之差转换为可读的值，并将该值添加到搜索结果的 'reltime' 字段中。
rename	给指定字段重新命名。使用通配符指定多个字段。
replace	将指定字段的值替换为指定的新值。

查找异常事件

以下命令用于查找您的数据中的异常事件。搜索不常见或无关事件和字段或将相似事件组成群集。

命令	描述
analyzefields, af	分析数值字段，以确定其预知另一个离散字段的能力。
anomalies	计算事件的 "unexpectedness" 分数。
anomalousvalue	查找并汇总不规则或不常见的搜索结果。
anomalydetection	先计算每个事件的概率然后再检测异常低的概率值，由此来识别异常事件。
cluster	将相似事件组成群集。
kmeans	对所选字段执行 K 均值群集化。
outlier	移除无关的数字值。
rare	显示最不常用的字段值。

地理和位置

以下命令向您的搜索结果中添加地理信息。

命令	描述
iplocation	返回位置信息，如城市、国家、纬度、经度等（基于 IP 地址）。
geom	为每个事件添加一个名为 "geom" 的字段。此字段包含 JSON 中多边形几何体的地理数据结构，用于地区分布图可视化。此命令需要一个安装了 external_type=geo 的外部查找。
geomfilter	接受两个点，这两个点可以指定剪切地区分布图的边框。落在边框以外的点会被过滤掉。
geostats	生成将在世界地图上呈现且群集化成地理数据箱的统计信息。

指标

这些命令与指标数据一起使用。

命令	描述
mcollect	将事件转换为指标数据点，并将数据点插入到搜索头上的指标索引。
meventcollect	将事件转换为指标数据点并将数据点插入索引器层的指标索引。
mpreview, msearch	在指标索引中提供指标时间序列中原始指标数据点的样本。帮助您排除指标数据中的问题。
mstats	为指标索引中的 measurement、metric_name 和 dimension 字段计算可供可视化使用的统计数据。

预测和趋势

以下命令预知未来值并计算可用于新建可视化的趋势线。

命令	描述
predict	允许您使用时间系列算法预测字段的未来值。
trendline	计算字段的移动平均值。
x11	允许您通过删除季节模式来确定数据中的趋势。

报表

以下命令可用于构建转换搜索。以下命令返回图表和其他类型数据可视化所需的统计数据表。

命令	描述
----	----

addtotals	计算每个结果的所有数值字段值的总和。
autoregress	根据您指定的字段，让这些事件为自动回归或移动平均值的计算做好准备。
bin, discretize	将连续的数字值放入离散集中。
chart	以表格输出形式返回结果以绘制图表。还可以查看统计和图表函数。
contingency, counttable, ctable	构建两个字段的应变表。
correlate	计算不同字段之间的相关性。
eventcount	返回索引中的事件数。
eventstats	向所有搜索结果添加摘要统计信息。
gauge	将结果转换为适合由仪表图类型显示的格式。
makecontinuous	生成具有 X 轴连续性的字段（由 chart/timechart 调用）
mstats	计算指标索引中的测量、metric_name 和维度字段的统计数据。
outlier	移除无关的数字值。
rare	显示最不常用的字段值。
stats	提供统计信息，可以选择按字段进行分组。还可以查看统计和图表函数。
streamstats	以流化方式向所有搜索结果中添加摘要统计信息。
timechart	新建时间系列图和相应的统计信息表。还可以查看统计和图表函数。
top	显示字段最常见的值。
trendline	计算字段的移动平均值。
tstats	在 tsidx 文件中执行索引字段上的统计查询。
untable	把结果从表格形式转换为 stats 输出相似的格式。与 xyseries 和 maketable 颠倒。
xyseries	将结果转换为适用于绘图的格式。

结果

以下命令可用于管理搜索结果。例如，您可将一组结果附加到另一组结果，从结果中筛选更多事件、重新设置结果格式等。

告警

使用此命令将搜索结果通过电子邮件发送。

命令	描述
sendemail	将搜索结果通过电子邮件（内联或以附件形式）发送给一个或更多指定的电子邮件地址。

附加

使用以下命令将一组结果附加到另一组结果或其自身。

命令	描述
append	将子搜索结果附加到当前结果。
appendcols	将子搜索结果的字段附加到当前结果，第一组子搜索结果对应第一个结果，第二组对应第二个，依此类推。
join	类似 SQL 的连接，将来自主结果管道的结果与来自子管道的结果连接在一起。
selfjoin	将结果与自身连接。

筛选

使用以下命令从您的当前结果中删除更多事件或字段。

命令	描述
dedup	删除与指定条件相匹配的后续结果。
fields	删除搜索结果中的字段。
from	从数据集（例如数据模型数据集、CSV 查找、KV 存储查找、保存的搜索或表数据集）检索数据。
mvcombine	将搜索结果中有单个不同字段值的事件合并到一个结果中，该结果具有包含不同字段的多值字段。
regex	删除与指定正则表达式不匹配的结果。
searchtx	根据指定的搜索约束查找交易事件。
table	使用指定字段新建表。
uniq	删除与之前的结果完全重复的任何搜索。
where	对数据执行任意筛选。还可以参阅“评估函数”。

格式

使用以下命令重新设置您当前结果的格式。

命令	描述
fieldformat	使用 eval 表达式可在呈现字段值时更改其格式，而无需更改其基础值。不适用于导出的数据。
transpose	将搜索结果的行重新格式化为列。有助于解决图表的 X 轴和 Y 轴显示问题，或将数据集转换为一系列数据以生成图表。
untable	把结果从表格形式转换为 stats 输出相似的格式。与 xyseries 和 maketable 颠倒。
xyseries	将结果转换为适用于绘图的格式。

生成

使用以下命令生成或返回事件。

命令	描述
gentimes	返回与时间范围匹配的结果。
loadjob	加载先前完成的搜索任务的事件或结果。
makeresults	新建指定数量的空搜索结果。
mvexpand	将多值字段的值扩展到多值字段每个值的不同事件中。
savedsearch	返回保存的搜索的搜索结果。
search	搜索匹配事件的索引。如果搜索管道并未从另一个生成命令开始，则在每个搜索管道的开始处，此命令是隐式的。

分组

使用以下命令对当前结果进行分组或分类。

命令	描述
cluster	将相似事件组成群集。
kmeans	对所选字段执行 K 均值群集化。
mvexpand	将多值字段的值扩展到多值字段每个值的不同事件中。
transaction	将搜索结果分组成交易。

typelearner	生成建议的 eventtype。
typer	计算搜索结果的 eventtype。

重新排序

使用以下命令更改当前搜索结果的顺序。

命令	描述
head	返回前 n 个指定结果。
reverse	颠倒结果的顺序。
sort	按照指定的字段对搜索结果进行排序。
tail	返回后 N 个指定结果。

读取

使用以下命令读取来自外部文件或先前搜索的结果。

命令	描述
inputcsv	从指定 CSV 文件加载搜索结果。
inputlookup	从指定的静态查找表加载搜索结果。
loadjob	加载先前完成的搜索任务的事件或结果。

写入

使用以下命令定义当前搜索结果的输出方式。

命令	描述
collect, stash	将搜索结果放入摘要索引。
meventcollect	将事件转换为指标数据点并将数据点插入索引器层的指标索引。
mcollect	将事件转换为指标数据点，并将数据点插入到搜索头上的指标索引。
outputcsv	将搜索结果输出到指定的 CSV 文件中。
outputlookup	将搜索结果保存到指定的静态查找表中。
outputtext	将结果的原始文本字段 (_raw) 输出到 _xml 字段中。
sendemail	将搜索结果通过电子邮件（内联或以附件形式）发送给一个或更多指定的电子邮件地址。

搜索

命令	描述
localop	在本地而非远程对等节点上运行后续命令（即此命令之后的所有命令）。
map	循环运算符，对每个搜索结果执行搜索。
redistribute	调用并行化简搜索处理缩短受支持的 SPL 命令集的搜索运行时间。
search	搜索匹配事件的索引。如果搜索管道并未从另一个生成命令开始，则在每个搜索管道的开始处，此命令是隐式的。
sendemail	将搜索结果通过电子邮件（内联或以附件形式）发送给一个或更多指定的电子邮件地址。

子搜索

以下是您可用子搜索使用的命令。

命令	描述
append	将子搜索结果附加到当前结果。
appendcols	将子搜索结果的字段附加到当前结果，第一组子搜索结果对应第一个结果，第二组对应第二个，依此类推。
appendpipe	将应用于当前结果集的子管道的结果附加到结果。
foreach	为通配符字段列表中的每个字段运行模板化流子搜索。
format	获取子搜索的结果，并将结果格式化为单个结果。
join	将子搜索的结果与主搜索的结果合并。
return	指定从子搜索返回的值。
set	对子搜索执行 set 操作 (union, diff, intersect)。

时间

使用以下命令基于时间范围搜索或向您的事件添加时间信息。

命令	描述
gentimes	返回与时间范围匹配的结果。
localize	返回找到的搜索结果所属时间范围的列表。
reltime	将 'now' 和 '_time' 之差转换为可读的值，并将该值添加到搜索结果的 'reltime' 字段中。

命令类型

所有搜索命令都可分为六大类：可分配流命令、集中流命令、转换命令、生成命令、安排命令和数据集处理命令。这些类型并不相互排斥。命令可能是流式命令或转换命令，也可能是生成命令。

下列表格列出了符合上述四种类型的命令。有关每种类型的详细解释，请参阅《搜索手册》中的“命令类型”。

流命令

流命令在搜索返回事件时作用于每个事件。

- 可分配流命令在索引器或搜索头上运行，具体视该命令在搜索内的调用位置而定。可分配流命令可并行应用于索引数据的子集。
- 集中流命令将转换应用到搜索返回的每个事件。与可分配流命令不同，集中流命令仅在搜索头中运行。

命令	注释
addinfo	可分配流
addtotals	可分配流。用于计算总列数（非总行数）时为转换命令。
anomalydetection	
append	
arules	
autoregress	集中流。
bin	若使用 span 参数指定则为流命令。
bucketdir	
cluster	某些模式中的流。
convert	可分配流。
dedup	默认流。使用 sortby 参数或指定 keepevents=true 可使 dedup 命令成为数据集处理命令。

Eval	可分配流。
extract	可分配流。
fieldformat	可分配流。
字段	可分配流。
fillnull	指定 field-list 时的可分配流。未指定 field-list 时，使用数据集处理命令。
head	集中流。
highlight	可分配流。
iconify	可分配流。
iplocation	可分配流。
join	集中流，如果有要加入的已定义字段集。未指定 field-list 时，使用数据集处理命令。
lookup	如果指定默认的 local=false，使用可分配流命令。当 local=true 时，使用安排命令。
makemv	可分配流。
multikv	可分配流。
mvexpand	可分配流。
nomv	可分配流。
rangemap	可分配流。
regex	可分配流。
reltime	可分配流。
rename	可分配流。
replace	可分配流。
rex	可分配流。
search	若进一步沿搜索管道向下使用则为可分配流。搜索中的首个命令为生成命令。
spath	可分配流。
strcat	可分配流。
streamstats	集中流。
tags	可分配流。
transaction	集中流。
typer	可分配流。
where	可分配流。
untable	可分配流。
xmlkv	可分配流。
xmlunescape	
xpath	可分配流。
xyseries	如果指定默认的 grouped=false 参数，使用可分配流命令。否则使用转换命令。

生成命令

生成命令在未转换事件的情况下，从一个或多个索引中生成事件或报表。

命令	注释
datamodel	报表生成

dbinspect	报表生成。
eventcount	报表生成。
from	根据命令引用的搜索或知识对象，可以是生成报表或者生成事件。
gentimes	事件生成。
inputcsv	事件生成（集中）。
inputlookup	当 append=false（默认）时为事件生成（集中）。
loadjob	事件生成（集中）。
makeresults	报表生成。
metadata	报表生成。尽管元数据从所有对等节点上获取数据，在元数据之后运行的命令仅在搜索头上运行。
metasearch	事件生成。
mstats	报表生成命令，当指定 append=true 时除外。
multisearch	事件生成。
pivot	报表生成。
rest	
search	搜索中的第一个命令（默认）为事件生成（可分配）命令。如果之后在搜索管道中使用，则为流（可分配）命令。
searchtxn	事件生成。
设置	事件生成。
tstats	prestats=true 时报告生成（可分配）。prestats=false 时，tstats 是事件生成。

转换命令

转换命令会对结果进行排序并生成数据表格。该命令将每个事件的指定单元格值“转换”成可用于统计目的的数字值。

在较早的 Splunk 软件版本中，转换命令称为“报表命令”。

命令	注释
addtotals	用于计算总列数（非总行数）时为转换命令。默认用于计算行总行数，则为可分配的流命令。
chart	
cofilter	
contingency	
history	
makecontinuous	
mvcombine	
rare	
stats	
table	
timechart	
top	
xyseries	如果 grouped=true，则为转换命令。如果是默认设置 grouped=false，则为流（可分配）命令。

安排命令

安排命令是控制如何处理搜索某些方面的命令。这些命令不会直接影响搜索集的最终结果。例如，您可以对搜索应用安排命

令，以启用或禁用可有助于快速完成整个搜索的搜索优化。

命令	注释
localop	
lookup	仅当 <code>local=true</code> 时变成安排命令。这样会强制在搜索头而非远程对等节点上运行 <code>lookup</code> 命令。如果是默认设置 <code>local=false</code> ，则为流（可分配）命令。
noop	
redistribute	
require	

数据集处理命令

数据集处理命令是在命令可以运行之前需要完整数据集的命令。某些命令在特定情况下或使用特定参数时符合其他命令类型。

命令	注释
anomalousvalue	某些模式
anomalydetection	某些模式
append	某些模式
bin	某些模式。如果指定 <code>span</code> 参数，则为流命令。
cluster	某些模式
concurrency	
datamodel	
dedup	使用 <code>sortby</code> 参数或指定 <code>keepevents=true</code> 可使 <code>dedup</code> 命令成为数据集处理命令。否则， <code>dedup</code> 是流命令。
eventstats	
fieldsummary	某些模式
fillnull	某些模式
from	某些模式
join	某些模式
map	
outlier	
pivot	某些模式
reverse	
sort	
tail	
transaction	某些模式
union	某些模式

面向 SQL 用户的 Splunk SPL

以下内容不是 SQL 与 Splunk 搜索处理语言 (SPL) 之间的精确映射，但是，如果您熟悉 SQL，这一快速对比可以帮助您快速熟悉搜索命令的使用。

概念

Splunk 平台不会将数据存储在常规数据库中。相反，它将数据存储在具有隐式时间维度的分布式、非关系型、半结构化的数据库中。关系数据库要求提前定义所有表列并且不会在只插入新硬件的情况下自动扩展。但是，Splunk 中有许多与数据库领域相

似的概念。

数据库概念	Splunk 概念	注释
SQL 查询	Splunk 搜索	Splunk 搜索是对索引数据进行检索，并且可以执行转换和报告操作。可以将搜索所获得的结果通过管道符从一个命令传递或传输到另一个命令，以对这些结果进行过滤、修改、重新排序和分组。
表/视图	搜索结果	可将搜索结果视为数据库视图，一个动态生成的具有行和列的表。
index	index	所有值和字段通过 Splunk 软件进行索引，因此不需要手动添加、更新、丢弃，甚至不需要考虑对列建立索引的问题。系统可以自动快速地检索一切内容。
row	结果/事件	Splunk 搜索中的结果是一个包含字段（即，列）值的列表，对应于表格的行。事件是指具有时间戳和原始文本的结果。通常，事件是日志文件中的一条记录，例如： 173.26.34.223 - - [01/Jul/2009:12:05:27 -0700] "GET /trade/app?action=logout HTTP/1.1" 200 2953
柱形图	字段	字段从搜索中动态地返回，这意味着一个搜索可能会返回一组字段，而另一个搜索可能会返回另一组字段。在告知 Splunk 软件如何从原始底层数据提取更多字段后，同一个搜索将返回比之前多的字段。字段不与数据类型关联。
数据库/方案	索引/应用	Splunk 索引是一个数据集，类似于包含一组表集合的数据库。对应数据的域知识、如何提取、运行哪些报表等等都存储在 Splunk 应用中。

从 SQL 到 Splunk SPL

SQL 设计用于搜索由列组成的关系数据库表。SPL 设计用于搜索由字段组成的事件。在 SQL 中，您可经常看到使用 "mytable" 和 "mycolumn" 的示例。在 SPL 中，您可看到引用“字段”的示例。在这些示例中，"source" 字段用作 "table" 的代理。在 Splunk 软件中，"source" 是文件、流或其他生成特定数据的输入的名称，例如 /var/log/messages 或 UDP:514。

当从任何语言翻译成另一种语言时，通常会由于原语言中的习语而导致译文很长。下面显示的一些 Splunk 搜索示例可能更加简明，但为了保持并行性和清晰性，SPL 表和字段使用了与 SQL 示例相同的名称。

- SPL 搜索不需要 FIELDS 命令筛选列，因为用户界面会提供更方便的筛选方式。FIELDS 命令用于 SPL 示例中以保持并行性。
- 使用 SPL 时，您不用在布尔搜索中使用 AND 运算符，因为两个术语之间隐含 AND 运算符。但是，当您使用 AND 或 OR 运算符时，必须以大写形式指定。
- SPL 命令不需要大写指定。在这些 SPL 示例中，以大写形式指定命令更易于识别和清晰。

SQL 命令	SQL 示例	Splunk SPL 示例
SELECT *	SELECT * FROM mytable	source=mytable
WHERE	SELECT * FROM mytable WHERE mycolumn=5	source=mytable mycolumn=5
SELECT	SELECT mycolumn1, mycolumn2 FROM mytable	source=mytable FIELDS mycolumn1, mycolumn2
AND/OR	SELECT * FROM mytable WHERE (mycolumn1="true" OR mycolumn2="red") AND mycolumn3="blue"	source=mytable AND (mycolumn1="true" OR mycolumn2="red") AND mycolumn3="blue" 注意：SPL 中隐含有 AND 运算符，不需要指定。对于此示例，您还可以使用： source=mytable (mycolumn1="true" OR mycolumn2="red") mycolumn3="blue"
AS (别名)	SELECT mycolumn AS column_alias	source=mytable

	FROM mytable	RENAME mycolumn as column_alias FIELDS column_alias
BETWEEN	SELECT * FROM mytable WHERE mycolumn BETWEEN 1 AND 5	source=mytable mycolumn>=1 mycolumn<=5
GROUP BY	SELECT mycolumn, avg(mycolumn) FROM mytable WHERE mycolumn=value GROUP BY mycolumn	source=mytable mycolumn=value STATS avg(mycolumn) BY mycolumn FIELDS mycolumn, avg(mycolumn) 不少命令使用 by-clause 进行信息分组，包括 chart、 rare、sort、stats 和 timechart。
HAVING	SELECT mycolumn, avg(mycolumn) FROM mytable WHERE mycolumn=value GROUP BY mycolumn HAVING avg(mycolumn)=value	source=mytable mycolumn=value STATS avg(mycolumn) BY mycolumn SEARCH avg(mycolumn)=value FIELDS mycolumn, avg(mycolumn)
LIKE	SELECT * FROM mytable WHERE mycolumn LIKE "%some text%"	source=mytable mycolumn="*some text*" 注意：Splunk SPL 中最常见的搜索在 SQL 中是几乎无法实现的 – 即在所有字段中搜索子字符串。下面 SPL 搜索将返回在任意位置包含 "some text" 的所有行： source=mytable "some text"
ORDER BY	SELECT * FROM mytable ORDER BY mycolumn desc	source=mytable SORT -mycolumn 在 SPL 中，您可在字段名称前面使用负号（-）进行降序排列。
SELECT DISTINCT	SELECT DISTINCT mycolumn1, mycolumn2 FROM mytable	source=mytable DEDUP mycolumn1 FIELDS mycolumn1, mycolumn2
SELECT TOP	SELECT TOP(5) mycolumn1, mycolumn2 FROM mytable1 WHERE mycolumn3 = "bar" ORDER BY mycolumn1 mycolumn2	Source=mytable1 mycolumn3="bar" FIELDS mycolumn1 mycolumn2 SORT mycolumn1 mycolumn2 HEAD 5
INNER JOIN	SELECT * FROM mytable1 INNER JOIN mytable2 ON mytable1.mycolumn= mytable2.mycolumn	index=myIndex1 OR index=myIndex2 stats values(*) AS * BY myField 注意：还可以通过其他两种方法来加入表格： <ul style="list-style-type: none">• 使用 lookup 命令添加来自外部表格的字段： ... LOOKUP myvaluelookup mycolumn OUTPUT myoutputcolumn <ul style="list-style-type: none">• 使用子搜索： source=mytable1 [SEARCH source=mytable2 mycolumn2=myvalue FIELDS mycolumn2] 如果您想要加入的列有不同的名称，请使用 rename 命令来重命名其中一列。例如：要重命名 mytable2 中的列： source=mytable1 JOIN type=inner mycolumn [SEARCH source=mytable2

		<pre> RENAME mycolumn2 AS mycolumn]</pre> <p>要重命名 myindex1 中的列:</p> <pre>index=myIndex1 OR index=myIndex2 rename myfield1 as myField stats values(*) AS * BY myField</pre> <p>无论您是使用搜索命令、查找或是子搜索，您都可以重命名列。</p>
LEFT (OUTER) JOIN	<pre>SELECT * FROM mytable1 LEFT JOIN mytable2 ON mytable1.mycolumn= mytable2.mycolumn</pre>	<pre>source=mytable1 JOIN type=left mycolumn [SEARCH source=mytable2]</pre>
SELECT INTO	<pre>SELECT * INTO new_mytable IN mydb2 FROM old_mytable</pre>	<pre>source=old_mytable EVAL source=new_mytable COLLECT index=mydb2</pre> <p>注意: COLLECT 通常用于将计算开销极大的字段存储回 Splunk 部署中，以提高将来的访问速度。目前列举的示例是一个非典型示例，仅供与 SQL 命令对比之用。source 将被重命名为 orig_source</p>
TRUNCATE TABLE	TRUNCATE TABLE mytable	<pre>source=mytable DELETE</pre>
INSERT INTO	<pre>INSERT INTO mytable VALUES (value1, value2, value3,....)</pre>	<p>注意: 请参阅 SELECT INTO。如果需要，各个记录的添加可以不使用搜索语言，而是使用 API。</p>
UNION	<pre>SELECT mycolumn FROM mytable1 UNION SELECT mycolumn FROM mytable2</pre>	<pre>source=mytable1 APPEND [SEARCH source=mytable2] DEDUP mycolumn</pre>
UNION ALL	<pre>SELECT * FROM mytable1 UNION ALL SELECT * FROM mytable2</pre>	<pre>source=mytable1 APPEND [SEARCH source=mytable2]</pre>
DELETE	<pre>DELETE FROM mytable WHERE mycolumn=5</pre>	<pre>source=mytable1 mycolumn=5 DELETE</pre>
UPDATE	<pre>UPDATE mytable SET column1=value, column2=value,... WHERE some_column=some_value</pre>	<p>注意: 在 Splunk® Enterprise 中更新记录时需要考虑几点事项。首先，您可以只将新值添加到 Splunk 部署中（请参阅 INSERT INTO），而不必担心删除旧值，因为 Splunk 软件一律会先返回最新的结果。其次，在检索时，您可以始终去除重复结果，以确保只使用最新值（请参阅 SELECT DISTINCT）。最后，您实际上可以删除旧记录（请参阅 DELETE）。</p>

另请参阅

- 了解 SPL 语法

SPL 数据类型和子句

数据类型

bool

<bool> 参数值代表布尔数据类型。文档将指定 'true' 或 'false'。命令中亦可使用布尔值的其他变量。例如，'true' 也可以改用 't'、'T'、'TRUE'，或数字 '1'。'false' 则可以使用 'f'、'F'、'FALSE'，或数字 '0'。

整型

<int> 参数值代表整数数据类型。

num

<num> 参数值代表数字数据类型。

float

<float> 参数值代表 float 数据类型。

通用语法子句

bin-span

语法: span=<span-length> | <log-span>

描述: 设置每个 bin 的大小。

示例: span=2d

示例: span=5m

示例: span=10

by-clause

语法: by <field-list>

描述: 分组所依据的字段。

示例: BY addr, port

示例: BY host

eval-function

语法: abs | case | cidrmatch | coalesce | exact | exp | floor | if | ifnull | isbool | isint | isnotnull | isnull | isnum | isstr | len|like | ln|log | lower | match | max | md5 | min | mvcount | mvindex | mvfilter | now | null | nullif | pi | pow | random | replace | round | searchmatch | sqrt | substr | tostring | trim | ltrim | rtrim | typeof | upper | urldecode | validate

描述: eval 使用的函数。

示例: md5(field)

示例: typeof(12) + typeof("string") + typeof(1==2) + typeof(badfield)

示例: searchmatch("foo AND bar")

示例: sqrt(9)

示例: round(3.5)

示例: replace(date, "^(\\d{1,2})/(\\d{1,2})/", "\\\2\\\\1/")

示例: pi()

示例: nullif(fielda, fieldb)

示例: random()

示例: pow(x, y)

示例: mvfilter(match(email, ".net\$") OR match(email, ".org\$"))

示例: mvindex(multifield, 2)

示例: null()

示例: now()

示例: isbool(field)

示例: exp(3)

示例: floor(1.9)

示例: coalesce(null(), "Returned value", null())

示例: exact(3.14 * num)

示例: case(error == 404, "Not found", error == 500, "Internal Server Error", error == 200, "OK")

示例: cidrmatch("123.132.32.0/25", ip)

示例: abs(number)

示例: isnotnull(field)

示例: substr("string", 1, 3) + substr("string", -3)

示例: if(error == 200, "OK", "Error")

示例: len(field)

示例: log(number, 2)

示例: lower(username)

示例: match(field, "^\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}\\$")

示例: max(1, 3, 6, 7, "f"^\d{1,3}\.\.\d{1,3}\.\.\d{1,3}\.\.\d{1,3}\$\$)oo", field)
示例: like(field, "foo%")
示例: ln(bytes)
示例: mvcount(multifield)
示例: urldecode("http%3A%2F%2Fwww.splunk.com%2Fdownload%3Fr%3Dheader")
示例: validate(isint(port), "ERROR: Port is not an integer", port >= 1 AND port <= 65535, "ERROR: Port is out of range")
示例: tostring(1==1) + " " + tostring(15, "hex") + " " + tostring(12345.6789, "commas")
示例: trim(" ZZZabcZZ ", " Z")

eval-ed-field

语法: eval(<eval-expression>)
描述: 进行动态 eval 处理的字段

field

field-list

regex-expression

语法: ("")?<string>("")?
描述: PCRE 库所支持的 Perl 兼容的正则表达式。
示例: ... | regex _raw="(?<!\\d)10.\d{1,3}\.\.\d{1,3}\.\.\d{1,3}(?!\\d)"

single-agg

语法: count | stats-func (<field>)
描述: 应用到一个单一字段（可以是经 Eval 处理的字段）的单一聚合。不允许使用通配符。必须指定此字段，除非对事件整体应用特殊的 'count' 聚合器。
示例: avg(delay)
示例: sum({date_hour * date_minute})
示例: count

sort-by-clause

语法: ("-" | "+")<sort-field> ","
描述: 列出排序所依据的字段及其排序顺序（升序或降序）
示例: - time, host
示例: -size, +source
示例: _time, -host

span-length

语法: <int:span>(<timescale>)?
描述: 每个数据箱的跨度。如果使用 timescale，则将其用作时间范围。否则，这是一个绝对的数据桶“长度”。
示例: 2d
示例: 5m
示例: 10

split-by-clause

语法: <field> (<tc-option>)* (<where-clause>)?
描述: 指定拆分所依据的字段。如果字段为数字值，将应用默认离散化。

stats-agg

语法: <stats-func> ("(" (<eval-ed-field> | <wc-field>)?) ")")?
描述: 由应用到某一个字段或一组字段的聚合函数形成的指示符。自 4.0 起，它还可以是应用到任意 eval 表达式的聚合函数。eval 表达式必须包括在 "[" 和 "]" 中。如果没有指定括号内的字段，聚合将被独立应用到所有字段，这等同于调用 * 的字段值。将数字聚合器应用到不全是数字的字段时，不会为该聚合生成任何列。
示例: count({sourcetype="splunkd"})
示例: max(size)
示例: stdev(*delay)
示例: avg(kbps)

stats-agg-term

语法: <stats-agg> (as <wc-field>) ?
描述: 统计说明符, 可选择重命名为新字段名称。
示例: count(device) AS numdevices
示例: avg(kbps)

subsearch

语法: [<string>]
描述: 指定一个子搜索。
示例: [search 404 | select url]

tc-option

语法: <bins-options> | (usenull=<bool>) | (useother=<bool>) | (nullstr=<string>) | (otherstr=<string>)
描述: 用于控制拆分方式字段行为的选项。除 bin 选项外: usenull 用于控制是否为不包含拆分字段的事件新建一个系列。此系列由 nullstr 选项值进行标记, 默认为 NULL。useother 指定是否应该为图形中不包括的数据系列 (因为它们不满足 <where-clause> 的条件) 添加一个系列。此系列由 otherstr 选项值进行标记, 默认为 OTHER。
示例: otherstr=OTHERFIELDS
示例: usenull=f
示例: bins=10

timeformat

语法: timeformat=<string>
描述: 设置用于 starttime 和 endtime 条件的时间格式。
示例: timeformat=%m/%d/%Y:%H:%M:%S

timestamp

语法: (MM/DD/YY)?: (HH:MM:SS)? | <int>
描述: 无
示例: 2007 年 10 月 1 日: 12:34:56
示例: -5

where-clause

语法: where <single-agg> <where-comp>
描述: 指定在 tc-by 子句中给出某一字段时包括特定数据系列的条件。这是可选子句, 如果被忽略, 则默认为 "where sum in top10"。将对每个数据系列应用聚合术语, 并将这些聚合的结果与条件进行比较。此选项最常见的用法是, 在系列选择中选择峰值而不是分布总和。默认值将根据曲线下的区域查找前十个系列。或者, 可将总和替换为最大值, 以查找具有十个最高峰值的系列。
示例: where max < 10
示例: where count notin bottom10
示例: where avg > 100
示例: where sum in top5

wc-field

评估函数

评估函数

使用评估函数根据您的事件评估表达式并返回结果。

快速参考

请参阅“支持的函数和语法”部分获取评估函数快速参考列表。

命令

您可以将评估函数和 eval、fieldformat 及 where 命令结合使用，并作为 eval 表达式的一部分和其他命令结合使用。

用法

- 所有能接受字符串的函数都能接受文本字符串或任何字段。
- 所有能接受数字的函数都能接受文本数字或任何数字字段。

字符串参数和字段

对大多数评估函数而言，若应指定字符串参数，您可以指定文字字符串或字段名称。文字字符串必须用双引号引起来。换句话说，如果函数语法指定了字符串，您可以指定可产生字符串的任何表达式。例如，您有包含服务器名称的名为 name 的字段。您想要在名称末尾添加文字字符串 server。您可以指定：name + "server"。

嵌套函数

您可以指定一个函数为另一个函数的参数。

在以下示例中，cidrmatch 函数用作 if 函数的第一个参数。

```
... | eval isLocal=if(cidrmatch("123.132.32.0/25",ip), "local", "not local")
```

以下示例显示如何使用 true() 函数为 case 函数提供默认值。

```
... | eval error=case(status == 200, "OK", status == 404, "Not found", true(), "Other")
```

支持的函数和语法

您可通过两种方式查看支持的评估函数的信息：

- 按类别排列的函数列表
- 按字母顺序排列的函数列表

按类别排列的函数列表

下表是支持的评估函数的快速参考信息，按类别排列。此表格提供每个函数的简要描述。使用表格中的链接以了解有关每个函数的更多信息，并查看示例。

函数类型	支持的函数和语法	描述
	case(X, "Y", ...)	接受其他条款和值。返回条件评估为 TRUE 的第一个值。
	cidrmatch("X", Y)	根据 IP 地址是否匹配 CIDR 表示法返回 TRUE 或 FALSE。
	coalesce(X, ...)	此函数会获取任意数量的参数并返回第一个不是空值的值。
	false()	返回 FALSE。
	if(X, Y, Z)	如果条件 X 评估为 TRUE，返回 Y，否则返回 Z。
	in(FIELD, VALUE-LIST)	如果列表中的一个值和您指定字段中的值匹配，则函数返回 TRUE。

对比和条件函数	<code>like(TEXT, PATTERN)</code>	如果 TEXT 匹配 PATTERN，返回 TRUE。
	<code>lookup(<lookup_table>, <json_object>, <json_array>)</code>	此函数执行 CSV 查找。它以 JSON 对象的形式返回一个或多个输出字段。
	<code>match(SUBJECT, "REGEX")</code>	根据 REGEX 是否匹配 SUBJECT 返回 TRUE 或 FALSE
	<code>null()</code>	此函数不获取任何参数并返回 NULL。
	<code>nullif(X, Y)</code>	此函数用于比较字段。该函数获取两个参数，X 和 Y，若 X = Y 将返回空值，否则将返回 X。
	<code>searchmatch(X)</code>	如果搜索字符串 (X) 匹配事件，使用此函数返回 TRUE。
	<code>true()</code>	返回 TRUE。
	<code>validate(X, Y, ...)</code>	使用此函数返回与评估为 FALSE 的第一个表达式 X 对应的字符串 Y。此函数与 case 函数对立。
转换函数	<code>printf("format", arguments)</code>	根据您提供的格式描述新建格式化字符串。
	<code>tonumber(NUMSTR, BASE)</code>	将字符串转换成数字。
	<code>tostring(X, Y)</code>	将输入（如数字或布尔值）转换成字符串。
加密函数	<code>md5(X)</code>	计算值 X 的 md5 哈希。
	<code>sha1(X)</code>	计算值 X 的 sha1 哈希。
	<code>sha256(X)</code>	计算值 X 的 sha256 哈希。
	<code>sha512(X)</code>	计算值 X 的 sha512 哈希。
日期和时间函数	<code>now()</code>	返回搜索开始的时间。
	<code>relative_time(X, Y)</code>	通过相对时间说明符调整时间。
	<code>strftime(X, Y)</code>	取 UNIX 时间，并用用户可读格式呈现。
	<code>strptime(X, Y)</code>	取用户可读时间，并用 UNIX 时间呈现。
	<code>time()</code>	eval 函数已计算的时间。每个事件的时间都不同，具体取决于处理事件的时间。
信息函数	<code>isbool(X)</code>	如果字段值为布尔值，返回 TRUE。
	<code>isint(X)</code>	如果字段值为整数，返回 TRUE。
	<code>isnotnull(X)</code>	如果字段值不为空值，返回 TRUE。
	<code>isnull(X)</code>	如果字段值为空值，返回 TRUE。
	<code>isnum(X)</code>	如果字段值是数字，返回 TRUE。
	<code>isstr(X)</code>	如果字段值是字符串，返回 TRUE。
	<code>typeof(X)</code>	返回表示字段类型（如数字、字符串、布尔值等）的字符串
JSON 函数	<code>json_object(<members>)</code>	根据键值对的成员创建一个新的 JSON 对象。
	<code>json_append(<json>, <path_value_pairs>)</code>	将值附加到 JSON 文档中指定数组的末尾处。
	<code>json_array(<values>)</code>	使用值列表创建 JSON 数组。
	<code>json_array_to_mv(<json_array>, <Boolean>)</code>	将适当的 JSON 数组的元素映射到多值字段。
	<code>json_extend(<json>, <path_value_pairs>)</code>	将数组展平为其组件值，并将这些值附加到有效 JSON 文档中指定数组的末尾处。
	<code>json_get(<path>)</code>	此函数从 JSON 片段返回一个值，并且返回零个或多个路径。该值以 JSON

	<code>json_extract(<json>, <path>)</code>	数组或 Splunk 软件本机类型值的形式返回。
	<code>json_keys(<json>)</code>	返回 JSON 对象中键值对中的键。键将以 JSON 数组的形式返回。
	<code>json_set(<json>, <path_value_pairs>)</code>	使用提供的值插入或覆盖 JSON 节点的值，然后返回更新的 JSON 对象。
	<code>json_valid(<json>)</code>	评估 JSON 片段是否使用有效的 JSON 语法并返回 TRUE 或 FALSE。
	<code>mv_to_json_array(<field>, <Boolean>)</code>	将多值字段的元素映射到 JSON 数组。
数学函数	<code>abs(X)</code>	返回绝对值。
	<code>ceiling(X)</code>	将值舍入到下一个最大整数。
	<code>exact(X)</code>	在格式化输出中以更高精度呈现数字 eval 计算的结果。
	<code>exp(X)</code>	返回指数函数 e^X 。
	<code>floor(X)</code>	将值舍入到下一个最小整数。
	<code>ln(X)</code>	返回自然对数。
	<code>log(X, Y)</code>	将 Y 用作基数返回 X 的对数。如果 Y 省略，使用基数 10。
	<code>pi()</code>	返回精确至 11 位的常数 pi。
	<code>pow(X, Y)</code>	返回 X 的 Y 次幂， X^Y 。
	<code>round(X, Y)</code>	返回 X 时，舍入到由 Y 指定的小数位数。默认是取整。
	<code>sigfig(X)</code>	将 X 舍入到重要数据的合适数字。
	<code>sqrt(X)</code>	返回值的平方根。
多值 eval 函数	<code>commands(X)</code>	返回包含在 X 中使用的命令列表的多值字段。
	<code>json_array_to_mv(<json_array>, <Boolean>)</code>	将适当的 JSON 数组的元素映射到多值字段。
	<code>mvappend(X, ...)</code>	根据所有指定值返回多值结果。
	<code>mvcount(MVFIELD)</code>	返回指定字段中值的数量。
	<code>mvdedup(X)</code>	从多值字段中移除所有重复值。
	<code>mvfilter(X)</code>	根据任意布尔表达式 X 筛选多值字段。
	<code>mvfind(MVFIELD, "REGEX")</code>	在多值字段中找到匹配 REGEX 的值的索引。
	<code>mvindex(MVFIELD, STARTINDEX, ENDINDEX)</code>	返回 STARTINDEX 和 ENDINDEX 描述的多值字段中的一组值。
	<code>mvjoin(MVFIELD, STR)</code>	取多值字段中的所有值，用 STR 分隔将它们加在一起。
	<code>mvmap(X, Y)</code>	这个函数迭代一个多值字段 (X) 的值，对每个值执行一个操作 (Y)，然后返回一个包含结果列表的多值字段。
	<code>mvrange(X, Y, Z)</code>	新建范围在 X 到 Y 之间，以 Z 递增的多值字段。
	<code>mvsort(X)</code>	返回按字典顺序排序的多值字段值。
	<code>mv_to_json_array(<field>, <Boolean>)</code>	将多值字段的元素映射到 JSON 数组。
	<code>mvzip(X, Y, "Z")</code>	获取两个多值字段 X 和 Y，然后通过依次将字段 X 的第一个值与字段 Y 的第一个值接合，将 X 的第二个值与 Y 的第二个值接合，依此类推，从而合并这两个字段。
统计 eval 函数	<code>split(X, "Y")</code>	返回通过分隔字符 Y 拆分 X 的 mv 字段。
	<code>max(X, ...)</code>	返回字符串或数字值的最大值。
	<code>min(X, ...)</code>	返回字符串或数字值的最小值。
	<code>random()</code>	返回范围为零到 $2^{31}-1$ 的伪随机整数。

文本函数	<code>len(X)</code>	返回字符串中的字符（不是字节）数量。
	<code>lower(X)</code>	将字符串转换为小写。
	<code>ltrim(X, Y)</code>	从字符串的左侧裁剪掉 Y 中的字符。
	<code>replace(X, Y, Z)</code>	返回字符串形式为：字符串 X 中每次出现的正则表达式字符串 Y 都用字符串 Z 替换。
	<code>rtrim(X, Y)</code>	返回从右边裁剪掉 Y 中字符的 X。
	<code>spath(X, Y)</code>	根据 Y 中位置路径在 X 中从结构化数据类型（XML 或 JSON）提取值。
	<code>substr(X, Y, Z)</code>	根据开始位置 Y 和长度 Z 返回 X 的子字符串。
	<code>trim(X, Y)</code>	从字符串的两边裁剪掉 Y 中的字符。
	<code>upper(X)</code>	返回大写的字符串。
	<code>urldecode(X)</code>	用原始字符替换缺少 URL 的字符。
三角函数和双曲函数	<code>acos(X)</code>	计算 X 的反余弦。
	<code>acosh(X)</code>	计算 X 的反双曲余弦。
	<code>asin(X)</code>	计算 X 的反正弦。
	<code>asinh(X)</code>	计算 X 的反双曲正弦。
	<code>atan(X)</code>	计算 X 的反正切。
	<code>atan2(X, Y)</code>	计算 X、Y 的反正切。
	<code>atanh(X)</code>	计算 X 的反双曲正切。
	<code>cos(X)</code>	计算 X 弧度角的余弦。
	<code>cosh(X)</code>	计算 X 弧度的反双曲余弦。
	<code>hypot(X, Y)</code>	计算三角形的斜边。
	<code>sin(X)</code>	计算 X 的正弦。
	<code>sinh(X)</code>	计算 X 的双曲正弦。
	<code>tan(X)</code>	计算 X 的正切。
	<code>tanh(X)</code>	计算 X 的双曲正切。

按字母顺序排列的函数列表

下表是支持的评估函数的快速参考信息，按字母顺序排列。此表格提供每个函数的简要描述。使用表格中的链接以了解有关每个函数的更多信息，并查看示例。

支持的函数和语法	描述	函数类型
<code>abs(X)</code>	返回绝对值。	数学函数
<code>acos(X)</code>	计算 X 的反余弦。	三角函数和双曲函数
<code>acosh(X)</code>	计算 X 的反双曲余弦。	三角函数和双曲函数
<code>asin(X)</code>	计算 X 的反正弦。	三角函数和双曲函数
<code>asinh(X)</code>	计算 X 的反双曲正弦。	三角函数和双曲函数
<code>atan(X)</code>	计算 X 的反正切。	三角函数和双曲函数
<code>atan2(X, Y)</code>	计算 X、Y 的反正切。	三角函数和双曲函数

<code>atanh(X)</code>	计算 X 的反双曲正切。	三角函数和双曲函数
<code>case(X, "Y", ...)</code>	接受其他条款和值。返回条件评估为 TRUE 的第一个值。	对比和条件函数
<code>cidrmatch("X", Y)</code>	根据 IP 地址是否匹配 CIDR 表示法返回 TRUE 或 FALSE。	对比和条件函数
<code>ceiling(X)</code>	将值舍入到下一个最大整数。	数学函数
<code>coalesce(X, ...)</code>	此函数会获取任意数量的参数并返回第一个不是空值的值。	对比和条件函数
<code>commands(X)</code>	返回包含在 X 中使用的命令列表的多值字段。	多值 eval 函数
<code>cos(X)</code>	计算 X 弧度角的余弦。	三角函数和双曲函数
<code>cosh(X)</code>	计算 X 弧度的反双曲余弦。	三角函数和双曲函数
<code>exact(X)</code>	在格式化输出中以更高精度呈现数字 eval 计算的结果。	数学函数
<code>exp(X)</code>	返回指数函数 e^X 。	数学函数
<code>false()</code>	返回 FALSE。	对比和条件函数
<code>floor(X)</code>	将值舍入到下一个最小整数。	数学函数
<code>hypot(X, Y)</code>	计算三角形的斜边。	三角函数和双曲函数
<code>if(X, Y, Z)</code>	如果条件 X 评估为 TRUE, 返回 Y, 否则返回 Z。	对比和条件函数
<code>in(FIELD, VALUE-LIST)</code>	如果列表中的一个值和您指定字段中的值匹配，则函数返回 TRUE。	对比和条件函数
<code>isbool(X)</code>	如果字段值为布尔值, 返回 TRUE。	信息函数
<code>isint(X)</code>	如果字段值为整数, 返回 TRUE。	信息函数
<code>isnotnull(X)</code>	如果字段值不为空值, 返回 TRUE。	信息函数
<code>isnull(X)</code>	如果字段值为空值, 返回 TRUE。	信息函数
<code>isnum(X)</code>	如果字段值是数字, 返回 TRUE。	信息函数
<code>isstr(X)</code>	如果字段值是字符串, 返回 TRUE。	信息函数
<code>json_append(<json>, <path_value_pairs>)</code>	将值附加到 JSON 文档中指定数组的末尾处。	JSON 函数
<code>json_array(<values>)</code>	使用值列表创建 JSON 数组。	JSON 函数
<code>json_array_to_mv(<json_array>, <Boolean>)</code>	将适当的 JSON 数组的元素映射到多值字段。	JSON 函数
<code>json_extend(<json>, <path_value_pairs>)</code>	将数组展平为其组件值，并将这些值附加到有效 JSON 文档中指定数组的末尾处。	JSON 函数
<code>json_extract(<json>, <paths>)</code>	从 JSON 片段和零个或多个路径返回一个值。该值以 JSON 数组或 Splunk 软件本机类型值的形式返回。	JSON 函数
<code>json_keys(<json>)</code>	返回 JSON 对象中键值对中的键。键将以 JSON 数组的形式返回。	JSON 函数
<code>json_object(<members>)</code>	根据键值对的成员创建一个新的 JSON 对象。	JSON 函数
<code>json_set(<json>, <path_value_pairs>)</code>	使用提供的值插入或覆盖 JSON 节点的值，然后返回更新的 JSON 对象。	JSON 函数
<code>json_valid(<json>)</code>	评估 JSON 片段是否使用有效的 JSON 语法并返回 TRUE 或 FALSE。	JSON 函数
<code>len(X)</code>	返回字符串中的字符（不是字节）数量。	文本函数
<code>like(TEXT, PATTERN)</code>	如果 TEXT 匹配 PATTERN, 返回 TRUE。	对比和条件函数
<code>log(X, Y)</code>	将 Y 用作基数返回 X 的对数。如果 Y 省略, 使用基数 10。	数学函数

<code>lookup(<lookup_table>, <json_object>, <json_array>)</code>	此函数执行 CSV 查找。它以 JSON 对象的形式返回一个或多个输出字段。	对比和条件函数
<code>ln(X)</code>	返回自然对数。	数学函数
<code>lower(X)</code>	将字符串转换为小写。	文本函数
<code>ltrim(X, Y)</code>	从字符串的左侧裁剪掉 Y 中的字符。	文本函数
<code>match(SUBJECT, "REGEX")</code>	根据 REGEX 是否匹配 SUBJECT 返回 TRUE 或 FALSE。	对比和条件函数
<code>max(X, ...)</code>	返回字符串或数字值的最大值。	统计 eval 函数
<code>md5(X)</code>	计算值 X 的 md5 哈希。	加密函数
<code>min(X, ...)</code>	返回字符串或数字值的最小值。	统计 eval 函数
<code>mvappend(X, ...)</code>	根据所有指定值返回多值结果。	多值 eval 函数
<code>mvcount(MVFIELD)</code>	返回指定字段中值的数量。	多值 eval 函数
<code>mvdedup(X)</code>	从多值字段中移除所有重复值。	多值 eval 函数
<code>mvfilter(X)</code>	根据任意布尔表达式 X 筛选多值字段。	多值 eval 函数
<code>mvfind(MVFIELD, "REGEX")</code>	在多值字段中找到匹配 REGEX 的值的索引。	多值 eval 函数
<code>mvindex(MVFIELD, STARTINDEX, ENDINDEX)</code>	返回 STARTINDEX 和 ENDINDEX 描述的多值字段中的一组值。	多值 eval 函数
<code>mvjoin(MVFIELD, STR)</code>	取多值字段中的所有值，用 STR 分隔将它们加在一起。	多值 eval 函数
<code>mvmap(X, Y)</code>	这个函数迭代一个多值字段 (X) 的值，对每个值执行一个操作 (Y)，然后返回一个包含结果列表的多值字段。	多值 eval 函数
<code>mvrange(X, Y, Z)</code>	新建范围在 X 到 Y 之间，以 Z 递增的多值字段。	多值 eval 函数
<code>mvsort(X)</code>	返回按字典顺序排序的多值字段值。	多值 eval 函数
<code>mv_to_json_array(<field>, <Boolean>)</code>	将多值字段的元素映射到 JSON 数组。	JSON 函数
<code>mvzip(X, Y, "Z")</code>	获取两个多值字段 X 和 Y，然后通过依次将字段 X 的第一个值与字段 Y 的第一个值接合，将 X 的第二个值与 Y 的第二个值接合，依此类推，从而合并这两个字段。	多值 eval 函数
<code>now()</code>	返回搜索开始的时间。	日期和时间函数
<code>null()</code>	此函数不获取任何参数并返回 NULL。	对比和条件函数
<code>nullif(X, Y)</code>	此函数用于比较字段。该函数获取两个参数，X 和 Y，若 X = Y 将返回空值，否则将返回 X。	对比和条件函数
<code>pi()</code>	返回精确至 11 位的常数 pi。	数学函数
<code>pow(X, Y)</code>	返回 X 的 Y 次幂， X^Y 。	数学函数
<code>printf("format", arguments)</code>	根据您提供的格式描述新建格式化字符串。	转换函数
<code>random()</code>	返回范围为零到 $2^{31}-1$ 的伪随机整数。	统计 eval 函数
<code>relative_time(X, Y)</code>	通过相对时间说明符调整时间。	日期和时间函数
<code>replace(X, Y, Z)</code>	返回字符串形式为：字符串 X 中每次出现的正则表达式字符串 Y 都用字符串 Z 代替。	文本函数
<code>round(X, Y)</code>	返回 X 时，舍入到由 Y 指定的小数位数。默认是取整。	数学函数
<code>rtrim(X, Y)</code>	返回从右边裁剪掉 Y 中字符的 X。	文本函数
<code>searchmatch(X)</code>	如果搜索字符串 (X) 匹配事件，使用此函数返回 TRUE。	对比和条件函数
<code>sha1(X)</code>	计算值 X 的 sha1 哈希。	加密函数

<code>sha256(X)</code>	计算值 X 的 sha256 哈希。	加密函数
<code>sha512(X)</code>	计算值 X 的 sha512 哈希。	加密函数
<code>sigfig(X)</code>	将 X 舍入到重要数据的合适数字。	数学函数
<code>sin(X)</code>	计算 X 的正弦。	三角函数和双曲函数
<code>sinh(X)</code>	计算 X 的双曲正弦。	三角函数和双曲函数
<code>spath(X, Y)</code>	根据 Y 中位置路径在 X 中从结构化数据类型（XML 或 JSON）提取值。	文本函数
<code>split(X, "Y")</code>	返回通过分隔字符 Y 拆分 X 的 mv 字段。	多值 eval 函数
<code>sqr(X)</code>	返回值的平方根。	数学函数
<code>strftime(X, Y)</code>	取 UNIX 时间，并用用户可读格式呈现。	日期和时间函数
<code>strptime(X, Y)</code>	取用户可读时间，并用 UNIX 时间呈现。	日期和时间函数
<code>substr(X, Y, Z)</code>	根据开始位置 Y 和长度 Z 返回 X 的子字符串。	文本函数
<code>tan(X)</code>	计算 X 的正切。	三角函数和双曲函数
<code>tanh(X)</code>	计算 X 的双曲正切。	三角函数和双曲函数
<code>time()</code>	eval 函数已计算的时间。每个事件的时间都不同，具体取决于处理事件的时间。	日期和时间函数
<code>tonumber(NUMSTR, BASE)</code>	将字符串转换成数字。	转换函数
<code>tostring(X, Y)</code>	将输入（如数字或布尔值）转换成字符串。	转换函数
<code>trim(X, Y)</code>	从字符串的两边裁剪掉 Y 中的字符。	文本函数
<code>true()</code>	返回 TRUE。	对比和条件函数
<code>typeof(X)</code>	返回表示字段类型（如数字、字符串、布尔值等）的字符串。	信息函数
<code>upper(X)</code>	返回大写的字符串。	文本函数
<code>urlencode(X)</code>	用原始字符替换缺少 URL 的字符。	文本函数
<code>validate(X, Y, ...)</code>	使用此函数返回与评估为 FALSE 的第一个表达式 X 对应的字符串 Y。此函数与 case 函数对立。	对比和条件函数

另请参阅

主题：
统计和图表函数

命令：
`Eval`
`fieldformat`
`where`

对比和条件函数

以下列表包含了您可以用于比较值或指定条件语句的函数。

有关在函数和嵌套函数中使用字符串和数字字段的信息，请参阅“评估函数”。

有关布尔运算符（如 AND 和 OR）的信息，请参阅“布尔运算符”。

`case(X, "Y", ...)`

描述

接受其他条款和值。返回条件评估为 TRUE 的第一个值。

此函数获取 X 和 Y 参数对。X 参数是从第一个到最后一个依次评估的布尔表达式。当遇到的第一个评估结果为 TRUE 的 X 表达式时，将返回相应的 Y 参数。如果评估结果均不为 true，则此函数默认为 NULL。

用法

您可以将此函数和 eval、fieldformat 及 where 命令结合使用，并作为 eval 表达式的一部分。

基本示例

此示例使用搜索教程中的示例数据，但应该可以与任意格式的 Apache Web 访问日志结合使用。要在您自己的 Splunk 实例中尝试此示例，您必须下载示例数据并遵循操作以将教程数据导入 Splunk。当运行搜索时，请使用时间范围昨天。

以下示例将返回相应 http 状态代码的描述。

```
sourcetype=access_* | eval description=case(status == 200, "OK", status == 404, "Not found", status == 500, "Internal Server Error") | table status description
```

统计选项卡中显示的结果如下所示：

status	描述
200	正常
200	正常
408	
200	正常
404	未找到
200	正常
406	
500	内部服务器错误
200	正常

关于状态不匹配指定的任意值时如何显示默认值的示例，请参阅 True 函数。

延伸示例

此示例显示如何以两种不同的方式使用 case 函数，以新建类别并新建自定义排序顺序。

此示例使用从 USGS 地震网站下载的近期地震数据。该数据是一个逗号分隔的 ASCII 文本文件，其中包含记录的每次地震的震级 (mag)、坐标 (经度、纬度)、区域 (地点) 等。

您可以从 [USGS 地震源](#) 下载当前 CSV 文件，然后将文件上传至您的 Splunk 实例（如果您想要遵循此示例的做法）。

您想要根据深度对地震进行分类。浅层地震发生深度小于 70 km。中层地震发生深度介于 70 到 300 km 之间。深层地震发生深度大于 300 km。我们将使用 Low、Mid 和 Deep 作为类别名称。

```
source=all_month.csv | eval Description=case(depth<=70, "Low", depth>70 AND depth<=300, "Mid", depth>300, "Deep") | stats count min(mag) max(mag) by Description
```

eval 命令用于新建一个名为 Description 的字段，该字段根据地震的 Depth 获取 "Low"、"Mid" 或 "Deep" 的值。case() 函数用于指定适合每种描述的深度等级。例如，若深度不足 70 公里，该地震将被描述为浅源地震，生成的描述 Description 因此为 Low。

该搜索还会通过管道符把 eval 命令的结果传递给 stats 命令，来统计地震的数量并显示每个描述的最大和最小震级。

统计选项卡中显示的结果如下所示：

描述	count	min(Mag)	max(Mag)
----	-------	----------	----------

Deep	35	4.1	6.7
低	6236	-0.60	7.70
Mid	635	0.8	6.3

您可单击 Splunk Web 中的排序图标对“说明”列的结果进行排序。而在此示例中，按字母顺序返回的结果可能为 Deep、Low、Mid 或 Mid、Low、Deep 顺序。

您还可以使用 case 函数对自定义顺序（如 Low、Mid、Deep）中的结果进行排序。您可以新建自定义排序顺序，方法是：为值提供数字排名，然后根据排名进行排序。

```
source=all_month.csv | eval Description=case(depth<=70, "Low", depth>70 AND depth<=300, "Mid", depth>300, "Deep") | stats count min(mag) max(mag) by Description | eval sort_field=case(Description="Low", 1, Description="Mid", 2, Description="Deep",3) | sort sort_field
```

统计选项卡中显示的结果如下所示：

描述	count	min(Mag)	max(Mag)
低	6236	-0.60	7.70
Mid	635	0.8	6.3
Deep	35	4.1	6.7

cidrmatch("X", Y)

描述

根据 IP 地址是否匹配 CIDR 表示法返回 TRUE 或 FALSE。

使用此函数确定 IP 地址是否属于特定子网。若 IP 地址 Y 属于特定子网 X，则此函数返回结果为 true。X 和 Y 都是字符串参数。X 是 CIDR 子网。Y 是和子网匹配的 IP 地址。此函数与 IPv6 兼容。

用法

您可以将此函数和 eval、fieldformat 及 where 命令结合使用，并作为 eval 表达式的一部分。

基本示例

以下示例使用 cidrmatch 和 if 函数将字段 isLocal 设为 "local"，前提是字段 ip 和子网匹配。如果 ip 字段和子网不匹配，则将 isLocal 字段设为 "not local"。

```
... | eval isLocal;if(cidrmatch("123.132.32.0/25",ip), "local", "not local")
```

以下示例使用 cidrmatch 函数作为筛选器以移除不匹配 ip 地址的事件：

```
... | where cidrmatch("123.132.32.0/25", ip)
```

coalesce(X, ...)

描述

此函数会获取任意数量的参数并返回第一个不是空值的值。

用法

您可以将此函数和 eval、fieldformat 及 where 命令结合使用，并作为 eval 表达式的一部分。

基本示例

您有一组事件，其 IP 地址已提取至 clientip 或 ipaddress。本示例定义一个名为 ip 的新字段，该字段获取 clientip 字段或 ipaddress 字段的值，具体取决于不是 NULL（不存在于该事件中）的那一个字段。如果 clientip 和 ipaddress 字段都存在于事

件中，此函数将返回第一个参数，即字段 clientip。

```
... | eval ip=coalesce(clientip,ipaddress)  
false()
```

描述

使用此函数返回 FALSE。

此函数使您能够指定明显为 false 的条件，例如 1==0。您不使用此函数指定字段。

用法

此函数通常用作其他函数的参数。

您可以将此函数和 eval、fieldformat 及 where 命令结合使用，并作为 eval 表达式的一部分。

基本示例

```
if(X, Y, Z)
```

描述

如果条件 X 评估为 TRUE，返回 Y，否则返回 Z。

此函数获取三个参数。第一个参数 X 必须是布尔表达式。如果 X 值为 TRUE，则结果为第二个参数 Y。如果 X 值为 FALSE，则结果为第三个参数 Z。

用法

您可以将此函数和 eval、fieldformat 及 where 命令结合使用，并作为 eval 表达式的一部分。

if 函数经常和其他函数结合使用。请参阅 [基本示例](#)。

基本示例

以下示例会查看字段 error 的值。如果 error=200，则函数将返回 err=OK，否则函数将返回 err=Error。

```
... | eval err;if(error == 200, "OK", "Error")
```

以下示例使用 cidrmatch 和 if 函数将字段 isLocal 设为 "local"，前提是字段 ip 和子网匹配。如果 ip 字段和子网不匹配，则将 isLocal 字段设为 "not local"。

```
... | eval isLocal;if(cidrmatch("123.132.32.0/25",ip), "local", "not local")
```

in(FIELD, VALUE-LIST)

描述

如果列表中的一个值和您指定字段中的值匹配，则函数返回 TRUE。

此函数会采用以逗号分隔的值的列表。

用法

您可以将此函数和 eval、fieldformat 及 where 命令结合使用，并作为 eval 表达式的一部分和其他命令结合使用。

支持以下语法：

```
...| where in(field,"value1","value2", ...)  
...| where field in("value1","value2", ...)  
...| eval new_field;if(in(field,"value1","value2", ...), "value-if_true","value-if-false")
```

eval 命令不接受布尔值。您必须在 IF 函数中指定 IN 函数，IF 函数可接受布尔值作为输入。

字符串值必须用引号引起。您不能用值指定通配符以指定一组类似的值，如 HTTP 错误代码或 CIDR IP 地址范围。可改用 IN 运算符。

IN 运算符和 in 函数类似。您可将 IN 运算符和 search 和 tstats 命令结合使用。您可以将 VALUE-LIST 中的通配符和这些命令结合使用。

基本示例

如果 status 字段中一个值与列表中一个值匹配，以下示例使用 where 命令返回 in=TRUE。

```
... | where status in("400", "401", "403", "404")
```

以下示例使用 in 函数作为 if 函数的第一个参数。如果 status 字段中的值匹配列表中其中一个值，评估表达式返回 TRUE。

```
... | eval error=if(in(status, "error", "failure", "severe"),"true","false")
```

延伸示例

以下示例将 in 函数和 if 函数结合使用以评估 status 字段。如果 status 字段包含值 404、500 或 503 中的一个，true 值放在新字段 error 中。然后统计 error 字段中这些值的数量。

```
... | eval error=if(in(status, "404","500","503"),"true","false") | stats count by error
```

另请参阅

博客

[Smooth 运算符 | 搜索多个字段值](#)

like(TEXT, PATTERN)

描述

如果 TEXT 匹配 PATTERN，此函数返回 TRUE。

此函数将采用两个参数，一个与 TEXT 匹配的字符串和一个与 PATTERN 匹配的字符串表达式。当且仅当 TEXT 和 PATTERN 匹配时返回 TRUE。模式匹配支持精确的文本匹配以及单个和多个字符匹配。

- 将（%）符号用作多个字符的通配符。
- 将下划线（_）字符用于单个字符匹配。

用法

您可以将此函数和 eval、fieldformat 及 where 命令结合使用，并作为 eval 表达式的一部分。

基本示例

以下示例在字段值以 foo 开头时，返回 like=TRUE：

```
... | eval is_a_foo=if(like(field, "foo%"), "yes a foo", "not a foo")
```

如果 ipaddress 字段以值 198. 开头，以下示例使用 where 命令返回 like=TRUE：百分号（%）是具有 like 函数的通配符：

```
... | where like(ipaddress, "198.%")
```

lookup(<lookup_table>, <json_object>, <json_array>)

描述

此功能执行 CSV 查找。它以 JSON 对象的形式返回一个或多个输出字段。

语法

```
lookup("<lookup_table>", json_object("<input_field>", <match_field>, ...), json_array("<output_field>", ...))
```

用法

您可以将 `lookup()` 函数和 `eval`、`fieldformat` 及 `where` 命令结合使用，并作为 `eval` 表达式的一部分。

`lookup()` 函数从 CSV `<lookup_table>` 中获取一个 `<input_field>`，在搜索结果中查找有 `<match_field>` 的事件，然后从 CSV 表中识别对应于 `input_field` 的其他字段值对，并将它们以 JSON 对象的形式添加到匹配的事件中。

`lookup()` 需要 `<lookup_table>`。您可以提供 CSV 查找文件或 CSV 查找定义，并用引号括起来。要提供文件，请提供 CSV 查找文件的完整文件名，该文件存储在全局查找目录 (`$SPLUNK_HOME/etc/system/lookups/`) 或与您当前应用上下文匹配的查找目录中，例如 `$SPLUNK_HOME/etc/users/<user>/<app>/lookups/`。

如果第一个带引号的字符串没有以 `".csv"` 结尾，则 `eval` 处理器会假定它是 CSV 查找定义的名称。指定的 CSV 查找定义必须全局共享。CSV 查找定义不能设为私有或只共享给特定的应用程序。

如果您希望应用与定义关联的各种设置，例如匹配限制、区分大小写的匹配选项等，请指定查找定义。

`lookup()` 函数可以使用多个 `<input_field>/<match_field>` 对来标识事件，并且可以将多个 `<output_field>` 值应用于这些事件。以下是带有多个输入、匹配项和输出的有效 `lookup()` 语法的示例。

```
... | eval <string>=lookup("<lookup_table>", json_object("<input_field1>", <match_field1>, "<input_field2>", <match_field2>), json_array("<output_field1>", "<output_field2>", "<output_field3>")
```

有关上载 CSV 查找文件和创建 CSV 查找定义的更多信息，请参阅《知识管理器手册》中的“在 Splunk Web 中定义 CSV 查找”。

对于 `eval`、`lookup()` 函数使用两个 JSON 函数：`json_object` 和 `json_array`。JSON 函数使 `eval` 处理器可以有效地将事物组合在一起。有关更多信息，请参阅《搜索参考》中的“JSON 函数”。

示例

以下示例说明了使用 `lookup()` 函数的不同方法。

1. 返回带数组的 JSON 对象的简单示例

这个简单的 `makeresults` 示例会返回一个数组，该数组说明了在 `status_type` 为 `Successful` 的 `http_status.csv` 查找表中配对了哪些 `status_description` 值。

该搜索返回以下内容： `output={"status_description":["OK","Created","Accepted","Non-Authoritative Information","No Content","Reset Content","Partial Content"]}`

```
| makeresults | eval type = "Successful" | eval output=lookup("http_status.csv", json_object("status_type", type), json_array("status_description"))
```

2. 带多个输入和匹配字段对的搜索示例

此搜索使用多个输入和匹配字段对来说明，`type="Successful"` 且 `status="200"` 的事件与 `http_status.csv` 查找表中值为 `OK` 的 `status_description` 匹配。

该搜索返回以下内容： `output={"status_description":"OK"}`

```
| makeresults | eval type = "Successful", status="200" | eval output=lookup("http_status.csv", json_object("status_type", type, "status", status), json_array("status_description"))
```

3. 获取 HTTP 状态描述和类型对的计数

本示例将 `http_status.csv` 查找文件中的 `status` 字段的值与返回的事件中的 `status` 字段的值进行匹配。然后，它会生成 JSON 对象（作为 `status_details` 字段的值），以及相应的 `status_description` 和 `status_type` 字段-值对，并将它们添加到事件中。最后，它提供了按对象细分的 JSON 对象计数。

以下是由此搜索返回的一个 JSON 对象示例。

```
status_details=JSON:{ "status_description": "Created", "status_type": "Successful" }
```

```
index=_internal | eval output=lookup("http_status.csv", json_object("status", status), json_array("status_description", "status_type")), status_details="JSON:".output | stats count by status_details
```

4. 获取已通过 HTTP 状态 `eval` 查找应用于事件的 HTTP 状态描述值的计数

本示例说明您可以如何将 `lookup` 函数嵌套在另一个 `eval` 函数中。本例所用的是 `json_extract` JSON 函数。这将从

`json_array` 对象中提取 `status_description` 字段-值对，并将它们应用于相应的事件。然后，此搜索会返回带有 `status_description` 字段的事件的计数，并按 `status_description` 的值细分。

以下是由此搜索返回的一个已提取的 `status_description` 示例。将其与第三个示例返回的结果进行比较：

`status_details=Created`

```
index=_internal | eval status_details=json_extract(lookup("http_status.csv", json_object("status", status),  
json_array("status_description")), "status_description") | stats count by status_details
```

match(SUBJECT, "REGEX")

描述

根据 REGEX 是否匹配 SUBJECT，此函数返回 TRUE 或 FALSE。

此函数将正则表达式字符串 REGEX 与 SUBJECT 值进行比较并返回一个布尔值。如果 REGEX 能找到 SUBJECT 任意子串的匹配，返回 TRUE。

用法

match 函数基于正则表达式。例如，使用反斜杠（\）字符转义特殊字符（如引号）。使用管道符（|）指定 OR 条件。

您可以将此函数和 eval、fieldformat 及 where 命令结合使用，并作为 eval 表达式的一部分。

基本示例

当且仅当 field 与 IP 地址的基本模式匹配时，以下示例返回 TRUE。此示例将使用脱字符（^）字符和美元（\$）符号执行完全匹配。

```
... | eval n;if(match(field, "^\\d{1,3}\\.\\d{1,3}\\.\\d{1,3}\\.\\d{1,3}$"), 1, 0)
```

以下示例在 <eval-expression> 中使用 match 函数。SUBJECT 是名为 test 的已计算字段。“REGEX” 是字符串 yes。

本示例在 <eval-expression> 中使用 match 函数。SUBJECT 是名为 test 的已计算字段。“REGEX” 是字符串 yes。

```
... | eval matches = if(match(test,"yes"), 1, 0)
```

如果值存储在引号中，您必须使用反斜杠（\）字符转义嵌入的引号。例如：

```
| makeresults | eval test="\\"yes\\\" | eval matches = if(match(test, '\"yes\""), 1, 0)
```

null()

描述

此函数不获取任何参数并返回 NULL。评估引擎使用 NULL 代表“无值”。将字段值设为 NULL 会清空字段值。

用法

NULL 值是特定结果中缺失但存在于其他结果中的字段值。

您可以将此函数和 eval、fieldformat 及 where 命令结合使用，并作为 eval 表达式的一部分。

基本示例

假设您想要计算某字段值的平均值，但是有几个值为零。如果零是无值的占位符，那么零会影响平均值的精确度。您可以使用 null 函数删除零。

另请参阅

- 您可以使用 fillnull 命令用指定值替换 NULL 值。
- 您可以使用 nullif(X,Y) 函数比较两个字段，且如果 X = Y，则返回 NULL。

nullif(X, Y)

描述

此函数用于比较字段。该函数获取两个参数，`X` 和 `Y`，若 `X = Y` 将返回空值，否则将返回 `X`。

用法

您可以将此函数和 `eval`、`fieldformat` 及 `where` 命令结合使用，并作为 `eval` 表达式的一部分。

基本示例

如果 `fieldA=fieldB`，以下示例将返回 `NULL`。否则函数返回 `fieldA`。

```
... | eval n=nullif(fieldA,fieldB)
```

searchmatch(X)

描述

如果搜索字符串（`X`）匹配事件，使用此函数返回 `TRUE`。

此函数获取一个搜索字符串参数 `X`。当且仅当事件与搜索字符串匹配时，函数返回 `TRUE`。

用法

您必须使用 `if` 函数中的 `searchmatch` 函数。

您可以将此函数和 `eval`、`fieldformat` 及 `where` 命令结合使用，并作为 `eval` 表达式的一部分。

基本示例

以下示例使用 `makeresults` 命令创建一些简单结果。`searchmatch` 函数用于确定是否有任何结果与搜索字符串 "`x=hi y=*`" 匹配。

```
| makeresults 1 | eval _raw = "x=hi y=bye" | eval x="hi" | eval y="bye" | eval test=if(searchmatch("x=hi y=*"), "yes", "no") | table _raw test x y
```

`if` 函数的结果为 `yes`；此结果与使用 `searchmatch` 函数指定的搜索字符串匹配。

true()

描述

使用此函数返回 `TRUE`。

您可使用此函数指定明显为 `true` 的条件，例如 `1==1`。您不使用此函数指定字段。

用法

此函数通常用作其他函数的参数。

您可以将此函数和 `eval`、`fieldformat` 及 `where` 命令结合使用，并作为 `eval` 表达式的一部分。

基本示例

此示例使用搜索教程中的示例数据，但应该可以与任意格式的 Apache Web 访问日志结合使用。要在您自己的 Splunk 实例中尝试此示例，您必须下载示例数据并遵循操作以将教程数据导入 Splunk。当运行搜索时，请使用时间范围昨天。

以下示例显示如何使用 `true()` 函数为 `case` 函数提供默认值。如果状态字段中的值不是 `200` 或 `404`，使用的值为“其他”。

```
sourcetype=access_* | eval description=case(status==200,"OK", status==404, "Not found", true(), "Other") | table status description
```

统计选项卡中显示的结果如下所示：

status	描述
200	正常
200	正常
408	其他
200	正常
404	未找到
200	正常
200	正常
406	其他
200	正常

validate(X, Y, ...)

描述

使用此函数返回与评估为 FALSE 的第一个表达式 X 对应的字符串 Y。此函数与 case 函数对立。

此函数含参数对、布尔值表达式 X 和字符串 Y。此函数返回与评估为 FALSE 的首个表达式 X 相对应的字符串 Y。如果所有表达式都评估为 TRUE，此函数默认为空。

用法

您可以将此函数和 eval、fieldformat 及 where 命令结合使用，并作为 eval 表达式的一部分。

基本示例

以下示例是对有效端口的一个简单检查。

```
... | eval n=validate(isint(port), "ERROR: Port is not an integer", port >= 1 AND port <= 65535, "ERROR: Port is out of range")
```

转换函数

以下列表包含了您可以用于将数字和字符串相互转换的函数。

有关在函数和嵌套函数中使用字符串和数字字段的信息，请参阅“评估函数”。

printf("format", arguments)

描述

printf 函数会根据您指定的字符串格式和参数构建字符串值。

- 可不指定参数或指定多个参数。参数可以是字符串值、数字、计算或字段。

SPL printf 函数与 C sprintf() 函数相似，也与像 Python、Perl 和 Ruby 等其他语言中的函数相似。

用法

您可以将此函数和 eval、fieldformat 及 where 命令结合使用，并作为 eval 表达式的一部分。

format

描述：format 是可以包括一个或多个格式转换指示符的字符串。每个转换指示符均可包括可选组件，如标记字符、宽度规格和精度规格。format 必须用引号引起来。

语法："(%[flags][width][.precision]<conversion_specifier>)..."

参数

描述：arguments 为可选参数并可包括宽度、精度和要设置格式的值。值可以是字符串、数字或字段名称。

语法: [width][.precision][value]

支持的转换指示符

下表描述了支持的转换指示符。

转换指示符	别名	描述	示例
%a 或 %A		十六进制的浮点数	此示例将返回十六进制的 <i>pi</i> 值，保留 3 位小数。 <code>printf("%.3A",pi())</code> 将返回 <code>0X1.922P+1</code>
%c		单一 Unicode 码位	此示例将返回 unicode 码位 65 以及字符串 "Foo" 的第一个字母。 <code>printf("%c,%c",65,"Foo")</code> 将返回 <code>A,F</code>
%d	%i	有符号的十进制整数	此示例将返回正整数或负整数，包括这些值指定的任何符号。 <code>printf("%d,%i,%d",-2,+4,30)</code> 将返回 <code>-2,4,30</code>
%e 或 %E		指数格式的浮点数	此示例将以指数格式返回数字 5139，保留两位小数。 <code>printf("%.2e",5139)</code> 将返回 <code>5.14e+03</code>
%f 或 %F		浮点数	此示例将返回保留 2 位小数的 <i>pi</i> 值。 <code>printf("%.2f",pi())</code> 将返回 <code>3.14</code>
%g 或 %G		浮点数。此指示符将根据要设置格式的数字范围确定使用 %e 还是 %f。	此示例将返回保留 2 位小数的 <i>pi</i> 值（使用 %f 指示符），并以指数格式返回数字 123，保留两位小数（使用 %e 指示符）。 <code>printf("%.2g,.2g",pi(),123)</code> 将返回 <code>3.1,1.2e+02</code>
%o		无符号的八进制数字	此示例将返回 255 的八进制数字。 <code>printf("%o",255)</code> 将返回 <code>377</code>
%s	%z	字符串	此示例将返回 "foo" 和 "bar" 的已连接的字符串值。 <code>printf("%s%z", "foo", "bar")</code> 将返回 <code>foobar</code>
%u		无符号或非负十进制整数	此示例将返回参数中的整数值。 <code>printf("%u",99)</code> 返回 <code>99</code>
%x 或 %X	%p	无符号的十六进制数（小写或大写）	此示例将返回与参数中数字相等的十六进制值。使用此指示符时，此示例将显示大小写两种结果。 <code>printf("%x,%X,%p",10,10,10)</code> 将返回 <code>a,A,A</code>
%%		百分号	此示例将返回带百分号的字符串值。 <code>printf("100%%")</code> 将返回 <code>100%</code>

标记字符

下表描述了支持的标记字符。

标记字符	描述	示例
------	----	----

单引号或撇号 (')	添加逗号作为千分位分隔符。	printf("%'d", 12345) 将返回 12,345
短划线或减号 (-)	左对齐。如果未指定此标志，则结果会保留其默认对齐方式。 printf 函数只有默认采用右对齐方式时才支持以右对齐方式显示结果。	printf("%-4d", 1) 将返回 1 在输出中采用左对齐方式。
零 (0)	零填充	此示例将返回参数中的值，值前面补零，这样该值就有 4 位。 printf("%04d", 1) 将返回 0001
加号 (+)	始终包括符号 (+ 或 -)。如果未指定此标记，则转换只显示复制的标记。	printf("%+4d", 1) 将返回 +1
<space>	预留标记空格。如果有符号转换的第一个字符并非符号或有符号转换无字符，则在输出结果前面补以 <space>。如果 <space> 和 + 标记都指定了，则忽略 <space> 标记。	printf("% -4d", 1) 将返回 1
哈希值、数字或井号键 (#)	使用另一种形式。对于 %o 转换指示符，# 标记会增加精确度，使结果的第一位为零。对于 %x 或 %X 转换指示符，非零结果输出加前缀 0x (或 0X)。对于 %a、%A、%e、%E、%f、%F、%g 和 G 转换指示符，结果始终包含基数字符，即使基数字符后无数字。如无此标记，只有基数字符后有数字时，该基数字符才会出现在这些转换结果中。对于 %g 和 %G 转换指示符，会保持尾随零原样不动，不会将这些零从结果中删除。其他转换指示符的行为尚未定义。	printf("%#x", 1) 将返回 0x1

指定字段宽度

您可以将星号 (*) 和 printf 函数结合使用，以返回参数中的字段宽度或精度。

示例

以下示例将返回正整数值或负整数值，包括这些值指定的任何符号。

```
printf("%*d", 5, 123) 将返回 123
```

以下示例中，返回带 1 个小数位的浮点数。

```
printf("%.1f", 1, 1.23) 将返回 1.2
```

此示例将以指数格式返回 pi() 值，保留两位小数。

```
printf("%.*e", 9, 2, pi()) 将返回 3.14e+00
```

可使用数字或星号 (*) 字符所表示的参数表示字段宽度。

字段宽度指示符	描述	示例
数字	要打印的最少字符数。如果要打印的值少于此数，结果则补以空格。即使结果较大，该值也不会被截断。	
* (星号)	未在格式字符串中指定宽度，但是作为要设置格式的参数之前的其他整数值。	

指定精度

精度	描述
%d、%i、%o、%u、%x 或 %X	精度会指定要返回的最少位数。如果要返回的值少于此数，将补以前导零。即使结果较长，值也不会被截断。若精度为 0，则表示没有为值 0 返回字符。

%d 或 %A、%e 或 %E、%l 或 %F	这是要返回的小数点后的位数。默认值为 6。
%g 或 %G	这是要返回的有效数字的最大数。
%s	这是要返回的字符的最大数量。默认打印所有字符，除非以无效字符结束。
指定没有精确值的期限	如果指定的期限没有明确的精确值，则假定为 0。
指定星号为精确值，例如 .*	未在格式字符串中指定精确值，但是作为要设置格式的参数之前的其他整数值参数。

不支持的转换指示符

不支持以下几种 C `sprintf()` 函数的转换指示符，包括：

- %C，但支持 %c
- %n
- %S，但支持 %s
- 用于挑选要使用哪种参数的 %<num>\$ 指示符

基本示例

此示例将新建名为 `new_field` 的字段，并根据 `field_one` 和 `field_two` 中的值新建字符串值。这些值格式设为小数点前后各 4 位。- 指定为左对齐字符串值。30 指定字段宽度。

```
... | eval new_field=printf("%04.4f %-30s", field_one, field_two)
```

tonumber (NUMSTR, BASE)

描述

此函数将输入字符串 `NUMSTR` 转换为数字。`NUMSTR` 可以是一个字段名称或字段值。

用法

您可以将此函数和 `eval`、`fieldformat` 及 `where` 命令结合使用，并作为 `eval` 表达式的一部分。

`BASE` 为可选参数，用于定义 `NUMSTR` 中的数字进制。`BASE` 可以是 2 到 36。默认是 10 以对应十进制系统。

如果 `tonumber` 函数无法将字段值解析为一个数字，例如，如果值包含前导和尾随空格，函数返回空值。使用 `trim` 函数将前导或尾随空格删除。

如果 `tonumber` 函数无法将文字字符串解析为数字，它将返回一个错误。

基本示例

以下示例将 `store_sales` 字段的字符串值转换为数字。

```
... | eval n=tonumber(store_sales)
```

以下示例以十六进制数为例，以 16 为 `BASE` 返回数字 "164"。

```
... | eval n=tonumber("0A4",16)
```

以下示例在将 `celsius` 字段中的值转换为数字之前，先修剪该字段的所有前导或尾随空格。

```
... | eval temperature=tonumber(trim(celsius))
```

tostring (X, Y)

描述

此函数将输入值转换为字符串。如果输入值为数字，则将其重新格式化为字符串。如果输入值为布尔值，则返回对应的字符串

值，即 "True" 或 "False"。

用法

您可以将此函数和 eval、fieldformat 及 where 命令结合使用，并作为 eval 表达式的一部分。

此功能需要至少一个参数 X。

与 eval 命令一起使用时，这些值会转换成 ASCII 码，因此无法以预期的方式排序。将 fieldformat 命令和 tostring 函数合并使用即可对显示的值进行格式化。fieldformat 命令不会更改基础值。

若 X 是数字，第二个参数 Y 为可选参数，此时 Y 可以是 "hex"、"commas" 或 "duration"。

示例	描述
tostring(X, "hex")	将 X 转换为十六进制。
tostring(X, "commas")	用逗号转换 X 格式。如果数字包括小数点，函数将四舍五入到最接近的两位小数。
tostring(X, "duration")	将秒数 X 转换为易读时间格式 HH:MM:SS。

基本示例

以下示例返回 "True 0xF 12, 345. 68"。

```
... | eval n=tostring(1==1) + " " + tostring(15, "hex") + " " + tostring(12345.6789, "commas")
```

以下示例将返回 foo=615 和 foo2=00:10:15。615 秒转化成分和秒。

```
... | eval foo=615 | eval foo2 = tostring(foo, "duration")
```

以下示例将 totalSales 列格式设为显示带货币符号和逗号的值。在货币值与 tostring 函数之间必须使用句点。

```
... | fieldformat totalSales="$".tostring(totalSales,"commas")
```

另请参阅

命令

convert

函数

strptime

加密函数

以下列表包含了您可以用于计算字符串值的安全哈希值的函数。

有关在函数和嵌套函数中使用字符串和数字字段的信息，请参阅“评估函数”。

md5 (X)

描述

此函数计算并返回字符串值 X 的 MD5 哈希值。

用法

您可以将此函数和 eval、fieldformat 及 where 命令结合使用，并作为 eval 表达式的一部分。

基本示例

以下示例针对词组 "Hello World" 返回一个带 message-digest (MD5) 128 位哈希值的新字段 n。

```
... | eval n=md5("Hello World")
```

以下示例会新建一个大的随机字符串。

```
| makeresults count=32768 | eval message=md5("$. random()) | stats values(message) as message | eval message = mvjoin(message, "")
```

- `makeresults` 命令创建 32768 个带时间戳的结果。
- `eval` 命令创建名为 `message` 的新字段：
 - `random` 函数会为 32768 个结果中的每个返回随机的数字字段值。`"$."` 会将 `random` 函数生成的数字转换为字符串值。
 - `md5` 函数会通过字符串值创建 128 位哈希值。
 - `md5` 函数的结果被放在 `eval` 命令创建的 `message` 字段中。
- 具有 `values` 函数的 `stats` 命令被用于将各随机值转换为一个多值结果。
- 具有 `mvjoin` 函数的 `eval` 命令被用于将多值条目和单一值结合起来。

sha1(X)

描述

此函数计算并返回基于兼容 FIPS 的 SHA-1 哈希函数的字符串值 X 的安全哈希值。

用法

您可以将此函数和 `eval`、`fieldformat` 及 `where` 命令结合使用，并作为 `eval` 表达式的一部分。

基本示例

```
... | eval n=sha1("Put that in your | and Splunk it.")
```

sha256(X)

描述

此函数计算并返回基于兼容 FIPS 的 SHA-256 哈希函数的字符串值 X 的安全哈希值。

用法

您可以将此函数和 `eval`、`fieldformat` 及 `where` 命令结合使用，并作为 `eval` 表达式的一部分。

基本示例

```
... | eval n=sha256("Can you SPL?")
```

sha512(X)

描述

此函数计算并返回基于兼容 FIPS 的 SHA-512 哈希函数的字符串值 X 的安全哈希值。

用法

您可以将此函数和 `eval`、`fieldformat` 及 `where` 命令结合使用，并作为 `eval` 表达式的一部分。

基本示例

```
... | eval n=sha512("You bet your sweet SaaS.")
```

日期和时间函数

以下列表包含了您可以用于计算日期和时间的函数。

有关在函数和嵌套函数中使用字符串和数字字段的信息，请参阅“评估函数”。

除了本主题中列出的函数外，还有一些您可以在搜索中使用的变量和调节器。

- 日期和时间格式变量
- 时间调节器

now()

描述

此函数不获取任何参数，它将返回搜索的开始时间。

用法

now() 函数通常和其他日期和时间函数结合使用。

now() 函数返回的时间以 Unix 时间或秒表示（Epoch 时间之后）。

在搜索中使用时，搜索运行后，此函数返回 UNIX 时间。如果您想要在返回每个结果后，返回 UNIX 时间，请使用 time() 函数。

您可以将此函数和 eval、fieldformat 及 where 命令结合使用，并作为 eval 表达式的一部分。

基本示例

以下示例根据 now() 值确定昨天开始的 UNIX 时间值。

```
... | eval n=relative_time(now(), "-1d@d")
```

延伸示例

如果您要寻找发生在之前 30 分钟内的事件，您需要计算事件小时、事件分钟、当前小时和当前分钟。使用 now() 函数计算当前小时 (curHour) 和当前分钟 (curMin)。_time 字段中的事件时间戳用于计算事件小时 (eventHour) 和事件分钟 (eventMin)。例如：

```
... earliest=-30d | eval eventHour=strftime(_time,"%H") | eval eventMin=strftime(_time,"%M") | eval curHour=strftime(now(),"%H") | eval curMin=strftime(now(),"%M") | where (eventHour=curHour and eventMin > curMin - 30) or (curMin < 30 and eventHour=curHour-1 and eventMin>curMin+30) | bucket _time span=1d | chart count by _time
```

relative_time(X, Y)

描述

此函数获取 UNIX 时间 X 作为第一个参数以及相对时间指示符 Y 作为第二个参数，并返回应用于 X 的 Y 的 UNIX 时间值。

用法

您可以将此函数和 eval、fieldformat 及 where 命令结合使用，并作为 eval 表达式的一部分。

基本示例

以下示例根据 now() 值确定昨天开始的 UNIX 时间值。

```
... | eval n=relative_time(now(), "-1d@d")
```

strftime(X, Y)

描述

此函数取 UNIX 时间值 X 作为第一个参数，用由 Y 指定的格式将时间作为字符串呈现。UNIX 时间必须以秒为单位。使用 UNIX 时间的前 10 位以使用秒计时间。

用法

如果时间以毫秒、微秒或纳秒计，您必须将时间转换为秒。您可以使用 `pow` 函数转换数字。

- 要从毫秒转换为秒时，将数字除以 1000 或 10^3 。
- 要从微秒转换为秒时，将数字除以 10^6 。
- 要从纳秒转换为秒时，将数字除以 10^9 。

以下搜索使用 `pow` 函数将纳秒转换为秒：

```
| makeresults | eval StartTimestamp="1521467703049000000" | eval starttime=strftime(StartTimestamp/pow(10,9),"%Y-%m-%dT%H:%M:%S.%Q")
```

结果显示在“统计”选项卡中，如下所示：

StartTimeStamp	_time	starttime
1521467703049000000	2018-08-10 09:04:00	2018-03-19T06:55:03.049

在这些结果中，`_time` 值是运行搜索的日期和时间。

有关格式选项的列表和描述，请参阅“常见的时间格式变量”。

您可以将此函数和 `eval`、`fieldformat` 及 `where` 命令结合使用，并作为 `eval` 表达式的一部分。

基本示例

以下示例将返回 `_time` 字段的小时和分钟。

```
... | eval hour_min=strftime(_time, "%H:%M")
```

如果 `_time` 字段值是 2018-08-10 11:48:23，则 `hour_min` 字段中的返回值是 11:48。

以下示例在搜索结果中创建一个名为 `starttime` 的新字段。对于 `strftime` 值，`now()` 函数用于生成当前 UNIX 时间，日期时间变量用于指定 ISO 8601 时间戳格式；

```
... | eval starttime=strftime(now(), "%Y-%m-%dT%H:%M:%S.%Q")
```

结果形式如下所示：

_starttime
2021-02-11T01:55:00.000

有关日期和时间变量的更多信息，请参阅“日期和时间格式变量”。

延伸示例

以下示例使用 `makeresults` 命令新建一个单一结果。

```
| makeresults
```

例如：

_time
2018-08-14 14:00:15

`_time` 字段存储在 UNIX 时间中，尽管该字段以用户可读格式显示。要将 UNIX 时间转换为某些其他格式，您可将 `strftime` 函数和日期和时间格式变量结合使用。变量必须用引号引起。

例如，要返回事件发生那一年中的星期，请使用 `%V` 变量。

```
| makeresults | eval week=strftime(_time,"%V")
```

结果显示，8 月 14 日在第 33 周。

_time	周
	周

要返回带有次秒和时间指示器（即位于时间组件格式前面的字母 T）的日期和时间，请使用 %Y-%m-%dT%H:%M:%S.%Q 变量。例如：

```
| makeresults | eval mytime=strftime(_time,"%Y-%m-%dT%H:%M:%S.%Q")
```

结果是：

_time	mytime
2018-08-14 14:00:15	2018-08-14T14:00:15.000

strftime(X, Y)

描述

此函数获取由字符串 X 代表的时间，并将该时间解析为 UNIX 时间戳。您可使用日期和时间变量指定和字符串 X 匹配的格式 Y。

例如，如果字符串 X 是 2018-08-13 11:22:33，那么格式 Y 必须是 %Y-%m-%d %H:%M:%S。字符串 X 日期必须是 1971 年 1 月 1 日或者更晚的日期。

Time 字段是 UNIX 时间。在 Splunk Web 中，_time 字段在 UI 中以用户可读格式显示，但存储为 UNIX 时间。如果您想要使用 _time 字段中的 strftime 函数，则不对该字段中的值执行任何操作。

用法

使用 strftime 函数时，您必须指定字符串 X 的时间格式，这样函数可将字符串时间转换为正确的 UNIX 时间。以下表格提供一些示例：

字符串时间	匹配时间格式变量
Mon July 23 2018 17:19:01.89	%a %B %d %Y %H:%M:%S.%N
Mon 7/23/2018 17:19:01.89	%a %m/%d/%Y %H:%M:%S.%N
2018/07/23 17:19:01.89	%Y/%m/%d %H:%M:%S.%N
2018-07-23T17:19:01.89	%Y-%m-%dT%H:%M:%S.%N

有关格式选项的列表和描述，请参阅“常见的时间格式变量”。

您可以将此函数和 eval、fieldformat 及 where 命令结合使用，并作为 eval 表达式的一部分。

基本示例

如果 timeStr 字段中的值为小时和分钟，如 11:59，以下示例将时间返回为时间戳：

```
... | eval n=strptime(timeStr, "%H:%M")
```

延伸示例

此示例显示使用 strftime 函数的结果。以下搜索执行若干操作：

- gentimes 命令可生成一组时间，时间间隔为 6 小时。此命令将返回四个字段：starttime、starthuman、endtime 和 endhuman。
- fields 命令只返回 starthuman 和 endhuman 字段。
- eval 命令获取 starthuman 字段中的字符串时间值，并返回与该时间值相对应的 UNIX 时间。

```
| gentimes start=8/13/18 increment=6h | fields starthuman endhuman | eval startunix=strptime(starthuman,"%a %B %d %H:%M:%S.%N %Y")
```

统计选项卡中显示的结果如下所示：

starthuman	endhuman	startunix

Mon Aug 13 00:00:00 2018	Mon Aug 13 05:59:59 2018	534143600.000000
Mon Aug 13 06:00:00 2018	Mon Aug 13 11:59:59 2018	1534165200.000000
Mon Aug 13 12:00:00 2018	Mon Aug 13 17:59:59 2018	534186800.000000
Mon Aug 13 18:00:00 2018	Mon Aug 13 23:59:59 2018	1534208400.000000
Tue Aug 14 00:00:00 2018	Tue Aug 14 05:59:59 2018	1534230000.000000
Tue Aug 14 06:00:00 2018	Tue Aug 14 11:59:59 2018	1534251600.000000
Tue Aug 14 12:00:00 2018	Tue Aug 14 17:59:59 2018	1534273200.000000
Tue Aug 14 18:00:00 2018	Tue Aug 14 23:59:59 2018	1534294800.000000

time()

描述

此函数以微秒级的分辨率返回挂钟时间，格式为 UNIX 时间。

用法

eval 命令处理事件的时间不同，每个事件的 time() 函数值也会有所不同。

您可以将此函数和 eval、fieldformat 及 where 命令结合使用，并作为 eval 表达式的一部分。

基本示例

此示例显示使用 time() 函数的结果。以下搜索执行若干操作”

- gentimes 命令可生成一组时间，时间间隔为 6 小时。此命令将返回四个字段：starttime、starthuman、endtime 和 endhuman。
- fields 命令只返回 starttime 和 starthuman 字段。
- 第一个 eval 命令获取 starttime 字段中数字，并添加微秒数进行返回。
- 第二个 eval 命令新建 testtime 字段，并返回 eval 命令处理结果时的 UNIX 时间。

```
| gentimes start=8/13/18 increment=6h | fields starttime starthuman | eval epoch_time=strptime(starttime,"%s") | eval testtime=time()
```

统计选项卡中显示的结果如下所示：

starttime	starthuman	epoch_time	testtime
1534143600	Mon Aug 13 00:00:00 2018	1534143600.000000	1534376565.299298
1534165200	Mon Aug 13 06:00:00 2018	1534165200.000000	1534376565.299300
1534186800	Mon Aug 13 12:00:00 2018	1534186800.000000	1534376565.299302
1534208400	Mon Aug 13 18:00:00 2018	1534208400.000000	1534376565.299304
1534230000	Tue Aug 14 00:00:00 2018	1534230000.000000	1534376565.299305
1534251600	Tue Aug 14 06:00:00 2018	1534251600.000000	1534376565.299306
1534273200	Tue Aug 14 12:00:00 2018	1534273200.000000	1534376565.299308
1534294800	Tue Aug 14 18:00:00 2018	1534294800.000000	1534376565.299309

注意 epoch_time 和 test_time 字段之间值的微秒差异。您可以看到 test_time 值随结果而增加。

信息函数

以下列表包含了您可以用于返回值信息的函数。

有关在函数和嵌套函数中使用字符串和数字字段的信息，请参阅“评估函数”。

isbool(X)

描述

此函数获取一个参数 X 并评估 X 是否为布尔数据类型。若 X 是布尔数据类型，则函数返回 TRUE。

用法

将此函数和返回布尔数据类型的其他函数结合使用，如 cidrmatch 和 mvfind。

此函数无法用于确定字段值为 "true" 还是 "false"，因为字段值不是字符串，就是数字数据类型。可以改用 <fieldname>=true 或 <fieldname>=false 这类语法确定字段值。

您可以将此函数和 eval、fieldformat 及 where 命令结合使用，并作为 eval 表达式的一部分。

isint(X)

描述

此函数会获取一个参数 X，如果 X 为整型值，则返回 TRUE。

用法

您可以将此函数和 eval、fieldformat 及 where 命令结合使用，并作为 eval 表达式的一部分。

基本示例

以下示例会将 isint 函数和 if 函数结合使用。字段 "n" 添加到每个结果，根据 isint 函数的结果，值为 "int" 或 "not int"。如果 "field" 值是一个数字，那么 isint 函数返回 TRUE 且此值将添加值 "int" 到 "n" 字段。

```
... | eval n;if(isint(field),"int", "not int")
```

以下示例显示如何将 isint 函数和 where 命令结合使用。

```
... | where isint(field)
```

isnotnull(X)

描述

此函数会获取一个参数 X，如果 X 不是空值，则返回 TRUE。

用法

此函数可用于检查某个字段 (X) 是否包含值。

您可以将此函数和 eval、fieldformat 及 where 命令结合使用，并作为 eval 表达式的一部分。

基本示例

以下示例会将 isnotnull 函数和 if 函数结合使用。字段 "n" 添加到每个结果，根据 isnotnull 函数的结果，值为 "yes" 或 "no"。如果 "field" 值是一个数字，那么 isnotnull 函数会返回 TRUE 且此值会将值 "yes" 添加到 "n" 字段。

```
... | eval n;if(isnotnull(field),"yes", "no")
```

以下示例显示如何将 isnotnull 函数和 where 命令结合使用。

```
... | where isnotnull(field)
```

isnull(X)

描述

此函数会获取一个参数 X，如果 X 为空值，则返回 TRUE。

用法

您可以将此函数和 eval、fieldformat 及 where 命令结合使用，并作为 eval 表达式的一部分。

基本示例

以下示例会将 isnull 函数和 if 函数结合使用。字段 "n" 添加到每个结果，根据 isnull 函数的结果，值为 "yes" 或 "no"。如果结果中没有 "field" 值，isnull 函数返回 TRUE，向 "n" 字段添加值 "yes"。

```
... | eval n=if(isnull(field),"yes","no")
```

以下示例显示如何将 isnull 函数和 where 命令结合使用。

```
... | where isnull(field)
```

isnum(X)

描述

此函数会获取一个参数 X，如果 X 为数字，则返回 TRUE。

用法

您可以将此函数和 eval、fieldformat 及 where 命令结合使用，并作为 eval 表达式的一部分。

基本示例

以下示例会将 isnum 函数和 if 函数结合使用。字段 "n" 添加到每个结果，根据 isnum 函数的结果，值为 "yes" 或 "no"。如果 "field" 值是一个数字，那么 isnum 函数会返回 TRUE 且此值会将值 "yes" 添加到 "n" 字段。

```
... | eval n=if(isnum(field),"yes","no")
```

以下示例显示如何将 isnum 函数和 where 命令结合使用。

```
... | where isnum(field)
```

isstr(X)

描述

此函数会获取一个参数 X，如果 X 为字符串，则返回 TRUE。

用法

您可以将此函数和 eval、fieldformat 及 where 命令结合使用，并作为 eval 表达式的一部分。

基本示例

以下示例会将 isstr 函数和 if 函数结合使用。字段 "n" 添加到每个结果，根据 isstr 函数的结果，值为 "yes" 或 "no"。如果 "field" 值是一个字符串，那么 isstr 函数会返回 TRUE 且值会将值 "yes" 添加到 "n" 字段

```
... | eval n=if(isstr(field),"yes","no")
```

以下示例显示如何将 `isstr` 函数和 `where` 命令结合使用。

```
... | where isstr(field)  
typeof(X)
```

描述

此函数获取一个参数并返回代表该参数的数据类型。

用法

您可以将此函数和 `eval`、`fieldformat` 及 `where` 命令结合使用，并作为 `eval` 表达式的一部分。

基本示例

以下示例获取一个参数并返回代表该参数所属类型的字符串。以下示例返回 "NumberStringBoolInvalid"。

```
... | eval n=typeof(12) + typeof("string") + typeof(1==2) + typeof(badfield)
```

以下示例使用 `makeresults` 命令新建一个单一结果。

```
| makeresults
```

例如：

_time
2018-08-14 14:00:15

要确定 `_time` 字段的数据类型，请结合使用 `eval` 命令和 `typeof` 函数。例如：

```
| makeresults | eval t=typeof(_time)
```

结果是：

_time	t
2018-08-14 14:00:15	数字

JSON 函数

下表描述了可用于创建或操作 JSON 对象的函数：

描述	JSON 函数
从键值对创建一个新的 JSON 对象。	<code>json_object</code>
将元素附加到有效 JSON 对象的内容中。	<code>json_append</code>
使用值列表创建 JSON 数组。	<code>json_array</code>
将 JSON 数组的元素映射到多值字段。	<code>json_array_to_mv</code>
使用数组的值扩展有效 JSON 对象的内容。	<code>json_extend</code>
从字段和零个或多个路径返回 JSON 数组或 Splunk 软件本机类型值。	<code>json_extract</code>
从 JSON 对象中的键值对中返回键。键将以 JSON 数组的形式返回。	<code>json_keys</code>
使用提供的值插入或覆盖 JSON 节点的值，然后返回更新的 JSON 对象。	<code>json_set</code>
评估 JSON 对象是否使用有效的 JSON 语法并返回 TRUE 或 FALSE。	<code>json_valid</code>
将多值字段的元素映射到 JSON 数组。	<code>mv_to_json_array</code>

json_object(<members>)

根据键值对的成员创建一个新的 JSON 对象。

用法

如果为 `<key>` 或 `<value>` 指定一个字符串，则必须将此字符串用双引号引起来。`<key>` 必须是字符串。`<value>` 可以是字符串、数字、布尔值、空值、多值字段、数组或另一个 JSON 对象。

您可以将此函数和 `eval` 和 `where` 命令结合使用，并作为评估表达式的一部分和其他命令结合使用。

示例

这些示例说明了使用 `json_object` 函数在事件中创建 JSON 对象的不同方法。

1. 创建一个基本 JSON 对象

以下示例创建了一个基本 JSON 对象 `{ "name": "maria" }`。

```
... | eval name = json_object("name", "maria")
```

2. 使用多值字段创建 JSON 对象

以下示例创建了一个名为 `firstnames` 的多值字段，该字段使用键 `name` 并包含值 `"maria"` 和 `"arun"`。创建的 JSON 对象为 `{ "name": ["maria", "arun"] }`。

```
... | eval firstnames = json_object("name", json_array("maria", "arun"))
```

3. 使用 JSON 数组创建 JSON 对象

以下示例创建了一个使用 JSON 数组作为值的 JSON 对象。

```
... | eval locations = json_object("cities", json_array("London", "Sydney", "Berlin", "Santiago"))
```

结果为 JSON 对象 `{ "cities": ["London", "Sydney", "Berlin", "Santiago"] }`。

4. 创建一个嵌套的 JSON 对象

以下示例创建了一个嵌套的 JSON 对象，该对象使用其他 JSON 对象以及一个名为 `gamelist` 的多值或 JSON 数组字段。

```
... | eval gamelist = json_array("Pandemic", "Forbidden Island", "Castle Panic"), games = json_object("category",
json_object("boardgames", json_object("cooperative", gamelist)))
```

结果为以下 JSON 对象：

```
{
  "games": {
    "category": {
      "boardgames": {
        "cooperative": [ "Pandemic", "Forbidden Island", "Castle Panic" ]
      }
    }
  }
}
```

json_append(<json>, <path_value_pairs>)

此函数将值附加到 JSON 文档中指定数组的末尾处。它旨在提供等效于 `mvappend` 的 `eval` 函数。

用法

`json_append` 函数总是至少有三个函数输入：`<json>`（有效 JSON 文档的名称，例如 JSON 对象），以及至少一个 `<path>` 和 `<value>` 对。

如果 `<json>` 未引用有效的 JSON 文档（例如 JSON 对象），则该函数不输出任何内容。

`json_append` 从左到右评估 `<path_value_pairs>`。当 `json_append` 评估路径-值对时，它会更新 `<json>` 文档。然后将根据更新的文档评估下一个路径-值对。

您可以将此函数和 `eval` 和 `where` 命令结合使用，并作为评估表达式的一部分和其他命令结合使用。

使用 `<path>` 指定 JSON 文档值

每个 `<path>` 指定 `<json>` 文档中的一个数组或值。`json_append` 函数将相应的 `<value>` 添加到 `<path>` 指定的值的末尾。下表说明了 `json_append` 根据 `<path>` 指定的内容执行的操作。

如果 <code><path></code> 指定...	... 这是 <code>json_append</code> 对相应 <code><value></code> 所执行的操作
具有一个或多个值的数组。	<code>json_append</code> 将相应的 <code><value></code> 添加到该数组的末尾。
空数组	<code>json_append</code> 将相应的 <code><value></code> 添加到该数组，创建具有单个值的数组。
标量或对象值	<code>json_append</code> 自动包装数组中的标量或对象值，并将相应的 <code><value></code> 添加到该数组的末尾。

`json_append` 忽略 `<path>` 未标识 JSON 文档中任何有效值的路径-值对。

将数组作为单个元素附加

当新的 `<value>` 是一个数组时，`json_append` 将该数组作为单个元素附加。例如，如果 `json_array <path>` 指向数组 `["a", "b", "c"]`，其 `<value>` 是数组 `["d", "e", "f"]`，则结果是 `["a", "b", "c", ["d", "e", "f"]]`。

将数组作为单个元素进行附加，会将 `json_append` 与 `json_extend` 区分开来，后者是一个类似的函数，在附加数组和对象时将其展平为单独的元素。`json_extend` 以上一段为例，返回 `["a", "b", "c", "d", "e", "f"]`。

示例

以下示例说明了如何使用 `json_append` 将值附加到 JSON 文档中的数组。

将字符串添加到数组

假设您有一个名为 `ponies` 的对象，其中包含一个名为 `ponylist` 的数组：`["Minty", "Rarity", "Buttercup"]`。这是将 `"Fluttershy"` 附加到 `ponylist` 时运行的搜索。

```
... | eval ponies = json_object("ponylist", json_array("Minty", "Rarity", "Buttercup")), updatePonies = json_append(ponies, "ponylist", "Fluttershy")
```

该 `eval` 语句的输出是 `{"ponylist": ["Minty", "Rarity", "Buttercup", "Fluttershy"]}`。

将字符串附加到嵌套对象

此示例包含一个值为 `Fluttershy.ponySkills` 的 `<path>`。`Fluttershy.ponySkills` 引用了嵌套在源对象 `ponyDetails` 中的对象数组。该查询使用 `json_append` 将字符串添加到嵌套对象数组中。

```
... | eval ponyDetails = json_object("Fluttershy", json_object("ponySkills", json_array("running", "jumping"))), ponyDetailsUpdated = json_append(ponyDetails, "Fluttershy.ponySkills", "codebreaking")
```

此 `eval` 语句的输出是 `ponyDetailsUpdated = {"Fluttershy": {"ponySkills": ["running", "jumping", "codebreaking"]}}`

`json_array(<values>)`

使用值列表创建 JSON 数组。

用法

`<value>` 可以是任何类型的值，例如字符串、数字或布尔值。您也可以使用 `json_object` 函数来指定值。

您可以将此函数和 `eval` 和 `where` 命令结合使用，并作为评估表达式的一部分和其他命令结合使用。

示例

这些示例说明了使用 `json_array` 函数在事件中创建 JSON 数组的不同方法。

创建一个基本 JSON 数组

以下示例创建了一个简单数组 `["buttercup", "fluttershy", "rarity"]`。

```
... | eval ponies = json_array("buttercup", "fluttershy", "rarity")
```

通过字符串和 JSON 对象创建 JSON 数组

以下示例使用字符串 `dubois` 和 `json_object` 函数作为数组值。

```
... | eval surname = json_array("dubois", json_object("name", "patel"))
```

结果为 JSON 数组 `["dubois", {"name": "patel"}]`。

json_array_to_mv(<json_array>, <Boolean>)

此函数将适当的 JSON 数组的元素映射到多值字段。

用法

您可以将此函数和 `eval` 和 `where` 命令结合使用，并作为评估表达式的一部分和其他命令结合使用。

如果函数的 `<json_array>` 输入不是有效的 JSON 数组，则该函数不输出任何内容。

使用 `<Boolean>` 输入指定 `json_array_to_mv` 函数应对 JSON 格式的字符串保留括号。`<Boolean>` 输入默认为 `false()`。

语法	描述
<code>json_array_to_mv(<json_array>, false())</code> 或 <code>json_array_to_mv(<json_array>)</code>	默认情况下（或当您将其显式设置为 <code>false()</code> 时）， <code>json_array_to_mv</code> 函数在将数组转换为多值字段时，会从 JSON 字符串数据类型中删除括号。
<code>json_array_to_mv(<json_array>, true())</code>	当设置为 <code>true()</code> 时， <code>json_array_to_mv</code> 函数在将数组转换为多值字段时，会保留 JSON 字符串数据类型中的括号。

示例

此示例演示了如何使用 `json_array_to_mv` 函数从 JSON 数据中创建多值字段。

创建简单的多值字段

以下示例创建了一个简单数组：`["Buttercup", "Fluttershy", "Rarity"]`。然后将该数组映射到名为 `my_little_ponies` 的多值字段中，其值为 `Buttercup`、`Fluttershy` 和 `Rarity`。该函数在将数组元素转换为字段值时删除引号字符。

```
... | eval ponies = json_array("Buttercup", "Fluttershy", "Rarity"), my_sweet_ponies = json_array_to_mv(ponies)
```

如果您更改此搜索，使其具有 `my_sweet_ponies = json_array_to_mv(ponies,true())`，您将获得值为 `"Buttercup"`、`"Fluttershy"` 和 `"Rarity"` 的数组。将该函数设置为 `true` 会导致该函数在将数组元素转换为字段值时保留引号字符。

json_extend(<json>, <path_value_pairs>)

如果要一次将多个值附加到数组，请使用 `json_extract`。`json_extend` 将数组展平为其组件值，并将这些值附加到有效 JSON 文档中指定数组的末尾处。

用法

`json_extend` 函数总是至少有三个函数输入：`<json>`（有效 JSON 文档的名称，例如 JSON 对象），以及至少一个 `<path>` 和 `<value>` 对。`<value>` 必须是一个数组。当给定有效输入时，`json_extend` 总是输出一个数组。

如果 `<json>` 未引用有效的 JSON 文档（例如 JSON 对象），则该函数不输出任何内容。

`json_extend` 从左到右评估 `<path_value_pairs>`。当 `json_extend` 评估路径-值对时，它会更新 `<json>` 文档。`json_extend` 然后将根据更新的文档评估下一个路径-值对。

您可以将 `json_extend` 与 `eval` 和 `where` 命令结合使用，并作为评估表达式的一部分和其他命令结合使用。

使用 <path> 指定 JSON 文档值

每个 <path> 指定 <json> 文档中的一个数组或值。`json_extend` 函数在 <path> 指定的数组的最后一个值之后添加相应 <array> 的值。下表说明了 `json_extend` 根据 <path> 指定的内容执行的操作。

如果 <path> 指定...	... 这是 <code>json_extend</code> 对相应数组值所执行的操作
具有一个或多个值的数组。	<code>json_extend</code> 将相应的数组值添加到该数组的末尾。
空数组	<code>json_extend</code> 将相应的数组值添加到该数组中。
标量或对象值	<code>json_extend</code> 自动包装数组中的标量或对象值，并将相应的数组值添加到该数组的末尾。

`json_extend` 忽略 <path> 未标识 JSON 文档中任何有效值的路径-值对。

`json_extend` 如何在附加数组之前将数组展平

`json_extend` 函数在将数组附加到指定值时将数组展平。“展平”是指将数组分解成其组件值的行为。例如，如果 `json_extend` <path> 指向数组 `["a", "b", "c"]`，其 <value> 是数组 `["d", "e", "f"]`，则结果是 `["a", "b", "c", "d", "e", "f"]`。

将数组作为单个值附加会将 `json_extend` 与 `json_append` 区分开来，后者是将 <value> 作为单个元素进行附加的类似函数。`json_append` 以上一段为例，返回 `["a", "b", "c", ["d", "e", "f"]]`。

示例

以下示例说明了如何使用 `json_extend` 将多个值一次附加到 JSON 文档中的数组。

使用一组字符串值扩展数组

您从名为 `fakeBandsInMovies` 的对象开始，该对象包含一个名为 `fakeMovieBandList` 的数组：`["Wyld Stallyns", "Stillwater", "Spinal Tap"]`。这是您将运行的搜索，以使用电影中另外三个假乐队的名称来扩展该列表。

```
... | eval fakeBandsInMovies = json_object("fakeMovieBandList", json_array("The Blues Brothers", "Spinal Tap", "Wyld Stallyns")), updateBandList = json_extend(fakeBandsInMovies, "fakeMovieBandList", json_array("The Soggy Bottom Boys", "The Weird Sisters", "The Barden Bellas"))
```

此 `eval` 语句的输出是 `{"fakeMovieBandList": ["The Blues Brothers", "Spinal Tap", "Wyld Stallyns", "The Soggy Bottom Boys", "The Weird Sisters", "The Barden Bellas"]}`

使用对象扩展数组

此示例有一个名为 `dndChars` 的对象，其中包含一个名为 `characterClasses` 的数组。您想使用辅助数组中的对象更新此数组。您可以通过运行下面的搜索来实现该目标。

```
... | eval dndChars = json_object("characterClasses", json_array("wizard", "rogue", "barbarian")), array2 = json_array(json_object("artifact", "deck of many things")), updatedParty = json_extend(dndChars, "characterClasses", array2)
```

此 `eval` 语句的输出是 `{updatedParty = ["wizard", "rogue", "barbarian", {"artifact": "deck of many things"}]}`。请注意，当 `json_extend` 展平 `array2` 时，它将从数组中删除该对象。否则输出将是 `{updatedParty = ["wizard", "rogue", "barbarian", [{"artifact": "deck of many things"}]}}`。

`json_extract(<json>, <paths>)`

此函数从 JSON 片段和零个或多个路径返回一个值。该值以 JSON 数组或 Splunk 软件本机类型值的形式返回。

用法

转换或提取的内容取决于您指定的是 JSON 片段，还是 JSON 以及一个或多个路径。

语法	描述
<code>json_extract(<json>)</code>	将 JSON 字段转换为 Splunk 软件本机类型。例如： <ul style="list-style-type: none">将 JSON 字符串转换为字符串将 JSON 布尔值转换为布尔值将 JSON null 转换为 null

<code>json_extract(<json>, <path>)</code>	从 <code><json></code> 中提取由 <code><path></code> 指定的值，并将其转换为本机类型。如果路径指向数组，则它可能是 JSON 数组。
<code>json_extract(<json>, <path>, <path>, ...)</code>	从 <code><json></code> 提取所有路径，并将其作为 JSON 数组返回。

您可以将此函数和 `eval` 和 `where` 命令结合使用，并作为评估表达式的一部分和其他命令结合使用。

示例

以下示例使用此 JSON 对象，该对象位于某个事件的名为 `cities` 的字段中：

```
{
  "cities": [
    {
      "name": "London",
      "Bridges": [
        { "name": "Tower Bridge", "length": 801 },
        { "name": "Millennium Bridge", "length": 1066 }
      ]
    },
    {
      "name": "Venice",
      "Bridges": [
        { "name": "Rialto Bridge", "length": 157 },
        { "name": "Bridge of Sighs", "length": 36 },
        { "name": "Ponte della Paglia" }
      ]
    },
    {
      "name": "San Francisco",
      "Bridges": [
        { "name": "Golden Gate Bridge", "length": 8981 },
        { "name": "Bay Bridge", "length": 23556 }
      ]
    }
  ]
}
```

1. 在一个字段中提取整个 JSON 对象

以下示例将返回 `cities` 字段的整个 JSON 对象。`cities` 字段仅包含一个对象。该键是整个对象。这种提取可以返回任何类型的值。

```
... | eval extract_cities = json_extract(cities)
```

字段	结果
<code>extract_cities</code>	<code>{"cities": [{"name": "London", "Bridges": [{"name": "Tower Bridge", "length": 801}, {"name": "Millennium Bridge", "length": 1066}], "name": "Venice", "Bridges": [{"name": "Rialto Bridge", "length": 157}, {"name": "Bridge of Sighs", "length": 36}, {"name": "Ponte della Paglia"}]}, {"name": "San Francisco", "Bridges": [{"name": "Golden Gate Bridge", "length": 8981}, {"name": "Bay Bridge", "length": 23556}]}]}</code>

2. 提取字段中的第一个嵌套 JSON 对象

以下示例会从 JSON 对象提取有关伦敦市的信息。这种提取可以返回任何类型的值。

此示例搜索中演示的 `{<num>}` 索引仅在 `<path>` 映射到 JSON 数组时才有效。在这种情况下，`{0}` 映射到数组中的“0”项，即 London。如果示例使用 `{1}`，它将从数组中选择 Venice。

```
... | eval London=json_extract(cities,"cities{0}")
```

字段	结果
London	{"name": "London", "Bridges": [{"name": "Tower Bridge", "length": 801}, {"name": "Millennium Bridge", "length": 1066}]}]

3. 提取字段中的第三个嵌套 JSON 对象

以下示例会从 JSON 对象提取有关旧金山市的信息。这种提取可以返回任何类型的值。

```
... | eval "San_Francisco"=json_extract(cities,"cities{2}")
```

字段	结果
San_Francisco	{"name": "San Francisco", "Bridges": [{"name": "Golden Gate Bridge", "length": 8981}, {"name": "Bay Bridge", "length": 23556}]}]

4. 提取字段中每个嵌套 JSON 对象的特定键

以下示例从 JSON 对象中提取城市名称。这种提取可以返回任何类型的值。

```
... | eval my_cities=json_extract(cities,"cities{}.name")
```

字段	结果
my_cities	["London", "Venice", "San Francisco"]

5. 提取字段中每个嵌套 JSON 对象的一组特定键值对

以下示例从 JSON 对象中提取有关每个城市的每座桥梁的信息。这种提取可以返回任何类型的值。

```
... | eval Bridges=json_extract(cities,"cities{}.Bridges{}")
```

字段	结果
Bridges	[{"name": "Tower Bridge", "length": 801}, {"name": "Millennium Bridge", "length": 1066}, {"name": "Rialto Bridge", "length": 157}, {"name": "Bridge of Sighs", "length": 36}, {"name": "Ponte della Paglia"}, {"name": "Golden Gate Bridge", "length": 8981}, {"name": "Bay Bridge", "length": 23556}]

6. 提取字段中每个嵌套 JSON 对象的特定值

以下示例从 JSON 对象中提取所有城市中的桥梁的名称。这种提取可以返回任何类型的值。

```
... | eval Bridge_names=json_extract(cities,"cities{}.Bridges{}.name")
```

字段	结果
Bridge_names	["Tower Bridge", "Millennium Bridge", "Rialto Bridge", "Bridge of Sighs", "Ponte della Paglia", "Golden Gate Bridge", "Bay Bridge"]

7. 提取字段中特定嵌套 JSON 对象的特定键值对

以下示例从 JSON 对象中提取第三个城市的第一座桥梁的名称和长度。这种提取可以返回任何类型的值。

```
... | eval GG_Bridge=json_extract(cities,"cities{2}.Bridges{0}")
```

字段	结果
GG_Bridge	{"name": "Golden Gate Bridge", "length": 8981}

8. 提取字段中特定嵌套 JSON 对象的特定值

以下示例从 JSON 对象中提取第三个城市的第一座桥梁的长度。这种提取可以返回任何类型的值。

```
... | eval GG_Bridge_length=json_extract(cities,"cities[2].Bridges[0].length")
```

字段	结果
GG_Bridge_length	8981

json_keys(<json>)

返回 JSON 对象中键值对中的键。键将以 JSON 数组的形式返回。

用法

您可以将此函数和 eval 和 where 命令结合使用，并作为评估表达式的一部分和其他命令结合使用。

json_keys 函数不能用于 JSON 数组。

示例

从一个 JSON 对象返回键列表

考虑以下 JSON 对象，它位于 bridges 字段中：

bridges
{"name": "Clifton Suspension Bridge", "length": 1352, "city": "Bristol", "country": "England"}

此示例从 bridges 字段中的 JSON 对象中提取键：

```
... | eval bridge_keys = json_keys(bridges)
```

以下为该搜索的结果：

bridge_keys
["name", "length", "city", "country"]

从多个 JSON 对象返回键列表

考虑以下 JSON 对象，它们位于 bridges 字段的不同行中：

bridges
{"name": "Clifton Suspension Bridge", "length": 1352, "city": "Bristol", "country": "England"}
{"name": "Rialto Bridge", "length": 157, "city": "Venice", "region": "Veneto", "country": "Italy"}
{"name": "Helix Bridge", "length": 918, "city": "Singapore", "country": "Singapore"}
{"name": "Tilikum Crossing", "length": 1700, "city": "Portland", "state": "Oregon", "country": "United States"}

此示例从 bridges 字段中的 JSON 对象中提取键：

```
... | eval bridge_keys = json_keys(bridges)
```

以下为该搜索的结果：

bridge_keys
["name", "length", "city", "country"]
["name", "length", "city", "region", "country"]
["name", "length", "city", "country"]
["name", "length", "city", "state", "country"]

json_set(<json>, <path_value_pairs>)

使用提供的值插入或覆盖 JSON 节点的值，然后返回更新的 JSON 对象。

用法

您可以将此函数和 eval 和 where 命令结合使用，并作为评估表达式的一部分和其他命令结合使用。

- 如果路径包含键列表，则如果键不存在，则会创建链中的所有键。
- 如果 JSON 对象与路径之间不匹配，则会跳过更新，并且不会产生错误。例如，对于对象 {"a": "b"}, json_set(.., "a.c", "d") 不会产生任何结果，因为 "a" 具有字符串值，而 "a.c" 则表示嵌套对象。
- 如果该值已经存在并且是一种匹配的非值类型，则 json_set 函数默认会覆盖该值。不会强制执行值类型匹配。例如，您可以用字符串、布尔值、空值等覆盖数字。

示例

以下示例使用此 JSON 对象，该对象位于某个事件的名为 games 的字段中：

```
{  
  "category": {  
    "boardgames": {  
      "cooperative": [  
        {  
          "name": "Pandemic"  
        },  
        {  
          "name": "Forbidden Island"  
        },  
        {  
          "name": "Castle Panic"  
        }  
      ]  
    }  
  }  
}
```

1. 覆盖现有 JSON 数组中的值

以下示例将覆盖 JSON 对象的路径 [category.boardgames.cooperative] 中的 "Castle Panic" 值。该值将替换为 "name": "Sherlock Holmes: Consulting Detective"。结果放入一个名为 my_games 的新字段中。

位置计数从 0 开始。第三个位置是 2，这就是为什么该示例在路径中指定 {2} 的原因。

```
... | eval my_games = json_set(games, "category.boardgames.cooperative{2}", "name": "Sherlock Holmes: Consulting Detective")
```

以下为该搜索的结果：

字段	结果
my_games	{"category": {"boardgames": {"cooperative": ["name": "Pandemic", "name": "Forbidden Island", "name": "Sherlock Holmes: Consulting Detective"]}}}

2. 在现有 JSON 对象中插入值列表

以下示例将流行游戏列表 ["name": "Settlers of Catan", "name": "Terraforming Mars", "name": "Ticket to Ride"] 插入 JSON 对象的路径 [category.boardgames.competitive] 中。

由于键 competitive 在路径中不存在，因此会创建该键。json_array 函数用于将值列表附加到 boardgames JSON 对象。

```
... | eval my_games = json_set(games, "category.boardgames.competitive", json_array(json_object("name", "Settlers of Catan"), json_object("name", "Terraforming Mars"), json_object("name", "Ticket to Ride")))
```

以下为该搜索的结果：

字段	结果
my_games	{"category": {"boardgames": {"cooperative": [{"name": "Pandemic", "name": "Forbidden Island", "name": "Sherlock Holmes: Consulting Detective"}, {"name": "Settlers of Catan", "name": "Terraforming Mars", "name": "Ticket to Ride"}]}}}

JSON 对象现在会如下所示：

```
{
  "category": {
    "boardgames": {
      "cooperative": [
        {
          "name": "Pandemic"
        },
        {
          "name": "Forbidden Island"
        },
        {
          "name": "Castle Panic"
        }
      ]
    },
    "competitive": [
      {
        "name": "Settlers of Catan"
      },
      {
        "name": "Terraforming Mars"
      },
      {
        "name": "Ticket to Ride"
      }
    ]
  }
}
```

3. 在现有 JSON 对象中插入一组键值对

以下示例会插入一组键值对，它们使用布尔值指定游戏是否可用。这些键值对将插入 JSON 对象的路径 [category.boardgames.competitive] 中。json_array 函数用于将键值对列表附加到 boardgames JSON 对象。

```
... | eval my_games = json_set(games, "category.boardgames.competitive{} .available", true())
```

以下为该搜索的结果：

字段	结果
my_games	{"category": {"boardgames": {"cooperative": [{"name": "Pandemic", "name": "Forbidden Island", "name": "Sherlock Holmes: Consulting Detective"}, {"name": "Settlers of Catan", "name": "Terraforming Mars", "name": "Ticket to Ride"}], "competitive": [{"name": "Settlers of Catan", "available": true}, {"name": "Terraforming Mars", "available": true}, {"name": "Ticket to Ride", "available": true}]}}}

JSON 对象现在会如下所示：

```
{
  "category": {
    "boardgames": {
      "cooperative": [
        {
          "name": "Pandemic"
        },
        {
          "name": "Forbidden Island"
        },
        {
          "name": "Castle Panic"
        }
      ]
    },
    "competitive": [
      {
        "name": "Settlers of Catan", "available": true
      },
      {
        "name": "Terraforming Mars", "available": true
      },
      {
        "name": "Ticket to Ride", "available": true
      }
    ]
  }
}
```

```

        {
          "name": "Castle Panic"
        }
      ],
    },
    "competitive": [
      {
        "name": "Settlers of Catan",
        "available": true
      },
      {
        "name": "Terraforming Mars",
        "available": true
      },
      {
        "name": "Ticket to Ride",
        "available": true
      }
    ]
  }
}

```

如果 Settlers of Catan 游戏缺货，则可以用值 `false()` 覆盖 `available` 键的值。

例如：

```
... | eval my_games = json_set(games,"category.boardgames.competitive{0}.available", false())
```

以下为该搜索的结果：

字段	结果
my_games	{"category": {"boardgames": {"cooperative": [{"name": "Pandemic", "name": "Forbidden Island", "name": "Sherlock Holmes: Consulting Detective"}, {"competitive": [{"name": "Settlers of Catan", "available": false, "name": "Terraforming Mars", "available": true, "name": "Ticket to Ride", "available": true}]}]}}

JSON 对象现在会如下所示：

```
{
  "category": {
    "boardgames": {
      "cooperative": [
        {
          "name": "Pandemic"
        },
        {
          "name": "Forbidden Island"
        },
        {
          "name": "Castle Panic"
        }
      ]
    },
    "competitive": [
      {
        "name": "Settlers of Catan",
        "available": false
      },
      {
        "name": "Terraforming Mars",
        "available": true
      },
      {
        "name": "Ticket to Ride",
        "available": true
      }
    ]
  }
}
```

```
}
```

json_valid(<json>)

评估 JSON 片段是否使用有效的 JSON 语法并返回 TRUE 或 FALSE。

用法

您可以将此函数和 eval 和 where 命令结合使用，并作为评估表达式的一部分和其他命令结合使用。

示例

验证 JSON 对象

以下示例用于验证 firstnames 字段中的 JSON 对象 { "names": ["maria", "arun"] }。

由于字段不能包含布尔值，因此 if 函数与 json_valid 函数一起使用，将布尔值的等效字符串值放入 isValid 字段中。

```
... | eval IsValid = if(json_valid(firstnames), "true", "false")
```

mv_to_json_array(<field>, <Boolean>)

此函数将多值字段的元素映射到 JSON 数组。

用法

您可以将此函数和 eval 和 where 命令结合使用，并作为评估表达式的一部分和其他命令结合使用。

由于 JSON 数组的元素可以具有多种数据类型（例如字符串、数值、布尔值和 null），因此 mv_to_json_array 函数允许您指定如何将多值字段的内容映射到 JSON 数组。您可以将字段值作为字符串数据类型简单地写入数组，也可以让函数推断不同的 JSON 数据类型。

使用 <Boolean> 输入指定 mv_to_json_array 函数在将字段值转换为数组元素时应尝试推断 JSON 数据类型。<Boolean> 输入默认为 false。

语法	描述
<code>mv_to_json_array(<field>, false()) 或 mv_to_json_array(<field>)</code>	默认情况下（或当您将其显式设置为 false() 时），mv_to_json_array 函数会将多值字段中的所有值作为字符串数据类型映射到 JSON 数组，无论它们是数值、字符串、布尔值还是任何其他 JSON 数据类型。mv_to_json_array 函数有效地用逗号分割多值字段，并将每个引号括起来的值作为字符串数据类型的元素写入数组。
<code>mv_to_json_array(<field>, true())</code>	当您将 mv_to_json_array 函数设置为 true() 时，该函数会从它传输到 JSON 数组的每个值中删除一组括号字符。如果函数无法将产生的数组元素识别为正确的 JSON 数据类型（例如字符串、数值、布尔值或 null），则该函数会将元素转换为 null 数据类型。

示例

此示例显示 json_array_to_mv 函数如何在生成 JSON 数组时验证 JSON。

生成仅包含有效 JSON 字符串的简单多值字段

此搜索会创建一个名为 ponies 的多值字段。然后将该多值字段转换为名为 my_little_ponies 的 JSON 数组。产生的数组如下所示：[\"Buttercup\", \"Fluttershy\", \"Rarity\", \"true\", \"null\"]。当设置为 false（或未提供布尔值时），mv_to_json_array 函数会将字段值转换为数组元素而不更改它们。

```
... | eval ponies = mvappend(\"Buttercup\", \"Fluttershy\", \"Rarity\", \"true\", \"null\"), my_sweet_ponies =  
mv_to_json_array(ponies, false())
```

当您使用 my_sweet_ponies = mv_to_json_array(ponies, true()) 运行此搜索时，您会得到以下数组：[\"Buttercup\", \"Fluttershy\", \"Rarity\", true, null]。当设置为 true 时，mv_to_json_array 函数在将字段值转换为数组元素时，会从字段值中删除额外的引号和反斜杠转义字符。

另请参阅

函数

评估函数快速参考

数学函数

以下列表包含了您可以用于执行数学计算的函数。

- 有关在函数和嵌套函数中使用字符串和数字字段的信息，请参阅“评估函数”。
- 有关可与这些函数一起使用的数学运算符的列表，请参阅 eval 命令“用法”一节中的“运算符”。

abs(X)

描述

此函数获取一个数字 X 并返回其绝对值。

用法

您可以将此函数和 eval、fieldformat 及 where 命令结合使用，并作为 eval 表达式的一部分。

基本示例

以下示例新建名为 absnum 的字段，其值为数字字段 number 的绝对值。

```
... | eval absnum=abs(number)
```

ceiling(X) 或 ceil(X)

描述

此函数将数字 X 向上舍入到下一个最大整数。

用法

您可以将此函数和 eval、fieldformat 及 where 命令结合使用，并作为 eval 表达式的一部分。

您可以使用缩写 ceil(X) 而不是函数的全称。

基本示例

以下示例将返回 n=2。

```
... | eval n=ceil(1.9)
```

exact(X)

描述

此函数在格式化输出中以更高的精度呈现数字 eval 计算的结果。

用法

您可以将此函数和 eval、fieldformat 及 where 命令结合使用，并作为 eval 表达式的一部分。

基本示例

```
... | eval n=exact(3.14 * num)
```

exp(X)

描述

此函数获取一个数字 X 并返回指数函数 e^X 。

用法

您可以将此函数和 eval、fieldformat 及 where 命令结合使用，并作为 eval 表达式的一部分。

基本示例

以下示例将返回 $y=e^3$ 。

```
... | eval y=exp(3)
```

floor(X)

描述

此函数将数字 X 向下舍入到最近的整数。

用法

您可以将此函数和 eval、fieldformat 及 where 命令结合使用，并作为 eval 表达式的一部分。

基本示例

以下示例将返回 1。

```
... | eval n=floor(1.9)
```

ln(X)

描述

此函数获取数字 X 并返回其自然对数。

用法

您可以将此函数和 eval、fieldformat 及 where 命令结合使用，并作为 eval 表达式的一部分。

基本示例

以下示例将返回字节值的自然对数。

```
... | eval lnBytes=ln(bytes)
```

log(X, Y)

描述

此函数获取一个或两个数字参数，并返回以第二个参数 Y 为底的第一个参数 X 的对数。如果省略了第二个参数 Y，则此函数将评估为以 10 为底的数字 X 的对数。

用法

您可以将此函数和 eval、fieldformat 及 where 命令结合使用，并作为 eval 表达式的一部分。

基本示例

```
... | eval num=log(number,2)
```

pi()

描述

此函数不获取任何参数，并返回 11 位精度的常量值 *pi*。

用法

您可以将此函数和 eval、fieldformat 及 where 命令结合使用，并作为 eval 表达式的一部分。

基本示例

以下示例计算圆面积，即 pi() 乘以半径的 2 次幂。

```
... | eval area_circle=pi()*pow(radius,2)
```

pow(X, Y)

描述

此函数获取两个数字参数 X 和 Y，并返回 X^Y (X 的 Y 次幂)。

用法

您可以将此函数和 eval、fieldformat 及 where 命令结合使用，并作为 eval 表达式的一部分。

基本示例

以下示例计算圆面积，即 pi() 乘以半径的 2 次幂。

```
... | eval area_circle=pi()*pow(radius,2)
```

round(X, Y)

描述

此函数采用一个或两个数字参数 X 和 Y，并返回按 Y 所指定的小数位数进行向上舍入的 X。默认值是向上舍入为整数。

用法

您可以将此函数和 eval、fieldformat 及 where 命令结合使用，并作为 eval 表达式的一部分。

基本示例

以下示例将返回 n=4。

```
... | eval n=round(3.5)
```

以下示例将返回 n=2.56。

```
... | eval n=round(2.555, 2)
```

下面的示例使用 -1 指定四舍五入到十位的精度。

```
... | eval n=round(155, -1)
```

该搜索返回 n=150。

sigfig(X)

描述

此函数获取一个数字参数 X，并将该数字四舍五入到适合的有效数字数。

用法

您可以将此函数和 eval、fieldformat 及 where 命令结合使用，并作为 eval 表达式的一部分。

sigfig 计算是基于生成数字的计算类型。

- 对于乘法和除法，结果应具有所有操作数的最小有效数字数。
- 对于加法和减法，结果应该具有与所有操作数的最不精确数字相同的小数位数。

例如，数字 123.0 和 4.567 小数位数不同，精确度不同。第一个数字没那么精确，因为只有 1 个小数位。第二个数字就比较精确，因为有 3 个小数位。

如果计算是 $123.0 + 4.567 = 127.567$ ，那么 sigfig 函数将返回小数位最少的数字。在此示例中，将返回只有一个小数位的数字。由于最后一个有效数字右侧的数字大于 5，因此返回的结果为 127.6

基本示例

示例 1：下面的示例说明了 sigfig 函数的工作方式。计算 $1.00*1111$ 将返回值 n=1111，但是以下使用 sigfig 函数的搜索将返回 n=1110。

```
... | eval n=sigfig(1.00*1111)
```

在此示例中，1.00 有 3 个有效数字，1111 有 4 个有效数字。在此示例中，所有操作数的有效数字的最小数是 3。如果使用 sigfig 函数，最终结果会四舍五入到 3 位数，因此会返回 n=1110 而不是 1111。

示例 2：在某些情况下，在返回的计算结果中，小数点最右边的精度可能会有所不同。例如，以下搜索计算 100 个值的平均值：

```
| makeresults count=100 | eval test=3.99 | stats avg(test)
```

计算结果为：

avg(test)
3.990000000000055

当计数更改为 10000 时，结果会有所不同：

```
| makeresults count=10000 | eval test=3.99 | stats avg(test)
```

计算结果为：

avg(test)
3.99000000000215

发生这种情况的原因是，数字被视为双精度浮点数。

要缓解此问题，可以使用 sigfig 函数指定要返回的有效数字的位数。

但是，首先您需要对搜索的 stats 命令部分进行更改。您需要更改字段 avg(test) 的名称并移除括号。例如，stats avg(test) AS test。sigfig 函数需要 X 的数字或字段名。sigfig 函数不接受看起来像另一个函数的字段名，在本例中即为 avg。

要指定希望返回的小数位数，请将字段名乘以 1，然后用零来指定小数位数。例如，如果希望返回 4 个小数位，则将字段名称乘以 1.0000。如果希望返回 2 个小数位，则乘以 1.00，如以下示例所示：

```
| makeresults count=10000 | eval test=3.99 | stats avg(test) AS test | eval new_test=sigfig(test*1.00)
```

计算结果为：

test
3.99

sqrt(X)

描述

此函数获取一个数字参数 X 并返回其平方根。

用法

您可以将此函数和 eval、fieldformat 及 where 命令结合使用，并作为 eval 表达式的一部分。

基本示例

以下示例将返回 3。

```
... | eval n=sqrt(9)
```

多值 eval 函数

以下列表包含了您可以用予多值字段或返回多值字段的函数。

您还可以在多值字段上使用统计 eval 函数 max 和 min。请参阅“统计 eval 函数”。

有关在函数和嵌套函数中使用字符串和数字字段的信息，请参阅“评估函数”。

commands(X)

描述

此函数获取搜索字符串或包含搜索字符串 X 的字段，并返回包含 X 中所用命令列表的多值字段。

用法

通常不建议用此函数，除非分析 audit.log 事件。

您可以将此函数和 eval、fieldformat 及 where 命令结合使用，并作为 eval 表达式的一部分。

基本示例

以下示例返回包含 'search'、'stats' 和 'sort' 的多值字段 X。

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

mvappend(X, ...)

描述

此函数获取任意数量的参数并返回所有值的多值结果。参数可以是字符串、多值字段或单值字段。

用法

您可以将此函数和 eval、fieldformat 及 where 命令结合使用，并作为 eval 表达式的一部分。

基本示例

本示例说明如何附加两个值：localhost 是文字字符串值，而 srcip 是字段名称。

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

本示例说明如何使用嵌套 `mvappend` 函数。

- 内层 `mvappend` 函数包含两个值： `localhost` 是文字字符串值，而 `srcip` 是字段名称。
- 外层 `mvappend` 函数包含三个值： 内层 `mvappend` 函数， `destip` 是字段名称，而 `192.168.1.1` 是文字 IP 地址。

```
... | eval ipaddresses=mvappend(mvappend("localhost", srcip), destip, "192.168.1.1")
```

结果放在名为 `ipaddresses` 新字段中，该字段包含数组 `["localhost", <values_in_srcip>, <values_in_destip>, "192.168.1.1"]`。

mvcount (MVFIELD)

描述

此函数使用一个字段，并针对每个结果返回该字段中值的数量。如果字段是多值字段，则返回该字段中值的数量。如果字段包含单值，此函数返回 1。如果字段没有值，此函数返回空值。

用法

您可以将此函数和 `eval`、`fieldformat` 及 `where` 命令结合使用，并作为 `eval` 表达式的一部分。

基本示例

```
... | eval n=mvcount(multifield)
```

延伸示例

在以下示例中，`mvcount()` 函数返回 `To`、`From` 和 `Cc` 字段中电子邮件地址的数量。

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

此搜索取 `To` 字段中的值，并使用拆分函数在 @ 符号处拆分电子邮件地址。拆分函数也出于相同目的用于 `cc` 字段。

如果只有一个电子邮件地址存在于 `From` 字段中，正如您所期望的那样，`mvcount(From)` 返回 1。如果没有 `cc` 地址，事件可能不存在 `cc` 字段。在这种情况下 `mvcount(cc)` 返回空值。

mvdedup (X)

描述

此函数获取多值字段 `X`，并返回一个已删除重复值的多值字段。

用法

您可以将此函数和 `eval`、`fieldformat` 及 `where` 命令结合使用，并作为 `eval` 表达式的一部分。

基本示例

```
... | eval s=mvdedup(mvfield)
```

mvfilter (X)

描述

此函数基于任意布尔表达式 `X` 过滤多值字段。布尔表达式 `X` 一次只能引用一个字段。

用法

此函数还会返回字段 `x` 的 `NULL` 值。如果您不想要空值，请使用以下任意表达式：

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

您可以将此函数和 eval、fieldformat 及 where 命令结合使用，并作为 eval 表达式的一部分。

基本示例

以下示例将返回 email 字段以 .net 或 .org 结尾的所有值。

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

mvfind(MVFIELD, "REGEX")

描述

此函数将尝试查找多值字段 MVFIELD 中与 "REGEX" 中正则表达式匹配的值。如果找到匹配项，则返回第一个匹配值的索引（以零开头）。如果未找到匹配值，则返回空值。

用法

您可以将此函数和 eval、fieldformat 及 where 命令结合使用，并作为 eval 表达式的一部分。

基本示例

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

mvindex(MVFIELD, STARTINDEX, ENDINDEX)

描述

此函数获取两个或三个参数，并返回使用所提供索引值的多值字段的子集。字段 MVFIELD 和数字 STARTINDEX 是必填项。包含数字 ENDINDEX 且是可选的。

用法

您可以将此函数和 eval、fieldformat 及 where 命令结合使用，并作为 eval 表达式的一部分。

索引从零开始。如果您在多值字段中有 5 个值，第一个值的索引为 0。第二个值的索引为 1。

参数 STARTINDEX 和 ENDINDEX 均可以是负数，其中 -1 代表最后一个元素。

如果未指定 ENDINDEX，则函数只返回 STARTINDEX 处的值。

如果索引超出范围或无效，则结果为空值。

基本示例

由于索引从零开始，因此以下示例将在 "multifield" 中返回第三个值（如果值存在）。

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

延伸示例

以下搜索最多显示 <field> 中的最后 10 个值。

STARTINDEX 是一个范围，从最后一个值 -1 开始。范围是最后的 10 个值 -1-10。ENDINDEX 是 -1，返回字段中的最后一个值。

- 如果多值字段有 20 个值，则仅返回最后 10 个值。
- 如果多值字段有 3 个值，则仅返回 3 个值。

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

mvjoin(MVFIELD, STR)

描述

此函数获取两个参数，多值字段 (MVFIELD) 和字符串分隔符 (STR)。此函数使用 STR 的值作为分割符连接 MVFIELD 中的各个值。

用法

您可以将此函数和 eval、fieldformat 及 where 命令结合使用，并作为 eval 表达式的一部分。

基本示例

您有包含值 "1" "2" "3" "4" "5" 的名为 "base" 的多值字段。值以空格隔开。您想要新建以 OR 为分隔符的单值字段。例如 "1 OR 2 OR 3 OR 4 OR 5"。

以下搜索用值新建 base 字段。然后使用 mvjoin 函数的结果新建 joined 字段。

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

以下示例将使用分号作为分隔符来连接 "foo" 的各个值：

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

mvmap (X, Y)

描述

这个函数迭代一个多值字段 (X) 的值，对每个值执行一个操作 (Y)，然后返回一个包含结果列表的多值字段。

- X 是引用了单一字段的多值表达式。
- Y 是结果表达式。

用法

您可以将此函数和 eval、fieldformat 及 where 命令结合使用，并作为 eval 表达式的一部分。

基本示例

以下示例将 foo 中的每个值乘以 10。

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

以下示例将 foo 中的每个值乘以 bar，其中 bar 是单值字段。

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

以下示例将 foo 中的第二个和第三个值乘以 bar，其中 bar 是单值字段。

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

mvrange (X, Y, Z)

描述

此函数将为一组数字新建一个多值字段。此函数最多可以包含三个参数：一个起始数 X、一个结尾数 Y（从字段中排除）和一个可选递增量 Z。如果增量是时间范围，如 7d，起始和结尾数将以 UNIX 时间呈现。

用法

您可以将此函数和 eval、fieldformat 及 where 命令结合使用，并作为 eval 表达式的一部分。

基本示例

以下示例将返回包含值 1、3、5、7、9 的多值字段。

```
... | eval mv=mvrange(1,11,2)
```

以下示例将 1/1/2018 的 UNIX 时间戳作为开始日期，4/19/2018 的 UNIX 时间戳作为结束日期并使用 7 天的时间增量。

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

此示例返回带有 UNIX 时间戳的多值字段。统计选项卡中显示的结果如下所示：

_time	mv
2018/4/10 12:31:03	1514834731 1515439531 1516044331 1516649131 1517253931 1517858731 1518463531 1519068331 1519673131 1520277931 1520879131 1521483931 1522088731 1522693531 1523298331 1523903131

mvsort(X)

描述

此函数使用多值字段 X，并返回一个多值字段，该字段的数值按字典顺序排序。

用法

您可以将此函数和 eval、fieldformat 及 where 命令结合使用，并作为 eval 表达式的一部分。

基于用于编码计算机内存中的项目的值按字典顺序对这些项目进行排序。在 Splunk 软件中，几乎都是使用 UTF-8 进行编码，这是 ASCII 的超集。

- 数字排在字母前面。根据第一位数字对数字进行排序。例如数字 10、9、70、100，按照字典顺序排序为 10、100、70、9。
- 大写字母排在小写字母前面。
- 符号的排序标准不固定。有些符号排在数字值前面。有些符号排在字母前面或后面。

基本示例

```
... | eval s=mvsort(mvfield)
```

mvzip(X, Y, "Z")

描述

此函数获取两个多值字段 X 和 Y，然后通过依次将字段 X 的第一个值与字段 Y 的第一个值接合，将 X 的第二个值与 Y 的第二个值接合，依此类推，从而合并这两个字段。第三个参数 Z 可选，用于指定分隔符以连接两个值。默认分隔符为逗号。

用法

这类似于 Python zip 命令。

您可以将此函数和 eval、fieldformat 及 where 命令结合使用，并作为 eval 表达式的一部分。

基本示例

```
... | eval nserver=mvzip(hosts,ports)
```

延伸示例

您可将若干 `mvzip` 函数嵌套在一起，以从三个独立的字段新建单一多值字段 `three_fields`。管道符（|）用作字段值之间的分隔符。

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

（感谢 Splunk 用户 cmerriman 提供此示例。）

split(X, "Y")

描述

此函数获取两个参数，字段 X 和分隔符 Y。它将在分隔符 Y 处分割 X 的值并以多值字段形式返回 X。

用法

您可以将此函数和 `eval`、`fieldformat` 及 `where` 命令结合使用，并作为 `eval` 表达式的一部分。

Splunk 软件包含一组多值函数。请参阅多值 `eval` 函数和多值 `stats` 和 `chart` 函数。

基本示例

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

另请参阅

请参阅以下多值命令：

`makemv`, `mvcombine`, `mvexpand`, `nomv`

统计 eval 函数

以下列表包含了您可用于计算统计信息的评估函数。

有关在函数和嵌套函数中使用字符串和数字字段的信息，请参阅“评估函数”。

除了这些函数外，还有一整套统计函数可与 `stats`、`chart` 及相关命令结合使用。

max(X, ...)

描述

此函数获取任意数量的数字或字符串参数，并返回最大值。字符串大于数字。

用法

您可以将此函数和 `eval`、`fieldformat` 及 `where` 命令结合使用，并作为 `eval` 表达式的一部分。

基本示例

以下示例会返回 “foo” 或 `name` 字段中的值。Splunk 搜索使用字典顺序，其中数字排在字母之前。如果 `name` 字段中的值为 “baz”，则返回 “foo”。如果 `name` 字段中的值为 “zaz”，则返回 “zaz”。

```
... | eval n=max(1, 3, 6, 7, "foo", name)
```

以下示例将返回多值字段中的最大值。

此搜索会创建一个名为 `n` 的字段，该字段只有单个值，即一系列数字。`makemv` 命令用于将单个值转换为多个值，每个值在结果中出现在它自己的行上。创建的另一个新字段名为 `maxn`，它采用 `n` 中的值并返回最大值 6。

```
| makeresults | eval n = "1 3 5 6 4 2" | makemv n | eval maxn = max(n)
```

结果如下所示：

_time	maxn	N
2021-01-29 10:42:37	6	1 3 5 6 4 2

`min(X, ...)`

描述

此函数获取任意数量的数字或字符串参数，并返回最小值。字符串大于数字。

用法

您可以将此函数和 eval、fieldformat 及 where 命令结合使用，并作为 eval 表达式的一部分。

基本示例

以下示例会返回 3 或 size 字段中的值。Splunk 搜索使用字典顺序，其中数字排在字母之前。如果 size 字段中的值为 9，则返回 3。如果 size 字段中的值为 1，则返回 1。

```
... | eval n=min(3, 6, 7, "maria", size)
```

以下示例将返回多值字段中的最小值。

此搜索会创建一个名为 n 的字段，该字段只有单个值，即一系列数字。makemv 命令用于将单个值转换为多个值，每个值在结果中出现在它自己的行上。创建的另一个新字段名为 minn，它采用 n 中的值并返回最小值 2。

```
| makeresults | eval n = "3 5 6 4 7 2" | makemv n | eval minn = min(n)
```

结果如下所示：

_time	minn	N
2021-01-29 10:42:37	2	3 5 6 4 7 2

`random()`

描述

此函数不获取任何参数，并返回一个 0 到 $2^{31}-1$ 之间的伪随机整数。

用法

您可以将此函数和 eval、fieldformat 及 where 命令结合使用，并作为 eval 表达式的一部分。

基本示例

以下示例会返回随机证书，例如 0...2147483647。

```
... | eval n=random()
```

以下示例返回指定范围内的随机数。在此示例中，随机数介于 1 到 100,000 之间。

```
... | eval n=(random() % 100000) + 1
```

本例以随机数为例，使用模数学运算符（%）用随机数除以 100000。这可以确保返回的随机数不会大于 100000。相除后得出的数字加 1 确保该数字至少大于或等于 1。

文本函数

以下列表包含了您可以和字符串值结合使用的函数。

有关在函数和嵌套函数中使用字符串和数字字段的信息，请参阅“评估函数”。

len(X)

描述

此函数会返回字符串 X 的字符长度。

用法

您可以将此函数和 eval、fieldformat 及 where 命令结合使用，并作为 eval 表达式的一部分。

多值字段不支持此函数。

基本示例

假设您有类似如下所示的一组结果：

_time	names
2020/1/9 16:35:14	buttercup
2020/1/9 16:35:14	rarity
2020/1/9 16:35:14	tenderhoof
2020/1/9 16:35:14	dash
2020/1/9 16:35:14	mistmane

您可以使用 len 函数确定 names 字段中的值的长度：

```
... | eval length=len(names)
```

结果显示了 names 字段中的值的字符长度计数：

_time	length	names
2020/1/9 16:35:14	9	buttercup
2020/1/9 16:35:14	6	rarity
2020/1/9 16:35:14	10	tenderhoof
2020/1/9 16:35:14	4	dash
2020/1/9 16:35:14	8	mistmane

lower(X)

描述

此函数获取一个字符串参数并返回小写字符串。

用法

您可以将此函数和 eval、fieldformat 及 where 命令结合使用，并作为 eval 表达式的一部分。

您可以在多值字段上使用此函数。

基本示例

以下示例会以小写形式返回 username 字段提供的值。

```
... | eval username=lower(username)
```

ltrim(X, Y)

描述

此函数获取一个或两个参数 X 和 Y，返回从左侧裁剪掉 Y 中字符的 X。如果未指定 Y，则删除空格和制表符。

用法

您可以将此函数和 eval、fieldformat 及 where 命令结合使用，并作为 eval 表达式的一部分。

多值字段不支持此函数。

基本示例

以下示例从左开始修剪字符串前导空格和出现的字母 Z。返回的值是 x="abcZZ"。

```
... | eval x=ltrim(" ZZZabcZZ ", " Z")
```

replace(X, Y, Z)

描述

此函数针对字符串 X 中出现的每个正则表达式 Y，返回一个由替代字符串 Z 所形成的字符串。第三个参数 Z 也可以引用正则表达式中匹配的组。

用法

您可以将此函数和 eval、fieldformat 及 where 命令结合使用，并作为 eval 表达式的一部分。

多值字段不支持此函数。

要替换反斜杠 (\) 字符，您必须转义反斜杠两次。因为 eval 表达式中使用 replace 函数。在将正则表达式传递给 PCRE 之前，eval 表达式执行一级转义。PCRE 自行转义。请参阅“SPL 和正则表达式”。

基本示例

以下示例将返回月份和日期交换之后的日期。如果输入是 1/14/2017，那么返回值可能是 14/1/2017。

```
... | eval n=replace(date, "^(\d{1,2})/(\d{1,2})/", "\2/\1/")
```

rtrim(X, Y)

描述

此函数获取一个或两个参数 X 和 Y，返回从右侧裁剪掉 Y 中字符的 X。如果未指定 Y，则删除空格和制表符。

用法

您可以将此函数和 eval、fieldformat 及 where 命令结合使用，并作为 eval 表达式的一部分。

多值字段不支持此函数。

基本示例

以下示例将返回 `n= ZZZabc`。

```
... | eval n=rtrim(" ZZZabcZZ ", " Z")
```

spath(X, Y)

描述

此函数获取两个参数：一个输入数据来源字段 `X` 和一个 `spath` 表达式 `Y`，这是您想要从 `X` 中提取的值的位置路径（采用 XML 或 JSON 格式）。

用法

您可以将此函数和 `eval`、`fieldformat` 及 `where` 命令结合使用，并作为 `eval` 表达式的一部分。

多值字段不支持此函数。

如果 `Y` 是文本字符串，则需要用括号括起来，即 `spath(X,"Y")`。如果 `Y` 是字段名称（所含值为位置路径），则不需要加引号。此函数将产生一个多值字段。阅读更多有关 `spath` 命令的信息。

基本示例

以下示例将返回 `locDesc` 元素的值。

```
... | eval locDesc=spath(_raw, "vendorProductSet.product.desc.locDesc")
```

以下示例将返回 `twitter` 事件的 `hashtags`。

```
index=twitter | eval output=spath(_raw, "entities.hashtags")
```

substr(X, Y, Z)

描述

此函数获取两个或三个参数。所需参数为字符串 `X` 和数字 `Y`。数字 `Z` 是可选的。此函数将返回 `X` 的子字符串，该字符串从 `Y` 指定的索引开始，并带有 `Z` 指定的字符数。如果未指定 `Z`，函数将返回字符串的剩余部分。

用法

索引遵循 SQLite 语义，从 1 开始。可使用负索引来表示从字符串的末尾开始。

您可以将此函数和 `eval`、`fieldformat` 及 `where` 命令结合使用，并作为 `eval` 表达式的一部分。

多值字段不支持此函数。

基本示例

以下示例会将 `"str"` 和 `"ing"` 连接在一起，从而返回 `"string"`：

```
... | eval n=substr("string", 1, 3) + substr("string", -3)
```

trim(X, Y)

描述

此函数获取一个或两个参数 `X` 和 `Y`，并返回从左右两侧截取了 `Y` 中字符的 `X`。如果未指定 `Y`，则删除空格和制表符。

用法

您可以将此函数和 eval、fieldformat 及 where 命令结合使用，并作为 eval 表达式的一部分。
多值字段不支持此函数。

基本示例

以下示例将返回 "abc"。

```
... | eval n=trim(" ZZZabcZZ ", " Z")
```

upper(X)

描述

此函数获取一个字符串参数并返回大写字符串。

用法

您可以将此函数和 eval、fieldformat 及 where 命令结合使用，并作为 eval 表达式的一部分。
您可以在多值字段上使用此函数。

基本示例

以下示例会以大写形式返回 username 字段提供的值。

```
... | eval n=upper(username)
```

urldecode(X)

描述

此函数获取 URL 字符串参数 X 并返回已取消转义或解码的 URL 字符串。

用法

您可以将此函数和 eval、fieldformat 及 where 命令结合使用，并作为 eval 表达式的一部分。
多值字段不支持此函数。

基本示例

以下示例将返回 "http://www.splunk.com/download?r=header"。

```
... | eval n=urldecode("http%3A%2F%2Fwww.splunk.com%2Fdownload%3Fr%3Dheader")
```

三角函数和双曲函数

以下列表包含了您可以用于计算三角和双曲值的函数。

有关在函数和嵌套函数中使用字符串和数字字段的信息，请参阅“评估函数”。

acos(X)

描述

此函数计算 X 的反余弦，弧度区间为 [0, pi]。

用法

您可以将此函数和 eval、fieldformat 及 where 命令结合使用，并作为 eval 表达式的一部分。

基本示例

以下示例计算 0 的反余弦。

```
... | eval n=acos(0)
```

以下示例计算 180 除以 π ，然后乘以 0 的反余弦结果。

```
... | eval degrees=acos(0)*180/pi()
```

acosh(X)

描述

此函数计算 X 的反双曲余弦，以弧度表示。

用法

您可以将此函数和 eval、fieldformat 及 where 命令结合使用，并作为 eval 表达式的一部分。

基本示例

```
... | eval n=acosh(2)
```

asin(X)

描述

此函数计算 X 的反正弦，弧度区间为 $[-\pi/2, +\pi/2]$ 。

用法

您可以将此函数和 eval、fieldformat 及 where 命令结合使用，并作为 eval 表达式的一部分。

基本示例

以下示例计算 1 的反正弦。

```
... | eval n=asin(1)
```

以下示例计算 180 除以 π ，然后乘以 1 的反正弦结果。

```
... | eval degrees=asin(1)*180/pi()
```

asinh(X)

描述

此函数计算 X 的反双曲正弦，以弧度表示。

用法

您可以将此函数和 eval、fieldformat 及 where 命令结合使用，并作为 eval 表达式的一部分。

基本示例

```
... | eval n=asinh(1)
```

atan(X)

描述

此函数计算 X 的反正切，弧度区间为 [-pi/2, +pi/2]。

用法

您可以将此函数和 eval、fieldformat 及 where 命令结合使用，并作为 eval 表达式的一部分。

基本示例

```
... | eval n=atan(0.50)
```

atan2(Y, X)

描述

此函数计算 Y 的反正切，弧度区间为 [-pi, +pi]。

Y 代表坐标的 Y 轴值。X 代表坐标的 X 轴值。

为计算此值，此函数会同时考虑两个参数的符号，以判断象限位置。

用法

您可以将此函数和 eval、fieldformat 及 where 命令结合使用，并作为 eval 表达式的一部分。

基本示例

```
... | eval n=atan2(0.50, 0.75)
```

atanh(X)

描述

此函数计算 X 的反双曲正切，以弧度表示。

用法

您可以将此函数和 eval、fieldformat 及 where 命令结合使用，并作为 eval 表达式的一部分。

基本示例

```
... | eval n=atanh(0.500)
```

cos(X)

描述

此函数计算 X 弧度角的余弦。

用法

您可以将此函数和 eval、fieldformat 及 where 命令结合使用，并作为 eval 表达式的一部分。

基本示例

以下示例计算 -1 的余弦。

```
... | eval n=cos(-1)
```

以下示例计算 π 的余弦。

```
... | eval n=cos(pi())
```

cosh (X)

描述

此函数计算 X 弧度的反双曲余弦。

用法

您可以将此函数和 eval、fieldformat 及 where 命令结合使用，并作为 eval 表达式的一部分。

基本示例

```
... | eval n=cosh(1)
```

hypot (X, Y)

描述

此函数计算边为 X 和 Y 的直角三角形的斜边。

此函数返回 X 和 Y 的平方和的平方根，符合毕氏定理。

用法

您可以将此函数和 eval、fieldformat 及 where 命令结合使用，并作为 eval 表达式的一部分。

基本示例

```
... | eval n=hypot(3,4)
```

sin (X)

描述

此函数计算 X 的正弦。

用法

您可以将此函数和 eval、fieldformat 及 where 命令结合使用，并作为 eval 表达式的一部分。

基本示例

以下示例计算 1 的正弦。

```
... | eval n=sin(1)
```

以下搜索计算 180 除以 π 的结果的正弦，然后乘以 90 。

```
... | eval n=sin(90 * pi()/180)
```

sinh(X)

描述

此函数计算 X 的双曲正弦。

用法

您可以将此函数和 eval、fieldformat 及 where 命令结合使用，并作为 eval 表达式的一部分。

基本示例

```
... | eval n=sinh(1)
```

tan(X)

描述

此函数计算 X 的正切。

用法

您可以将此函数和 eval、fieldformat 及 where 命令结合使用，并作为 eval 表达式的一部分。

基本示例

```
... | eval n=tan(1)
```

tanh(X)

描述

此函数计算 X 的双曲正切。

用法

您可以将此函数和 eval、fieldformat 及 where 命令结合使用，并作为 eval 表达式的一部分。

基本示例

```
... | eval n=tanh(1)
```

统计和图表函数

统计和图表函数

与 `chart`、`stats` 和 `timechart` 命令结合使用统计和图表函数。

支持相关命令

此函数还可与相关统计和图表命令结合使用。以下表格列出了统计和图表函数支持的命令和也可使用这些函数的相关命令。

命令	支持的相关命令
chart	<ul style="list-style-type: none"><code>sichart</code>
stats	<ul style="list-style-type: none"><code>eventstats</code><code>streamstats</code><code>geostats</code><code>sistats</code>对于 <code>tstats</code> 和 <code>mstats</code> 命令，请参阅文档中每个命令支持的函数的列表。
timechart	<ul style="list-style-type: none"><code>sitimechart</code>

可用于新建迷你图的函数在各函数的文档中注明。迷你图是只能应用于 `chart` 和 `stats` 命令的函数，让您可以调用其他函数。更多信息，请参阅《搜索手册》中的“为搜索结果添加迷你图”。

如何处理字段值

大多数统计和图表函数期望字段值为数字。所有值都将作为数字处理，任何非数字值都将被忽略。

以下函数将字段值作为文字字符串值处理，即使该值是数字。

<ul style="list-style-type: none"><code>count</code><code>distinct_count</code><code>earliest</code>	<ul style="list-style-type: none"><code>estdc</code><code>estdc_error</code><code>first</code>	<ul style="list-style-type: none"><code>latest</code><code>last</code><code>list</code>	<ul style="list-style-type: none"><code>max</code><code>min</code><code>mode</code><code>values</code>
--	--	---	---

例如，您使用 `distinct_count` 函数，并且该字段包含诸如 "1"、"1.0" 和 "01" 的值。每个值都被视为非重复的字符串值。

唯一的例外是 `max` 和 `min` 函数。这些函数尽可能将值作为数字处理。例如：值 "1"、"1.0" 和 "01" 将作为相同的数字值进行处理。

支持的函数和语法

您可通过两种方式查看支持的统计和图表函数的信息：

- 按类别排列的函数列表
- 按字母顺序排列的函数列表

按类别排列的函数列表

下表是支持的统计和图表函数的快速参考信息，按类别排列。此表格提供每个函数的简要描述。使用表格中的链接以了解有关每个函数的更多信息，并查看示例。

函数类型	支持的函数和语法	描述
	<code>avg(X)</code>	返回字段 X 中的值的平均值。
	<code>count(X)</code>	返回包含任意值（非空）的您指定的字段的出现次数。您还可以结合使用 <code>count</code> 函数与 <code>eval</code> 命令，统计字段中特定值的出现次数。例如： <code>count eval(field_name="value")</code> 。

	<code>distinct_count(X)</code>	返回字段 X 中非重复值的计数。
	<code>estdc(X)</code>	返回字段 X 中非重复值的估计计数。
	<code>estdc_error(X)</code>	返回字段 X 中非重复值的估计计数的理论误差。此误差表示 <code>absolute_value(estimate_distinct_count - real_distinct_count)/real_distinct_count</code> 之比。
	<code>max(X)</code>	返回字段 X 的最大值。如果 X 的值不是数字，则按字典顺序查找最大值。如果可能，此函数将字段值作为数字进行处理，否则，就将字段值作为字符串进行处理。
	<code>mean(X)</code>	返回字段 X 的算术平均值。
	<code>median(X)</code>	返回字段 X 最中间的值。
聚合函数	<code>min(X)</code>	返回字段 X 的最小值。如果 X 的值不是数字，则按字典顺序查找最小值。
	<code>mode(X)</code>	返回字段 X 最常出现的值。
	<code>percentile<X>(Y)</code>	返回数字字段 Y 的第 X 个百分位数。X 的有效值是从 1 到 99 的整数。 其他百分位数函数是 <code>upperperc<X>(Y)</code> 和 <code>exactperc<X>(Y)</code> 。
	<code>range(X)</code>	返回字段 X 的最大值与最小值之差（前提是 X 的值为数字）。
	<code>stdev(X)</code>	返回字段 X 的样本标准偏差。
	<code>stdevp(X)</code>	返回字段 X 的总体标准偏差。
	<code>sum(X)</code>	返回字段 X 的值的总和。
	<code>sumsq(X)</code>	返回字段 X 的值的平方和。
	<code>var(X)</code>	返回字段 X 的样本方差。
	<code>varp(X)</code>	返回字段 X 的总体方差。
事件顺序函数	<code>first(X)</code>	返回字段 X 出现的第一个值。通常，出现的第一个字段值是该字段的最新实例（相对于 stats 命令中事件的输入顺序）。
	<code>last(X)</code>	返回字段 X 最后出现一个值。最后出现的字段值通常为该字段的最旧实例（相对于事件输入到 stats 命令中的顺序）。
多值统计和 chart 函数	<code>list(X)</code>	以多值条目的形式返回字段 X 最多 100 个值的列表。值的顺序反映输入事件的顺序。
	<code>values(X)</code>	以多值条目的形式返回字段 X 所有唯一值的列表。各值按字典顺序排列。
时间函数	<code>earliest(X)</code>	返回时间上最早出现的（最老）字段 X 的值。
	<code>earliest_time(X)</code>	返回最早（最晚）发生的字段值的 UNIX 时间。与 <code>earliest(x)</code> 、 <code>latest(x)</code> 和 <code>latest_time(x)</code> 结合使用以计算累计计数器的增长率。
	<code>latest(X)</code>	返回时间上最晚出现的（最近）字段 X 的值。
	<code>latest_time(X)</code>	返回最早（最近）发生的字段值的 UNIX 时间。与 <code>earliest(x)</code> 、 <code>earliest_time(x)</code> 和 <code>latest(x)</code> 结合使用以计算累计计数器的增长率。
	<code>per_day(X)</code>	返回每天字段 X 或 eval 表达式 X 的值。
	<code>per_hour(X)</code>	返回每小时字段 X 或 eval 表达式 X 的值。
	<code>per_minute(X)</code>	返回每分钟字段 X 或 eval 表达式 X 的值。
	<code>per_second(X)</code>	返回每秒字段 X 或 eval 表达式 X 的值。
	<code>rate(X)</code>	返回字段值的每秒速率变化。代表 <code>(latest(X) - earliest(X)) / (latest_time(X) - earliest_time(X))</code> 要求 <code>earliest(X)</code> 和 <code>latest(X)</code> 字段值是数字， <code>earliest_time(X)</code> 和 <code>latest_time(X)</code> 值为不同值。

rate_avg(X)	返回与指定的累计计数器指标相关联的时间序列的平均速率。
rate_sum(X)	返回与指定的累计计数器指标相关联的时间序列的合计速率。

按字母顺序排列的函数列表

下表是支持的统计和图表函数的快速参考信息，按字母顺序排列。此表格提供每个函数的简要描述。使用表格中的链接以了解有关每个函数的更多信息，并查看示例。

支持的函数和语法	描述	函数类型
avg(X)	返回字段 X 中的值的平均值。	聚合函数
count(X)	返回包含任意值（非空）的您指定的字段的出现次数。您还可以结合使用 count 函数与 eval 命令，统计字段中特定值的出现次数。例如：count eval(field_name="value")。	聚合函数
distinct_count(X)	返回字段 X 中非重复值的计数。	聚合函数
earliest(X)	返回时间上最早出现的（最老）字段 X 的值。	时间函数
earliest_time(X)	返回最早（最晚）发生的字段值的 UNIX 时间。与 earliest(x)、latest(x) 和 latest_time(x) 结合使用以计算累计计数器的增长率。	时间函数
estdc(X)	返回字段 X 中非重复值的估计计数。	聚合函数
estdc_error(X)	返回字段 X 中非重复值的估计计数的理论误差。此误差表示 absolute_value(estimate_distinct_count - real_distinct_count)/real_distinct_count 之比。	聚合函数
first(X)	返回字段 X 出现的第一个值。通常，出现的第一个字段值是该字段的最新实例（相对于 stats 命令中事件的输入顺序）。	事件顺序函数
last(X)	返回字段 X 最后出现一个值。最后出现的字段值通常为该字段的最旧实例（相对于事件输入到 stats 命令中的顺序）。	事件顺序函数
latest(X)	返回时间上最晚出现的（最近）字段 X 的值。	时间函数
latest_time(X)	返回最早（最近）发生的字段值的 UNIX 时间。与 earliest(x)、earliest_time(x) 和 latest(x) 结合使用以计算累计计数器的增长率。	时间函数
list(X)	以多值条目的形式返回字段 X 最多 100 个值的列表。值的顺序反映输入事件的顺序。	多值统计和 chart 函数
max(X)	返回字段 X 的最大值。如果 X 的值不是数字，则按字典顺序查找最大值。如果可能，此函数将字段值作为数字进行处理，否则，就将字段值作为字符串进行处理。	聚合函数
mean(X)	返回字段 X 的算术平均值。	聚合函数
median(X)	返回字段 X 最中间的值。	聚合函数
min(X)	返回字段 X 的最小值。如果 X 的值不是数字，则按字典顺序查找最小值。	聚合函数
mode(X)	返回字段 X 最常出现的值。	聚合函数
percentile<X>(Y)	返回数字字段 Y 的第 X 个百分位数。X 的有效值是从 1 到 99 的整数。 其他百分位数函数是 upperperc<X>(Y) 和 exactperc<X>(Y)。	聚合函数
per_day(X)	返回每天字段 X 或 eval 表达式 X 的值。	时间函数
per_hour(X)	返回每小时字段 X 或 eval 表达式 X 的值。	时间函数

<code>per_minute(X)</code>	返回每分钟字段 X 或 eval 表达式 X 的值。	时间函数
<code>per_second(X)</code>	返回每秒字段 X 或 eval 表达式 X 的值。	时间函数
<code>range(X)</code>	返回字段 X 的最大值与最小值之差（前提是 X 的值为数字）。	聚合函数
<code>rate(X)</code>	返回字段值的每秒速率变化。代表 $(\text{latest}(X) - \text{earliest}(X)) / (\text{latest_time}(X) - \text{earliest_time}(X))$ 要求 earliest(X) 和 latest(X) 字段值是数字, earliest_time(X) 和 latest_time(X) 值为不同值。	时间函数
<code>rate_avg(X)</code>	返回与指定的累计计数器指标相关联的时间序列的平均速率。	时间函数
<code>rate_sum(X)</code>	返回与指定的累计计数器指标相关联的时间序列的合计速率。	时间函数
<code>stdev(X)</code>	返回字段 X 的样本标准偏差。	聚合函数
<code>stdevp(X)</code>	返回字段 X 的总体标准偏差。	聚合函数
<code>sum(X)</code>	返回字段 X 的值的总和。	聚合函数
<code>sumsq(X)</code>	返回字段 X 的值的平方和。	聚合函数
<code>values(X)</code>	以多值条目的形式返回字段 X 所有唯一值的列表。各值按字典顺序排列。	多值统计和 chart 函数
<code>var(X)</code>	返回字段 X 的样本方差。	聚合函数
<code>varp(X)</code>	返回字段 X 的总体方差。	聚合函数

另请参阅

命令

`chart`
`geostats`
`eventstats`
`stats`
`streamstats`
`timechart`

函数

评估函数

问答

有什么问题吗？请访问 [Splunk Answers](#)，并搜索某个特定函数或命令。

聚合函数

聚合函数汇总各事件的值以新建有意义的单个值。常用聚合函数包括 `Average`、`Count`、`Minimum`、`Maximum`、`Standard Deviation`、`Sum` 和 `Variance`。

大多数聚合函数与数字字段一起使用。但是，有一些函数可以用于字母字符串字段或数字字段。函数描述指示哪些函数可以使字母字符串。

如要查看概览，请参阅统计和图表函数。

`avg(X)`

描述

返回字段 X 的值的平均值。

用法

您可以将此函数与 `chart`、`mstats`、`stats`、`timechart` 和 `tstats` 命令结合使用，还可以和 `sparkline()` 图表结合使用。

有关您可以和此函数结合使用的相关统计和图表命令列表，请参阅“统计和图表函数”。

基本示例

示例 1

以下示例将返回每个非重复 "host" 的平均 "size"。

```
... | stats avg(size) BY host
```

示例 2

以下示例将返回每 5 分钟跨度内每个 "host" 的平均 "thruput"。

```
... | bin _time span=5m | stats avg(thruput) BY _time host
```

示例 3

以下示例针对每个非重复 "host" 和 "user" 对的平均 "size" 与最大 "delay" 的比绘制图表。

```
... | chart eval(avg(size)/max(delay)) AS ratio BY host user
```

示例 4

以下示例显示按处理器新建的 cpu_seconds 平均值（舍入到 2 个小数位）的时间图表。

```
... | timechart eval(round(avg(cpu_seconds),2)) BY processor
```

延伸示例

示例 1

在某些情况下，在返回的计算结果中，小数点最右边的精度可能会有所不同。例如，以下搜索计算 100 个值的平均值：

```
| makeresults count=100 | eval test=3.99 | stats avg(test)
```

计算结果为：

avg(test)
3.990000000000055

当计数更改为 10000 时，结果会有所不同：

```
| makeresults count=10000 | eval test=3.99 | stats avg(test)
```

计算结果为：

avg(test)
3.990000000000215

发生这种情况的原因是，数字被视为双精度浮点数。

要缓解此问题，可以使用 `sigfig` 函数指定要返回的有效数字的位数。

但是，首先您需要对搜索的 `stats` 命令部分进行更改。您需要更改字段 `avg(test)` 的名称并移除括号。例如，`stats avg(test) AS test`。`sigfig` 函数需要 X 的数字或字段名。`sigfig` 函数不接受看起来像另一个函数的字段名，在本例中即为 `avg`。

要指定希望返回的小数位数，请将字段名乘以 1，然后用零来指定小数位数。例如，如果希望返回 4 个小数位，则将字段名称乘以 1.0000。如果希望返回 2 个小数位，则乘以 1.00，如以下示例所示：

```
| makeresults count=10000 | eval test=3.99 | stats avg(test) AS test | eval new_test=sigfig(test*1.00)
```

计算结果：

test
3.99

示例 2

基于交易的持续时间显示交易中的平均事件数的图表。

此示例使用搜索教程中的示例数据。要在您自己的 Splunk 实例中尝试此示例，您必须下载示例数据并遵循操作以将教程数据导入 Splunk。当运行搜索时，请使用时间范围所有时间。

1. 运行以下搜索以新建一个基于交易的持续时间显示交易中的平均事件数的图表。

```
sourcetype=access_* status=200 action=purchase | transaction clientip maxspan=30m | chart avg(eventcount) by duration span=log2
```

transaction 命令向结果 duration 和 eventcount 添加两个字段。eventcount 字段追踪单个交易中的事件数量。

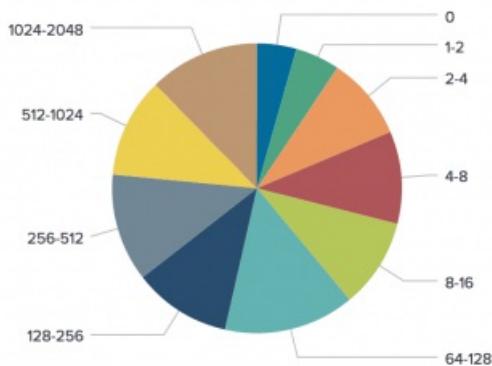
在此搜索中，交易传递到 chart 命令。avg() 函数用于计算每个期间的平均事件数。因为持续时间以秒为单位，预期会有许多值，搜索使用 span 通过基数为 2 的算法将数据箱持续时间分为数据箱。

2. 使用字段格式选项以启用数字格式化。

The screenshot shows the Splunk search interface with a search bar containing the command: sourcetype=access_* status=200 action=purchase | transaction clientip maxspan=30m | chart avg(eventcount) by duration span=log2. Below the search bar, there are tabs for '事件' (Events), '模式' (Mode), '统计信息 (438)' (Statistics (438)), and '可视化' (Visualization). The '可视化' tab is selected. In the center, there is a chart visualization showing data points for 'duration' and 'avg(eventcount)'. A context menu is open over the 'duration' field, with the '字段格式选项' (Field Format Options) option highlighted. A sub-menu for '数字格式' (Number Format) is open, showing two options: '已禁用' (Disabled) and '已启用' (Enabled), with '已禁用' currently selected.

3. 单击可视化选项卡并将显示形式更改为饼图。

Pie Chart 格式 棚架



饼图的每个扇形代表事件交易的持续时间。您可以悬停在扇形区以查看平均值。

count(X) 或 c(X)

描述

返回字段 X 的出现次数。为指示要匹配的特定字段值，X 可以采用此格式：Eval(field="value")。将字段值作为字符串进行处理。要使用这一功能，您可以指定 count(X) 或缩写 c(X)。

用法

您可以将 count(X) 函数与 chart、mstats、stats、timechart 和 tstats 命令结合使用，还可以和 sparkline() 图表结合使用。

基本示例

以下示例返回事件的计数结果，其中字段 status 的值为 "404"。

```
... | stats count(eval(status="404")) AS count_status BY sourcetype
```

此示例将 eval 表达式和 count 函数结合使用。请参阅 stats 函数中“使用 eval 表达式”。

以下示例将搜索结果分到 10 个数据箱，并返回每个数据箱中原始事件的计数。

```
... | bin size bins=10 | stats count(_raw) BY size
```

以下示例生成迷你图以统计使用 _raw 字段的事件的数量。

```
... sparkline(count)
```

以下示例生成迷你图以统计有 user 字段的事件的数量。

```
... sparkline(count(user))
```

以下示例使用 timechart 命令统计事件计数，其中 action 字段包含值 purchase。

```
sourcetype=access_* | timechart count(eval(action="purchase")) BY productName usenull=f useother=f
```

延伸示例

统计每个震级范围出现的地震次数

此搜索使用从 USGS 地震网站下载的近期地震数据。该数据是一个逗号分隔的 ASCII 文本文件，其中包含每次记录的地震的震级 (mag)、坐标 (经度、

纬度)、区域(地点)等。

您可以从 USGS 地震源下载当前 CSV 文件，然后将文件上载到 Splunk 实例。此示例使用过去 30 天的所有地震数据。

1. 运行以下搜索计算发生在每个震级范围内的地震数量。该数据集由 30 天之内的事件组成。

```
source=all_month.csv | chart count AS "Number of Earthquakes" BY mag span=1 | rename mag AS "Magnitude Range"
```

- o 此搜索使用 `span=1` 定义震级字段 `mag` 的每个范围。
- o 然后，使用 `rename` 命令将字段重命名为 "Magnitude Range" (震级范围)。

统计选项卡中显示的结果如下所示：

震级范围	地震数量
-1-0	18
0-1	2088
1-2	3005
2-3	1026
3-4	194
4-5	452
5-6	109
6-7	11
7-8	3

为每个 Web 服务器统计不同页面请求数

此示例使用搜索教程中的示例数据，但应该可以与任意格式的 Apache Web 访问日志配合使用。要在您自己的 Splunk 实例中尝试此示例，您必须下载示例数据并遵循操作以将教程数据导入 Splunk。当运行搜索时，请使用时间范围所有时间。

1. 运行以下搜索以使用 `chart` 命令确定每个 Web 服务器上发生的不同页面请求 (GET 和 POST) 的数量。

```
sourcetype=access_* | chart count(eval(method="GET")) AS GET, count(eval(method="POST")) AS POST BY host
```

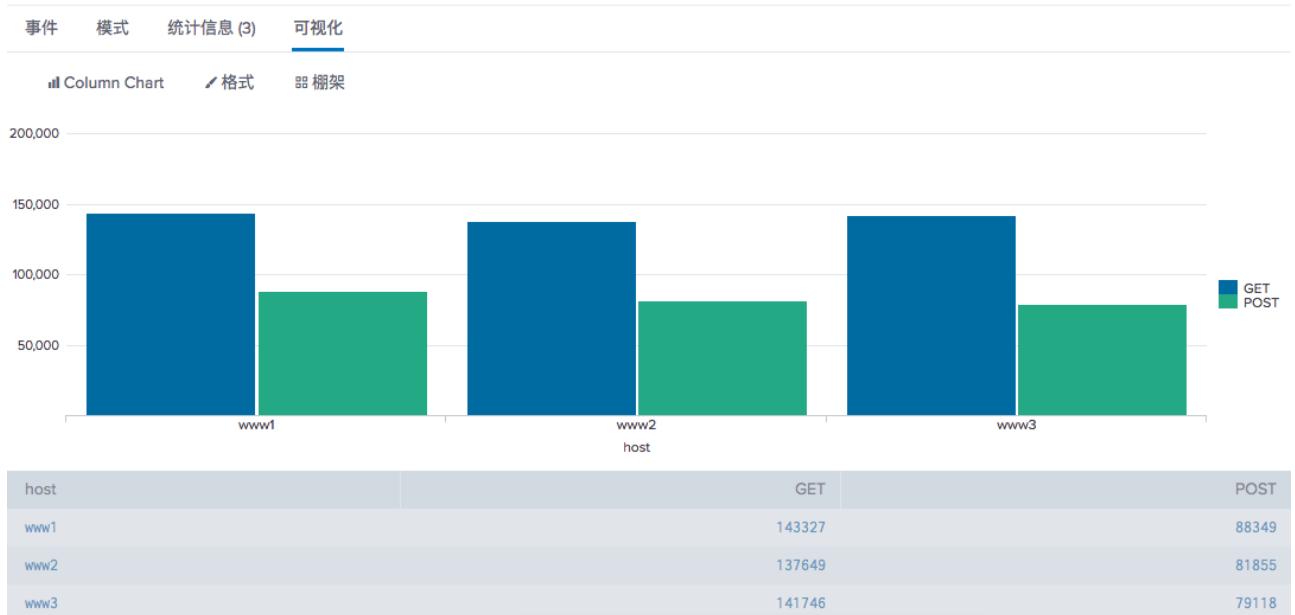
本示例使用 `eval` 表达式来指定 `stats` 命令要统计的不同字段值。第一个子句使用 `count()` 函数来统计包含 `method` 字段值 GET 的 Web 访问事件。然后，使用 `AS` 关键字，代表这些结果的字段重命名为 GET。

第二个子句对 POST 事件执行相同的操作。然后两种类型的事件计数都由 web 服务器分隔，使用有 `host` 字段的 `BY` 子句。

统计选项卡中显示的结果如下所示：

host	GET	POST
www1	8431	5197
www2	8097	4815
www3	8338	4654

2. 单击可视化选项卡。如需要，将结果格式设置为柱状图。此图表根据 `host` 值显示每个事件类型 (GET 或 POST) 的事件总数。



`distinct_count(X)` 或 `dc(X)`

描述

返回字段 X 的非重复值的计数。此函数将字段值作为字符串处理。要使用这一功能，您可以指定 `distinct_count(X)` 或缩写 `dc(X)`。

用法

您可以将此函数与 `chart`、`mstats`、`stats`、`timechart` 和 `tstats` 命令结合使用，还可以和 `sparkline()` 图表结合使用。

基本示例

以下示例将删除具有相同 "host" 值的重复结果，并返回剩余结果的总计数。

```
... | stats dc(host)
```

以下示例将为设备的不同计数生成迷你图并重命名为 "numdevices" 字段。

```
...sparkline(dc(device)) AS numdevices
```

以下示例将对每种来源类型的非重复来源进行计数，并且每五分钟将计数放入数据桶一次。

```
...sparkline(dc(source),5m) BY sourcetype
```

延伸示例

此示例使用搜索教程中的示例数据。要在您自己的 Splunk 实例中尝试此示例，您必须下载示例数据并遵循操作以将教程数据导入 Splunk。当运行搜索时，请使用时间范围昨天。

1. 运行以下搜索对昨天在 Buttercup Games 线上商店购买东西的不同客户进行计数。搜索按照客户购买的产品类型（配饰、T 恤和游戏类型）组织计数。

```
sourcetype=access_* action=purchase | stats dc(clientip) BY categoryId
```

- 本示例首先搜索购物事件 `action=purchase`。
- 然后将这些结果通过管道符传递给 `stats` 命令，再使用 `dc()` 函数统计在店内购物的不同用户的数量。
- 运用 `BY` 子句根据不同的产品类别 (`categoryId`) 细分用户数量。

统计选项卡中显示的结果如下所示：

categoryid	dc(clientip)
ACCESSORIES	37
ARCADE	58
NULL	8
SHOOTER	31
SIMULATION	34
SPORTS	13
STRATEGY	74
TEE	38

estdc(X)

描述

返回字段 X 的非重复值的估计计数。此函数将字段值作为字符串处理。将字符串值 1.0 和 1 视为非重复值并分开计数。

用法

您可以将此函数与 chart、stats、timechart 和 tstats 命令结合使用。

默认情况下，如果搜索返回的非重复值的实际数量小于 1000，则 Splunk 软件不会估算该搜索的非重复值计数。而是使用实际的非重复值计数。此阈值由 limits.conf 中的 approx_dc_threshold 设置决定。

基本示例

以下示例将删除具有相同 "host" 值的重复结果，并返回剩余结果的估计总计数。

```
... | stats estdc(host)
```

结果形式如下所示：

estdc(host)
6

以下示例将为 devices 字段的估算非重复计数生成迷你图，并将结果字段重命名为 "numdevices"。

```
...sparkline(estdc(device)) AS numdevices
```

以下示例估计每个来源类型的数据来源的非重复计数。在迷你图中显示每五分钟跨度内的结果。

```
...sparkline(estdc(source),5m) BY sourcetype
```

estdc_error(X)

描述

返回字段 X 非重复值的估计计数的理论误差。此误差表示 $\text{absolute_value}(\text{estimate_distinct_count} - \text{real_distinct_count})/\text{real_distinct_count}$ 之比。该函数将字段值作为字符串进行处理。

用法

您可以将此函数与 chart、stats 和 timechart 命令结合使用。

基本示例

以下示例确定了 "host" 值的估计非重复计数的错误比率。

```
... | stats estdc_error(host)  
exactperc<X>(Y)
```

描述

返回数字字段 Y 的百分位数值。

用法

您可以将此函数与 chart、stats、timechart 和 tstats 命令结合使用，还可以和 sparkline() 图表结合使用。

exactperc 函数提供准确值，但对于高基数组来说会非常占用资源。exactperc 函数在搜索头中会消耗大量内存，这可能会影响搜索完成所需的时间。

示例

请参阅 perc<X>(Y) 函数。

max(X)

描述

返回字段 X 的最大值。如果 X 的值不是数字，则按字典顺序查找最大值。

尽可能将字段值作为数字进行处理，否则，就将字段值作为字符串进行处理。

用法

您可以将此函数与 chart、mstats、stats 和 timechart 命令结合使用，还可以和 sparkline() 图表结合使用。

基于用于编码计算机内存中的项目的值按字典顺序对这些项目进行排序。在 Splunk 软件中，几乎都是使用 UTF-8 进行编码，这是 ASCII 的超集。

- 数字排在字母前面。根据第一位数字对数字进行排序。例如数字 10、9、70、100，按照字典顺序排序为 10、100、70、9。
- 大写字母排在小写字母前面。
- 符号的排序标准不固定。有些符号排在数字值前面。有些符号排在字母前面或后面。

基本示例

以下示例将返回 size 字段的最大值。

```
... | stats max(size)
```

延伸示例

计算一个地区地震震级的统计汇总统计数据

此搜索使用从 USGS 地震网站下载的近期地震数据。该数据是一个逗号分隔的 ASCII 文本文件，其中包含每次记录的地震的震级 (mag)、坐标 (经度、纬度)、区域 (地点) 等。

您可以从 [USGS 地震源](#) 下载当前 CSV 文件，然后将文件上载到 Splunk 实例。此示例使用过去 30 天的所有地震数据。

1. 搜索加利福尼亚及周边地区的地震。计算已记录的地震数量。使用统计函数计算最小、最大范围（最小和最大之间的差异）以及最近地震的平均震级。按震级类型列出值。

```
source=all_month.csv place=*California* | stats count, max(mag), min(mag), range(mag), avg(mag) BY magType
```

统计选项卡中显示的结果如下所示：

magType	count	max (mag)	min (mag)	range (mag)	avg (mag)
H	123	2.8	0.0	2.8	0.549593
MbLg	1	0	0	0	0.000000
Md	1565	3.2	0.1	3.1	1.056486
Me	2	2.0	1.6	.04	1.800000
Ml	1202	4.3	-0.4	4.7	1.226622
Mw	6	4.9	3.0	1.9	3.650000
ml	10	1.56	0.19	1.37	0.934000

mean(X)

描述

返回字段 X 的算术平均值。

mean 值应与使用 avg() 函数计算的值完全相同。

用法

您可以将此函数与 chart、mstats、stats 和 timechart 命令结合使用，还可以和 sparkline() 图表结合使用。

基本示例

以下示例将返回 "kbps" 值的平均值：

```
... | stats mean(kbps)
```

延伸示例

此搜索使用从 USGS 地震网站下载的近期地震数据。该数据是一个逗号分隔的 ASCII 文本文件，其中包含每次记录的地震的震级 (mag)、坐标（经度、纬度）、区域（地点）等。

您可以从 **USGS 地震源** 下载当前 CSV 文件，然后将文件上载到 Splunk 实例。此示例使用过去 30 天的所有地震数据。

- 运行以下搜索以找出按震级类型分类的最近地震震级的平均值、标准偏差和方差。

```
source=usgs place=*California* | stats count mean(mag), stdev(mag), var(mag) BY magType
```

统计选项卡中显示的结果如下所示：

magType	count	mean (mag)	std (mag)	var (mag)
H	123	0.549593	0.356985	0.127438
MbLg	1	0.000000	0.000000	0.000000
Md	1565	1.056486	0.580042	0.336449
Me	2	1.800000	0.346410	0.120000
Ml	1202	1.226622	0.629664	0.396476
Mw	6	3.650000	0.716240	0.513000
ml	10	0.934000	0.560401	0.314049

median(X)

描述

返回字段 X 最中间的值。

用法

您可以将此函数与 chart、mstats、stats 和 timechart 命令结合使用。

如果您的事件为偶数个， 默认情况下， 中值计算约等于两个值中更高的那个。

此函数本质上是不确定的。这意味着使用此函数对相同数据进行的后续搜索， 可能会在其结果中包含细微的差异。

如果您需要更精确和一致的结果， 您可以使用 exactperc50() 代替。但对于高基数字段来说， exactperc<X>(Y) 函数非常占用资源。请参阅 perc<X>(Y)。

基本示例

考虑以下值，这些值是对昨天在 Buttercup Games 线上商店购买了东西的不同客户数量进行的计数。这些值按照客户购买的产品类型（配饰、T 恤和游戏类型）进行组织。

categoryId	计数
ACCESSORIES	37
ARCADE	58
NULL	8
SIMULATION	34
SPORTS	13
STRATEGY	74
TEE	38

当对列表进行排序后，中间值（最中间的值）为 37。

categoryId	计数
NULL	8
SPORTS	13
SIMULATION	34
ACCESSORIES	37
TEE	38
ARCADE	58
STRATEGY	74

min(X)

描述

返回字段 X 的最小值。如果 X 的值不是数字，则按字典顺序查找最小值。

如果可能，此函数将字段值作为数字进行处理，否则，就将字段值作为字符串进行处理。

用法

您可以将此函数与 chart、mstats、stats 和 timechart 命令结合使用。

基于用于编码计算机内存中的项目的值按字典顺序对这些项目进行排序。在 Splunk 软件中，几乎都是使用 UTF-8 进行编码，这是 ASCII 的超集。

- 数字排在字母前面。根据第一位数字对数字进行排序。例如数字 10、9、70、100，按照字典顺序排序为 10、100、70、9。
- 大写字母排在小写字母前面。
- 符号的排序标准不固定。有些符号排在数字值前面。有些符号排在字母前面或后面。

基本示例

以下示例返回 _internal 索引中 HotBucketRoller 组件的最小大小和最大大小。

```
index=_internal component=HotBucketRoller | stats min(size), max(size)
```

以下示例返回处理器的列表并计算最小 cpu_seconds 和最大 cpu_seconds。

```
index=_internal | chart min(cpu_seconds), max(cpu_seconds) BY processor
```

延伸示例

请参阅 `max()` 函数更为详细的示例。该示例包括 `min()` 函数。

mode(X)

描述

返回字段 X 最常出现的值。

将字段值作为字符串进行处理。

用法

您可以将此函数与 `chart`、`stats` 和 `timechart` 命令结合使用。

基本示例

`mode` 返回最常出现的值。考虑以下数据：

firstname	surname	age
Claudia	Garcia	32
David	Mayer	45
Alex	Garcia	29
Wei	Zhang	45
Javier	Garcia	37

如果在 `age` 字段中搜索 `mode`，则会返回值 45。

```
...| stats mode(age)
```

您还可以对包含字符串值的字段使用 `mode`。如果在 `surname` 字段中搜索 `mode`，则会返回值 `Garcia`。

```
...| stats mode(surname)
```

以下是另一组示例数据：

_time	host	sourcetype
04-06-2020 17:06:23.000 PM	www1	access_combined
04-06-2020 10:34:19.000 AM	www1	access_combined
04-03-2020 13:52:18.000 PM	www2	access_combined
04-02-2020 07:39:59.000 AM	www3	access_combined

04-01-2020 19:35:58.000 PM | www1 | access_combined

如果运行一个搜索来针对 `host` 字段寻找 `mode`, 则会返回值 `www1`, 因为这个值是 `host` 字段中最常出现的值。例如:

```
... |stats mode(host)
```

结果形式如下所示:

mode (host)
www1

perc<X>(Y)

描述

百分位函数返回数字字段 `Y` 的第 `X` 个百分位值。您可以将其视为对最高 `X%` 起始位置的估计。例如, 第 95 个百分位表示字段 `Y` 中 95% 的值都低于估计值, 即字段 `Y` 中 5% 的值高于估计值。

`X` 的有效值是 0 到 100 之间的浮点数, 例如 99.95。

可用的百分位函数有三种:

函数	描述
perc<X>(Y) 或缩写 p<X>(Y) •	使用 perc 函数计算合适的阈值, 使得字段 <code>Y</code> 中值的百分之 <code>X</code> 低于阈值。perc 函数返回一个数字, 该数字表示所请求的百分位数的近似值的下限。
upperperc<X>(Y)	如果值的数量大于 1000, upperperc 函数针对所请求的百分位数给出近似值上限。否则, upperperc 函数返回的百分位数会与 perc 函数返回的相同。
exactperc<X>(Y)	exactperc 函数提供准确值, 但对于高基数组来说会非常占用资源。exactperc 函数会消耗大量内存, 这可能会影响搜索完成所需的时间。

百分位数函数将字段值作为字符串进行处理。

perc 和 upperperc 函数本质上是不确定的, 这意味着使用这些函数对相同数据进行的后续搜索, 可能在其结果中包含细微的差异。

如果您需要精确且一致的结果, 可以使用 exactperc<X>(Y) 代替。

用法

您可以将此函数与 chart、mstats、stats、timechart 和 tstats 命令结合使用。

Splunk 和 Excel 百分位数算法之间的差异

如果非重复值的数量少于 1000, Splunk 百分位数函数使用最接近排名算法。请参阅 http://en.wikipedia.org/wiki/Percentile#Nearest_rank。Excel 使用 NIST 插值算法, 也就是说您可以获得实际数据中并不存在的百分位数的值, 这对于最接近排名法来说是不可能的。

非重复值数超过 1000 时 Splunk 算法

如果字段的非重复值数超过 1000, 则系统会使用基于自定义基数树摘要的算法来计算近似百分位数。精确计算所需的内存量与非重复值的数量呈线性关系。因此, 相较于精确计算, 这种算法要快得多、使用的内存要少得多(恒定数量)。默认情况下, 此方法将近似误差限制为小于排名误差的 1%。也就是说, 如果您要求第 95 个百分位数, 则返回的数字会介于第 94 个百分位数和第 96 个百分位数之间。

与 perc 相比, 如果使用 exactperc 函数, 即使非重复值的数量大于 1000, 您也始终可以获得准确的百分位数。

基本示例

考虑这个值列表 $Y = \{10, 9, 8, 7, 6, 5, 4, 3, 2, 1\}$ 。

以下示例将返回 5.5。

```
... | stats perc50(Y)
```

以下示例将返回 9.55。

```
... | stats perc95(Y)
```

延伸示例

考虑以下数据集，这些数据表示一家商店营业时的每小时访客人数：

hour	visitors
0800	0
0900	212
1000	367
1100	489
1, 200	624
1300	609
1400	492
1500	513
1600	376
1700	337

此数据驻留在 visitor_count 索引中。您可以使用 streamstats 命令创建访客的累积总数。

```
index=visitor_count | streamstats sum(visitors) as 'visitors total'
```

搜索结果如下所示：

hour	visitors	visitors total
0800	0	0
0900	212	212
1000	367	579
1100	489	1068
1, 200	624	1692
1300	609	2301
1400	492	2793
1500	513	3306
1600	376	3673
1700	337	4010

让我们加上 stats 命令与perc 函数一起使用，以确定第 50 个和第 95 个百分位数。

```
index=visitor_count | streamstats sum(visitors) as 'visitors total' | stats perc50('visitors total') perc95('visitors total')
```

搜索结果如下所示：

perc50(visitors total)	perc95(visitors total)
1996.5	3858.35

perc50 估计第 50 个百分位数，即 50% 的访客到达时。从数据中您可以看出，访客人数在 1996 至 1997 之间时达到了第 50 个百分位数，发生时间介于 1200 至 1300 小时之间。perc95 估计第 95 个百分位数，即 95% 的访客到达时。当访客人数在 3858 时达到了第 95 个百分位数，发生时间介于 1600 和 1700 之间。

range(X)

描述

返回字段 X 的最大和最小值之间的差异。字段 X 的值必须为数字。

用法

您可以将此函数与 chart、mstats、stats、timechart 和 tstats 命令结合使用，还可以和 sparkline() 图表结合使用。

基本示例

此示例使用列出每个产品和季度数字销量的事件，例如：

产品	季度	销售	配额
ProductA	QTR1	1200	1000
ProductB	QTR1	1400	1550
ProductC	QTR1	1650	1275
ProductA	QTR2	1425	1300
ProductB	QTR2	1175	1425
ProductC	QTR2	1550	1450
ProductA	QTR3	1300	1400
ProductB	QTR3	1250	1125
ProductC	QTR3	1375	1475
ProductA	QTR4	1550	1300
ProductB	QTR4	1700	1225
ProductC	QTR4	1625	1350

如果您还可以确定 min 和 max 值，理解 range 就最简单了。要按产品确定销售范围，运行此搜索：

```
source="adddataData.csv" | chart sum(sales) min(sales) max(sales) range(sales) BY products
```

统计选项卡中显示的结果如下所示：

季度	总计（销售）	最小（销售额）	最大（销售额）	范围（销售额）
QTR1	4250	1200	1650	450
QTR2	4150	1175	1550	375
QTR3	3925	1250	1375	125
QTR4	4875	1550	1700	150

范围（销售额）是最大（销售额）减去最小（销售额）。

延伸示例

请参阅 `max()` 函数更为详细的示例。该示例包括 `range()` 函数。

stdev(X)

描述

返回字段 X 的样本标准偏差。

用法

您可以将此函数与 `chart`、`mstats`、`stats`、`timechart` 和 `tstats` 命令结合使用，还可以和 `sparkline()` 图表结合使用。

基本示例

以下示例将返回通配符字段 `*delay`（应用于 `"delay"` 和 `"xdelay"`）的标准偏差。

```
... | stats stdev(*delay)
```

延伸示例

此搜索使用从 USGS 地震网站下载的近期地震数据。该数据是一个逗号分隔的 ASCII 文本文件，其中包含每次记录的地震的震级 (`mag`)、坐标 (经度、纬度)、区域 (地点) 等。

您可以从 [USGS 地震源](#) 下载当前 CSV 文件，然后将文件上载到 Splunk 实例。此示例使用过去 30 天的所有地震数据。

1. 运行以下搜索以找出按震级类型分类的最近地震震级的平均值、标准偏差和方差。

```
source=usgs place=*California* | stats count mean(mag), stdev(mag), var(mag) BY magType
```

统计选项卡中显示的结果如下所示：

magType	count	mean (mag)	std (mag)	var (mag)
H	123	0.549593	0.356985	0.127438
MbLg	1	0.000000	0.000000	0.000000
Md	1565	1.056486	0.580042	0.336449
Me	2	1.800000	0.346410	0.120000
Ml	1202	1.226622	0.629664	0.396476
Mw	6	3.650000	0.716240	0.513000
ml	10	0.934000	0.560401	0.314049

stdevp(X)

描述

返回字段 X 的总体标准偏差。

用法

您可以将此函数与 `chart`、`mstats`、`stats`、`timechart` 和 `tstats` 命令结合使用，还可以和 `sparkline()` 图表结合使用。

基本示例

延伸示例

sum(X)

描述

返回字段 X 的值的总和。

用法

您可以将此函数与 chart、mstats、stats、timechart 和 tstats 命令结合使用，还可以和 sparkline() 图表结合使用。

基本示例

您可以为任何数字字段创建总计。例如：

```
...| stats sum(bytes)
```

结果形式如下所示：

sum(bytes)
21502

您可以使用 AS 关键字重命名该列：

```
...| stats sum(bytes) AS "total bytes"
```

结果形式如下所示：

total bytes
21502

您可以使用 BY 子句来组织结果：

```
...| stats sum(bytes) AS "total bytes" by date_hour
```

结果形式如下所示：

date_hour	total bytes
07	6509
11	3726
15	6569
23	4698

sumsq(X)

描述

返回字段 X 的值的平方和。

平方和用于评估数据集与数据集平均值的方差。平方和较大表示方差较大，即各个值与平均值之间的波动很大。

用法

您可以将此函数与 chart、mstats、stats、timechart 和 tstats 命令结合使用，还可以和 sparkline() 图表结合使用。

基本示例

下表列出了某一周每天早上 8 点的温度。

您计算温度的平均值，并得出 48.9 度。要计算每天的温度与平均值的偏差，请取温度值减去平均值。如果对每个数字求平方，则会得到如下结果：

星期几	温度	平均值	偏差	温度平方值
星期日	65	48.9	16.1	260.6
星期一	42	48.9	-6.9	47.0
星期二	40	48.9	-8.9	78.4
星期三	31	48.9	-17.9	318.9
星期四	47	48.9	-1.9	3.4
星期五	53	48.9	4.1	17.2
星期六	64	48.9	15.1	229.3

加总平方值，总计为 954.9，然后除以 6，即天数减去 1。这就可以获得这一系列温度的平方和。标准偏差为平方和的平方根。标准偏差越大，一周中的温度波动就越大。

您可以使用一些统计函数来计算平均值、平方和和标准偏差：

```
...|stats mean(temp), sumsq(temp), stdev(temp)
```

此搜索将返回以下结果：

mean(temp)	sumsq(temp)	stdev(temp)
48.857142857142854	17664	12.615183595289349

upperperc<X>(Y)

描述

根据针对数字字段 Y 所请求的百分位数 X 返回近似的百分位数值。

如果值的数量大于 1000，upperperc 函数针对所请求的百分位数给出近似值上限。否则，upperperc 函数返回的百分位数会与 perc 函数返回的相同。

请参阅 `perc<X>(Y)` 函数。

用法

您可以将此函数与 `chart`、`mstats`、`stats`、`timechart` 和 `tstats` 命令结合使用，还可以和 `sparkline()` 图表结合使用。

示例

请参阅 `perc` 函数。

var(X)

描述

返回字段 X 的样本方差。

用法

您可以将此函数与 `chart`、`mstats`、`stats`、`timechart` 和 `tstats` 命令结合使用，还可以和 `sparkline()` 图表结合使用。

示例

请参阅“延伸示例”了解 `mean()` 函数。该示例包括 `var()` 函数。

varp(X)

描述

返回字段 X 的总体方差。

用法

您可以将此函数与 chart、mstats、stats、timechart 和 tstats 命令结合使用，还可以和 sparkline() 图表结合使用。

基本示例

事件顺序函数

使用事件顺序函数根据事件的处理顺序从字段返回值，而顺序不要求一定是时间顺序或时间戳顺序。

下表列出返回的搜索结果事件集的时间戳。此表识别当您使用 first 和 last 事件顺序函数时将返回哪个事件，并将这些函数与 earliest 和 latest 函数进行比较，有关详细信息，可参阅“时间函数”。

_time	事件顺序函数	描述
2020/4/28 0:15:05	first	此事件是搜索结果中的第一个事件。但从时间上来看，这个事件并不是最早的事情。
2020/5/1 0:15:04		
2020/4/30 0:15:02		
2020/4/28 0:15:01		
2020/5/1 0:15:05	latest	从时间上来看，这个事件是搜索结果中的最晚事件。
2020/4/27 0:15:01	earliest last	从时间上来看，这个事件是搜索结果中的最早也是最晚的事件。

请参阅“统计和图表函数概述”。

first(X)

描述

返回字段 X 出现的第一个值。出现的第一个字段值是该字段的最新实例，基于 stats 命令发现事件的顺序。事件的出现顺序不一定是时间顺序。

用法

- 要按照时间顺序查找第一个值，请改用 earliest 函数。
- 当搜索包括紧接在统计或图表命令之前 sort 的命令时，效果最好。
- 该函数将字段值作为字符串进行处理。
- 可以结合 chart、stats 和 timechart 命令使用 first(X) 函数。

基本示例

此示例使用搜索教程中的示例数据。要在您自己的 Splunk 实例中尝试此示例，您必须下载示例数据并遵循操作以将教程数据导入 Splunk。当运行搜索时，请使用时间范围所有时间。

您可运行以下搜索查找特定的 sshd (Secure Shell 守护进程) 的无效用户登录尝试。您可使用 table 命令查看 _time、source 和 _raw 字段中的值。

```
sourcetype=secure invalid user "sshd[5258]" | table _time source _raw
```

统计选项卡中显示的结果如下所示：

_time	source	_raw
2020/4/28 0:15:05	tutorialdata.zip:./mailsv/secure.log	Mon Apr 28 2020 00:15:05 mailsv1 sshd[5258]: 来自 67.170.226.218 端口 1490 ssh2 无效用户 tomcat 的错误密码

2020/5/1 0:15:04	tutorialdata.zip:./www2/secure.log	Thu May 01 2020 00:15:04 www2 sshd[5258]: 来自 130.253.37.97 端口 4284 ssh2 无效用户 brian 的错误密码
2020/4/30 0:15:02	tutorialdata.zip:./www3/secure.log	Wed Apr 30 2020 00:15:02 www3 sshd[5258]: 来自 222.169.224.226 端口 1711 ssh2 无效用户 operator 的错误密码
2020/4/28 0:15:01	tutorialdata.zip:./www1/secure.log	Mon Apr 28 2020 0:15:01 www1 sshd[5258]: 来自 87.194.216.51 端口 3361 ssh2 无效用户 rightscale 的错误密码
2020/5/1 0:15:05	tutorialdata.zip:./mailsv/secure.log	Thu May 01 2020 0:15:05 mailsv1 sshd[5258]: 来自 194.8.74.23 端口 3626 ssh2 无效用户 testuser 的错误密码
2020/4/27 0:15:01	tutorialdata.zip:./www1/secure.log	Sun Apr 27 2020 00:15:01 www1 sshd[5258]: 来自 91.208.184.24 端口 3587 ssh2 无效用户 redmine 的错误密码

您可使用 `first` 函数扩展搜索。

```
sourcetype=secure invalid user "sshd[5258]" | table _time source _raw | stats first(_raw)
```

搜索会返回带时间戳 2020-04-28 00:15:05 的 `_raw` 字段的值，这是最初返回的值列表中的第一个事件。

first(_raw)
Mon Apr 28 2020 00:15:05 mailsv1 sshd[5258]: 来自 67.170.226.218 端口 1490 ssh2 无效用户 tomcat 的错误密码

延伸示例

基本示例使用 `_raw` 字段来显示 `first` 函数的工作方式。这种方式很有用，因为 `_raw` 字段包含时间戳。但是，您可以在任何字段上使用 `first` 函数。

让我们先来创建一些结果。您可以使用 `makergesults` 命令创建一系列结果以测试您的搜索语法。添加 `streamstats` 命令来计算结果：

```
| makergesults count=5 | streamstats count
```

结果如下所示：

_time	计数
2020/5/9 14:35:58	1
2020/5/9 14:35:58	2
2020/5/9 14:35:58	3
2020/5/9 14:35:58	4
2020/5/9 14:35:58	5

在 `count` 字段中，可以使用 `eval` 命令在 `_time` 字段中创建不同的日期。

```
| makergesults count=5 | streamstats count | eval _time=_time-(count*3600)
```

使用 3600（一小时的秒数）来创建一系列小时数。计算会将 `count` 字段中的值乘以一小时的秒数。从原始 `_time` 字段中减去结果，从而得到等于 1 小时前、2 小时前等等的新日期。

结果如下所示：

_time	计数
2020/5/9 13:45:24	1
2020/5/9 12:45:24	2
2020/5/9 11:45:24	3

2020/5/9 10:45:24	4
2020/5/9 9:45:24	5

结果中的每一个小时数比原始日期 2020-05-09 14:24 早。分钟数和秒数也稍有不同，因为每次运行搜索时都会刷新日期。

使用 eval 命令将一个字段添加到搜索中，其值按降序排列：

```
| makeresults count=5 | streamstats count | eval _time=_time-(count*3600) | eval field1=20-count
```

结果如下所示：

_time	计数	field1
2020/5/9 14:45:24	1	19
2020/5/9 13:45:24	2	18
2020/5/9 12:45:24	3	17
2020/5/9 11:45:24	4	16
2020/5/9 10:45:24	5	15

从结果中可以看到，第一个结果是 field1 的最大数字。这表示了结果的处理顺序。首先处理了第一个结果 (20-1=19)，然后再依次处理其余结果。

将 first 函数添加到搜索中时，唯一会返回的值是所指定字段的值：

```
| makeresults count=5 | streamstats count | eval _time=_time-(count*3600) | eval field1=20-count | stats first(field1)
```

结果如下所示：

first(field1)
19

last(X)

描述

返回字段 X 出现的最后一个值。出现的最后一个字段值是该字段的最旧实例，基于 stats 命令发现事件的顺序。事件的出现顺序不一定是时间顺序。

用法

- 要按照时间顺序查找最后一个值，请改用 latest 函数。
- 当搜索包括紧接着在统计或图表命令之前 sort 的命令时，效果最好。
- 该函数将字段值作为字符串进行处理。

可以结合 chart、stats 和 timechart 命令使用 last(X) 函数。

基本示例

以下示例返回每个非重复 "sourcetype" 的第一个 "log_level" 值。

此示例使用搜索教程中的示例数据。要在您自己的 Splunk 实例中尝试此示例，您必须下载示例数据并遵循操作以将教程数据导入 Splunk。当运行搜索时，请使用时间范围所有时间。

您可运行以下搜索查找特定的 sshd (Secure Shell 守护进程) 的无效用户登录尝试。您可使用表格命令查看 _time、source 和 _raw 字段中的值。

```
sourcetype=secure invalid user "sshd[5258]" | table _time source _raw
```

统计选项卡中显示的结果如下所示：

_time	source	_raw
2020/4/28 0:15:05	tutorialdata.zip:./mailsv/secure.log	Mon Apr 28 2020 00:15:05 mailsv1 sshd[5258]: 来自 67.170.226.218 端口 1490 ssh2 无效用户 tomcat 的错误密码
2020/5/1 0:15:04	tutorialdata.zip:./www2/secure.log	Thu May 01 2020 00:15:04 www2 sshd[5258]: 来自 130.253.37.97 端口 4284 ssh2 无效用户 brian 的错误密码
2020/4/30 0:15:02	tutorialdata.zip:./www3/secure.log	Wed Apr 30 2020 00:15:02 www3 sshd[5258]: 来自 222.169.224.226 端口 1711 ssh2 无效用户 operator 的错误密码
2020/4/28 0:15:01	tutorialdata.zip:./www1/secure.log	Mon Apr 28 2020 0:15:01 www1 sshd[5258]: 来自 87.194.216.51 端口 3361 ssh2 无效用户 rightscale 的错误密码
2020/5/1 0:15:05	tutorialdata.zip:./mailsv/secure.log	Thu May 01 2020 0:15:05 mailsv1 sshd[5258]: 来自 194.8.74.23 端口 3626 ssh2 无效用户 testuser 的错误密码
2020/4/27 0:15:01	tutorialdata.zip:./www1/secure.log	Sun Apr 27 2020 00:15:01 www1 sshd[5258]: 来自 91.208.184.24 端口 3587 ssh2 无效用户 redmine 的错误密码

您可使用 `last` 函数扩展搜索。

```
sourcetype=secure invalid user "sshd[5258]" | table _time source _raw | stats last(_raw)
```

搜索会返回带 `_time` 值 2020-04-27 00:15:01 的事件，这是事件列表中的最后一个事件。但是，这并不是最后一个时间事件。

_time	source	_raw
2020/4/27 0:15:01	tutorialdata.zip:./www1/secure.log	Sun Apr 27 2020 00:15:01 www1 sshd[5258]: 来自 91.208.184.24 端口 3587 ssh2 无效用户 redmine 的错误密码

延伸示例

基本示例使用 `_raw` 字段来显示 `last` 函数的工作方式。这种方式很有用，因为 `_raw` 字段包含时间戳。但是，您可以在任何字段上使用 `last` 函数。

让我们先来创建一些结果。您可以使用 `makergesults` 命令创建一系列结果以测试您的搜索语法。添加 `streamstats` 命令来计算结果：

```
| makergesults count=5 | streamstats count
```

结果如下所示：

_time	计数
2020/5/9 14:35:58	1
2020/5/9 14:35:58	2
2020/5/9 14:35:58	3
2020/5/9 14:35:58	4
2020/5/9 14:35:58	5

在 `count` 字段中，可以使用 `eval` 命令在 `_time` 字段中创建不同的日期。

```
| makergesults count=5 | streamstats count | eval _time=_time-(count*86400)
```

使用 86400（一天的秒数）来创建一系列天数。计算会将 `count` 字段中的值乘以一天的秒数。从原始 `_time` 字段中减去结果，从而得到等于 1 天前、2 天前等等的新日期。

结果如下所示：

_time	计数
2020/5/8 14:45:24	1

2020/5/7 14:45:24	2
2020/5/6 14:45:24	3
2020/5/5 14:45:24	4
2020/5/4 14:45:24	5

结果中的每一个天数比原始日期 2020-05-09 14:45:24 早。分钟数和秒数也稍有不同，因为每次运行搜索时都会刷新日期。

使用 eval 命令将一个字段添加到搜索中，其值按降序排列：

```
| makeresults count=5 | streamstats count | eval _time=_time-(count*86400) | eval field1=20-count
```

结果如下所示：

_time	计数	field1
2020/5/8 14:45:24	1	19
2020/5/7 14:45:24	2	18
2020/5/6 14:45:24	3	17
2020/5/5 14:45:24	4	16
2020/5/4 14:45:24	5	15

从结果中可以看到，最后一个结果是 field1 的最小数字。这表示了结果的处理顺序。系统先处理所有其他结果，最后才处理第五个结果 (20-5=15)。

将 last 函数添加到搜索中时，唯一会返回的值是所指定字段的值：

```
| makeresults count=5 | streamstats count | eval _time=_time-(count*86400) | eval field1=20-count | stats last(field1)
```

结果如下所示：

lastfield1
15

另请参阅

命令

Eval

makeresults

多值统计和 chart 函数

list(X)

描述

以多值条目的形式返回字段 X 最多 100 个值的列表。值的顺序反映输入事件的顺序。

用法

- 如果字段 X 中的值超过 100 个，则只返回前 100 个值。
- 该函数将字段值作为字符串进行处理。
- 您可以将 list(X) 函数与 chart、stats 和 timechart 命令结合使用。

基本示例

让我们生成一些简单的结果来介绍 list 函数的作用。

1. 使用 `makeresults` 和 `streamstats` 命令生成仅仅是时间戳的一组结果和用作行数的结果计数。

```
| makeresults count=1000 | streamstats count AS rowNumber
```

统计选项卡中显示的结果如下所示：

_time	rowNumber
2018/4/2 20:27:11	1
2018/4/2 20:27:11	2
2018/4/2 20:27:11	3
2018/4/2 20:27:11	4
2018/4/2 20:27:11	5

注意每个结果单独成行出现。

2. 使用 `list` 函数向搜索添加 `stats` 命令。数字以升序单个多值结果的形式返回。

```
| makeresults count=1000 | streamstats count AS rowNumber | stats list(rowNumber) AS numbers
```

统计选项卡中显示的结果如下所示：

数字
1
2
3
4
5

注意这是一个单个结果。没有可替换的行背景颜色。

3. 将此结果与 `values` 函数返回的结果进行比较。

values(X)

描述

以多值条目的形式返回字段 X 所有唯一值的列表。各值按字典顺序排列。

用法

- 默认情况下，返回的值的数量没有限制。拥有适当权限的用户可在 `limits.conf` 文件中指定限制。您可以使用 `maxvalues` 设置在 `[stats | sistats]` 段落中指定限制。
- 该函数将字段值作为字符串进行处理。
- 您可以将 `values(X)` 函数与 `chart`、`stats`、`timechart` 和 `tstats` 命令结合使用。

基于用于编码计算机内存中的项目的值按字典顺序对这些项目进行排序。在 Splunk 软件中，几乎都是使用 UTF-8 进行编码，这是 ASCII 的超集。

- 数字排在字母前面。根据第一位数字对数字进行排序。例如数字 10、9、70、100，按照字典顺序排序为 10、100、70、9。
- 大写字母排在小写字母前面。
- 符号的排序标准不固定。有些符号排在数字值前面。有些符号排在字母前面或后面。

基本示例

让我们生成一些简单的结果来介绍 `values` 函数的作用。

1. 使用 `makeresults` 和 `streamstats` 命令生成仅仅是时间戳的一组结果和用作行数的结果计数。

```
| makeresults count=1000 | streamstats count AS rowNum
```

统计选项卡中显示的结果如下所示：

_time	rowNumber
2018/4/2 20:27:11	1
2018/4/2 20:27:11	2
2018/4/2 20:27:11	3
2018/4/2 20:27:11	4
2018/4/2 20:27:11	5

注意每个结果单独成行出现。

2. 使用 values 函数向搜索添加 stats 命令。结果将按词典顺序返回。

```
| makeresults count=1000 | streamstats count AS rowNum | stats values(rowNum) AS numbers
```

统计选项卡中显示的结果如下所示：

数字
1
10
100
1000
101
102
103
104
105
106
107
108
109
11
110

注意这是一个单个结果。没有可替换的行背景颜色。

3. 将这些结果与 list 函数返回的结果进行比较。

时间函数

earliest(X)

描述

返回时间上最早出现的字段 X 的值。

用法

- 该函数将字段值作为字符串进行处理。
- 您可以将 earliest(X) 函数与 chart、mstats、stats、timechart 和 tstats 命令结合使用。

基本示例

此示例使用搜索教程中的示例数据。要在您自己的 Splunk 实例中尝试此示例，您必须下载示例数据并遵循操作以将教程数据导入 Splunk。当运行搜索时，请使用时间范围所有时间。

您可运行以下搜索查找 sshd (Secure Shell 守护进程) 的无效用户登录尝试。您可使用表格命令查看 _time、source 和 _raw 字段中的值。

```
sourcetype=secure invalid user "sshd[5258]" | table _time source _raw
```

统计选项卡中显示的结果如下所示：

_time	source	_raw
2018-04-30 00:15:05	tutorialdata.zip:./mailsv/secure.log	Mon Apr 28 2018 00:15:05 mailsv1 sshd[5258]: 来自 67.170.226.218 端口 1490 ssh2 无效用户 tomcat 的错误密码
2018-04-29 00:15:04	tutorialdata.zip:./www2/secure.log	Thu May 01 2018 00:15:04 www2 sshd[5258]: 来自 130.253.37.97 端口 4284 ssh2 无效用户 brian 的错误密码
2018-04-29 00:15:02	tutorialdata.zip:./www3/secure.log	Wed Apr 30 2018 00:15:02 www3 sshd[5258]: 来自 222.169.224.226 端口 1711 ssh2 无效用户 operator 的错误密码
2018-04-28 00:15:01	tutorialdata.zip:./www1/secure.log	Mon Apr 28 2018 00:15:01 www1 sshd[5258]: 来自 87.194.216.51 端口 3361 ssh2 无效用户 rightscale 的错误密码
2018-04-28 00:15:05	tutorialdata.zip:./mailsv/secure.log	Thu May 01 2018 00:15:05 mailsv1 sshd[5258]: 来自 194.8.74.23 端口 3626 ssh2 无效用户 testuser 的错误密码
2018-04-27 00:15:01	tutorialdata.zip:./www1/secure.log	Sun Apr 27 2018 00:15:01 www1 sshd[5258]: 来自 91.208.184.24 端口 3587 ssh2 无效用户 redmine 的错误密码

您可使用 earliest 函数扩展搜索。

```
sourcetype=secure invalid user "sshd[5258]" | table _time source _raw | stats earliest(_raw)
```

搜索会返回带 _time 值 2018-04-27 00:15:01 的事件，事件时间戳为最旧的时间戳。

_time	source	_raw
2018-04-27 00:15:01	tutorialdata.zip:./www1/secure.log	Sun Apr 27 2018 00:15:01 www1 sshd[5258]: 来自 91.208.184.24 端口 3587 ssh2 无效用户 redmine 的错误密码

earliest_time(x)

描述

返回指定字段值按时间顺序最早出现的 UNIX 时间。

用法

- 该函数将字段值作为字符串进行处理。
- 您可以将 earliest_time(X) 函数与 mstats、stats 和 tstats 命令结合使用。
- 如果您有指标数据，您可以将 earliest_time(x) 与 earliest(x)、latest(x) 和 latest_time(x) 结合使用以计算计数器的增长率。或者，您可以使用 rate(x) 计数器进行计算。

基本示例

以下搜索针对指标数据运行。旨在返回以 deploy 开始的各 metric_name 每分钟的最早 UNIX 时间值。

```
| mstats earliest_time(_value) where index=_metrics metric_name=deploy* BY metric_name span=1m
```

统计选项卡中显示的结果如下所示：

_time	metric_name	earliest_time(_value)
2018-11-11 18:14:00	deploy-connections.nCurrent	1,541,988,860.000000

2018-11-11 18:14:00	deploy-connections. nStarted	1, 541, 988, 860. 000000
2018-11-11 18:14:00	deploy-server. volumeCompletedKB	1, 541, 988, 860. 000000
2018-11-11 18:15:00	deploy-connections. nCurrent	1, 541, 988, 922. 000000
2018-11-11 18:15:00	deploy-connections. nStarted	1, 541, 988, 922. 000000
2018-11-11 18:15:00	deploy-server. volumeCompletedKB	1, 541, 988, 922. 000000

latest(X)

描述

返回时间上最晚出现的字段 X 的值。

用法

- 该函数将字段值作为字符串进行处理。
- 您可以将 `latest(X)` 函数与 `chart`、`mstats`、`stats`、`timechart` 和 `tstats` 命令结合使用。

基本示例

此示例使用搜索教程中的示例数据。要在您自己的 Splunk 实例中尝试此示例，您必须下载示例数据并遵循操作以将教程数据导入 Splunk。当运行搜索时，请使用时间范围所有时间。

您可运行以下搜索查找特定的 sshd (Secure Shell 守护进程) 的无效用户登录尝试。您可使用表格命令查看 `_time`、`source` 和 `_raw` 字段中的值。

```
sourcetype=secure invalid user "sshd[5258]" | table _time source _raw
```

统计选项卡中显示的结果如下所示：

<code>_time</code>	<code>source</code>	<code>_raw</code>
2018-04-28 00:15:05	tutorialdata.zip:./mailsv/secure.log	Mon Apr 28 2018 00:15:05 mailsv1 sshd[5258]: 来自 67.170.226.218 端口 1490 ssh2 无效用户 tomcat 的错误密码
2018-05-01 00:15:04	tutorialdata.zip:./www2/secure.log	Thu May 01 2018 00:15:04 www2 sshd[5258]: 来自 130.253.37.97 端口 4284 ssh2 无效用户 brian 的错误密码
2018-04-30 00:15:02	tutorialdata.zip:./www3/secure.log	Wed Apr 30 2018 00:15:02 www3 sshd[5258]: 来自 222.169.224.226 端口 1711 ssh2 无效用户 operator 的错误密码
2018-04-28 00:15:01	tutorialdata.zip:./www1/secure.log	Mon Apr 28 2018 00:15:01 www1 sshd[5258]: 来自 87.194.216.51 端口 3361 ssh2 无效用户 rightscale 的错误密码
2018-05-01 00:15:05	tutorialdata.zip:./mailsv/secure.log	Thu May 01 2018 00:15:05 mailsv1 sshd[5258]: 来自 194.8.74.23 端口 3626 ssh2 无效用户 testuser 的错误密码
2018-04-27 00:15:01	tutorialdata.zip:./www1/secure.log	Sun Apr 27 2018 00:15:01 www1 sshd[5258]: 来自 91.208.184.24 端口 3587 ssh2 无效用户 redmine 的错误密码

您可使用 `latest` 函数扩展搜索。

```
sourcetype=secure invalid user "sshd[5258]" | table _time source _raw | stats latest(_raw)
```

搜索会返回带 `_time` 值 2018-05-01 00:15:05 的事件，该事件有最新的时间戳。

<code>_time</code>	<code>source</code>	<code>_raw</code>
2018-05-01 00:15:05	tutorialdata.zip:./mailsv/secure.log	Thu May 01 2018 00:15:05 mailsv1 sshd[5258]: 来自 194.8.74.23 端口 3626 ssh2 无效用户 testuser 的错误密码

`latest_time(x)`

描述

返回指定字段值按时间顺序最晚出现的 UNIX 时间。

用法

- 该函数将字段值作为字符串进行处理。
- 您可以将 `latest_time(X)` 函数与 `mstats`、`stats` 和 `tstats` 命令结合使用。
- 如果您有指标数据，您可以将 `latest_time(x)` 与 `earliest(x)`、`latest(x)` 和 `earliest_time(x)` 结合使用以计算计数器的增长率。或者，您可以使用 `rate(x)` 计数器进行同样的操作。

基本示例

以下搜索针对指标数据运行。旨在返回名称以 `deploy` 开始的指标过去 60 分钟内最早的 UNIX 时间值。

```
| mstats latest_time(_value) where index=_metrics metric_name=queue.* BY metric_name span=1m
```

统计选项卡中显示的结果如下所示：

<code>_time</code>	<code>metric_name</code>	<code>earliest_time(_value)</code>
2018-11-13 14:43:00	queue.current_size	1, 542, 149, 039. 000000
2018-11-13 14:43:00	queue.current_size_kb	1, 542, 149, 039. 000000
2018-11-13 14:43:00	queue.largest_size	1, 542, 149, 039. 000000
2018-11-13 14:43:00	queue.max_size_kb	1, 542, 149, 039. 000000
2018-11-13 14:43:00	queue.smallest_size	1, 542, 149, 039. 000000
2018-11-13 14:44:00	queue.current_size	1, 542, 149, 070. 000000
2018-11-13 14:44:00	queue.current_size_kb	1, 542, 149, 070. 000000
2018-11-13 14:44:00	queue.largest_size	1, 542, 149, 070. 000000
2018-11-13 14:44:00	queue.max_size_kb	1, 542, 149, 070. 000000
2018-11-13 14:44:00	queue.smallest_size	1, 542, 149, 070. 000000

`per_day(X)`

描述

返回每天字段 X 或 eval 表达式 X 的值。

用法

- 可以将 `per_day(X)` 函数与 `timechart` 命令结合使用。

基本示例

以下示例返回每天字段 `total` 的值。

```
... | timechart per_day(total)
```

以下示例返回 eval 表达式 `eval(method="GET") AS Views` 的结果。

```
... | timechart per_day(eval(method="GET")) AS Views
```

延伸示例

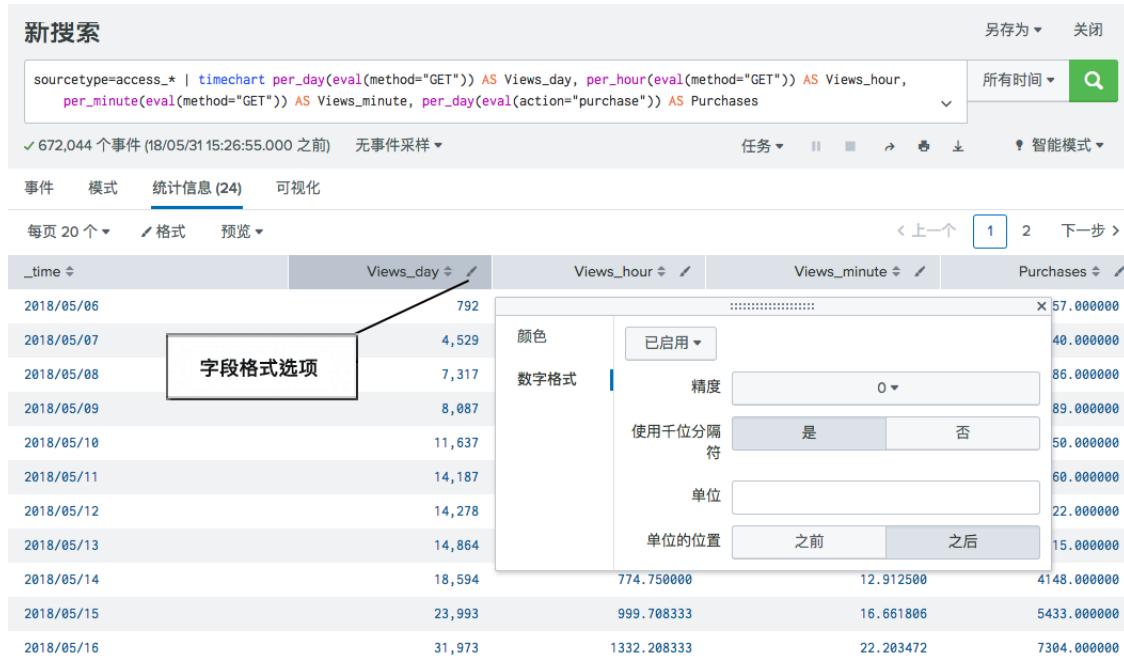
此示例使用搜索教程中的示例数据集，但应该可以与任意格式的 Apache Web 访问日志配合使用。从搜索教程中的本主题下载数据集，并按照说明将其上载到 Splunk 部署。

此搜索使用 `per_day()` 函数和 `eval` 表达式确定查看网页的次数以及购买项目的次数。结果将显示在“统计”选项卡中。

```
sourcetype=access_* | timechart per_day(eval(method="GET")) AS Views_day, per_day(eval(action="purchase")) AS Purchases
```

要确定每小时、每分钟或者每秒的查看或购买次数，您可以将其他时间函数添加到搜索。例如：

```
sourcetype=access_* | timechart per_day(eval(method="GET")) AS Views_day, per_hour(eval(method="GET")) AS Views_hour,  
per_minute(eval(method="GET")) AS Views_minute, per_day(eval(action="purchase")) AS Purchases
```



使用字段格式选项更改字段值的数字格式。

per_hour(X)

描述

返回每小时字段 X 或 eval 表达式 X 的值。

用法

- 可以将 `per_hour(X)` 函数与 `timechart` 命令结合使用。

基本示例

以下示例每小时返回字段 `total` 的值。

```
... | timechart per_hour(total)
```

以下示例返回 eval 表达式 `eval(method="POST")) AS Views` 的结果。

```
... | timechart per_hour(eval(method="POST")) AS Views
```

per_minute(X)

描述

返回每分钟字段 X 或 eval 表达式 X 的值。

用法

- 可以将 `per_minute(X)` 函数与 `timechart` 命令结合使用。

基本示例

以下示例返回每分钟字段 `total` 的值。

```
... | timechart per_minute(total)
```

以下示例返回 `eval` 表达式 `eval(method="GET")) AS Views` 的结果。

```
... | timechart per_minute(eval(method="GET")) AS Views
```

per_second(X)

描述

返回每秒字段 `X` 或 `eval` 表达式 `X` 的值。

用法

- 可以将 `per_second(X)` 函数与 `timechart` 命令结合使用。

基本示例

以下示例返回每秒字段 `kb` 的值。

```
... | timechart per_second(kb)
```

rate(X)

描述

返回字段值的每秒速率变化。表示 `(latest(X) - earliest(X)) / (latest_time(X) - earliest_time(X))`。如果至少有一次重置，还可以处理最大的值重置。

用法

- 您可以将 `rate(X)` 函数与 `mstats`、`stats` 和 `tstats` 命令结合使用。
- 针对累计计数器指标提供每秒速率变化。累计计数器指标报告自上次计数器重置以来的总计数器值。
- 要求 `earliest(X)` 和 `latest(X)` 字段值是数字，`earliest_time(X)` 和 `latest_time(X)` 值为不同值。
- 要求搜索时间范围内至少两个指标数据点。
- 应该用于提供有关单个而非多个计数器的速率信息。

基本示例

以下搜索针对指标数据运行。针对提供传入网络流量测试值的指标提供每小时点击率。使用 `processor` 筛选器确保不报告多个指标系列（`name` 和 `processor` 组合）。

```
| mstats rate(traffic.incoming) as rate_hits where index=_metrics name=indexerpipe processor=index_thrput span=1h
```

结果图表向您显示 `traffic.incoming` 指标的计数器点击率在下午 1 点、4 点和上午 11 点达到高峰，但是其他时间保持稳定。



rate_avg(X)

描述

计算累计计数器指标的每个指标时间序列速率。返回这些速率的平均值。

有关指标时间序列的详细介绍，请参阅《指标》中的“对指标时间序列执行统计计算”。

用法

- 可以将 `rate_avg(X)` 函数与 `mstats` 命令结合使用。
- 当您计算累计计数器指标的平均速度时，最清楚的方法是按指标时间序列将计数器指标速度计算拆分出来，然后计算所有指标时间序列的平均速度。
- 与 `rate(X)` 不同，`rate_avg(X)` 函数即使在每个时间跨度的每个时间序列中只有一个指标数据点时也可以计算速率。为了计算速度，它可以在必要时跨时间跨度地提取数据。
- `rate_avg(X)` 函数不支持 `prestats=true`。它需要最终的维度列表才能进行拆分。

基本示例

在 `_metrics` 索引中，您具有指标 `spl.intr.resource_usage.PerProcess.data.elapsed` 的数据。这是一个累计计数器指标。它包含多个指标时间序列。

以下示例搜索使用 `rate_avg` 函数为时间范围内的每个 `spl.mlog.thruput.thruput.total_k_processed` 时间序列计算 `rate(X)`。然后，它会得出所有时间序列的平均速率。最后，它会按时间拆分结果，以便这些结果可以绘制在图表上。

```
| mstats rate_avg(spl.mlog.thruput.thruput.total_k_processed) where index=_metrics span=1h
```

rate_sum(X)

描述

计算累计计数器指标的每个指标时间序列速率。返回这些速率的总和。

有关指标时间序列的详细介绍，请参阅《指标》中的“对指标时间序列执行统计计算”。

用法

- 可以将 `rate_sum(X)` 函数与 `mstats` 命令结合使用。
- 当您计算累计计数器指标的合计速度时，最清楚的方法是按指标时间序列将计数器指标速度计算拆分出来，然后计算所有指标时间序列的合计速度。
- 与 `rate(X)` 不同，`rate_sum(X)` 函数即使在每个时间跨度的每个时间序列中只有一个指标数据点时也可以计算速率。为了计算速度，它可以在必要时跨时间跨度地提取数据。
- `rate_sum(X)` 函数不支持 `prestats=true`。它需要最终的维度列表才能进行拆分。

基本示例

在 `_metrics` 索引中，您具有指标 `spl.intr.resource_usage.PerProcess.data.elapsed` 的数据。这是一个累计计数器指标。它包含多个指标时间序列。

以下示例搜索使用 `rate_sum` 函数为时间范围内的每个 `spl.mlog.thruput.thruput.total_k_processed` 时间序列计算 `rate(X)`。然后，它会得出所有时间序列的合计速率。最后，它会按时间拆分结果，以便这些结果可以绘制在图表上。

```
| mstats rate_sum(spl.mlog.thruput.thruput.total_k_processed) where index=_metrics span=1h
```

时间格式变量和修饰符

日期和时间格式变量

本主题中列出的变量可用于定义评估函数、`strftime()` 和 `strptime()` 中的时间格式。您还可以使用这些变量来描述事件数据中的时间戳。

此外，您可以将 `relative_time()` 和 `now()` 时间函数用作参数。

有关使用日期和时间的更多信息，请参阅《搜索手册》中“与搜索结合使用的时间调节器”和“关于涉及时间的搜索”。

请参考所有允许的时区值的 `tz` 数据库时区列表。有关 Splunk 软件如何确定时区和 `tz` 数据库的更多信息，请参阅《数据导入》中的“指定时间戳的时区”。

亚秒级时间变量（例如 `%N` 和 `%Q`）可用于拥有毫秒级时间戳分辨率的指标索引的指标搜索。

有关启用指标索引以便为毫秒级时间戳精度的指标数据点建立索引的更多信息，请参阅：

- 《Splunk Cloud 管理员手册》中的“管理 Splunk Cloud 索引”（如果您使用的是 Splunk Cloud）。
- 《管理索引器和索引器群集》中“创建自定义索引”（如果您使用的是 Splunk Enterprise）。

日期和时间变量

变量	描述
<code>%c</code>	由服务器操作系统定义，以当前区域设置的格式显示的日期和时间。例如，Linux 上的美国英语日期和时间为 <code>Thu Jul 18 09:30:00 2019</code> 。
<code>%+</code>	由服务器操作系统定义，以当前区域设置的格式显示的带时区的日期和时间。例如，Linux 上的美国英语日期和时间为 <code>Thu Jul 18 09:30:00 PDT 2019</code> 。

时间变量

变量	描述
<code>%Ez</code>	Splunk 专用的时区（分钟）。
<code>%H</code>	以十进制表示的小时数（24 小时制）。以值 00 至 23 表示小时数。可使用前导零，但不做强制要求。
<code>%I</code>	以 01 至 12 之间的值表示小时（12 小时制）。可使用前导零，但不做强制要求。
<code>%k</code>	如 <code>%H</code> ，其中小时以十进制表示（24 小时制）。前导零由空格代替，例如 0 至 23。
<code>%M</code>	以十进制表示的分钟。以值 00 至 59 表示分钟数。可使用前导零，但不做强制要求。
<code>%N</code>	次秒的位数。默认值为 <code>%9N</code> 。您可以指定 <code>%3N</code> = 毫秒， <code>%6N</code> = 微秒， <code>%9N</code> = 纳秒。
<code>%p</code>	AM 或 PM。
<code>%Q</code>	UTC 时间戳的次秒组件。默认值为毫秒 <code>%3Q</code> 。有效值包括： <ul style="list-style-type: none">• <code>%3Q</code> = 毫秒，其中值为 000-999• <code>%6Q</code> = 微秒，其中值为 000000-999999• <code>%9Q</code> = 纳秒，其中值为 00000000-999999999
<code>%S</code>	以十进制表示的秒钟，例如 00 至 59。
<code>%s</code>	Unix Epoch Time 时间戳，或从 Epoch 时间：1970-01-01 00:00:00 +0000 (UTC)。（1484993700 代表 2020 年 1 月 21 日周二 10:15:00）
<code>%T</code>	以 24 小时制表示的时间（ <code>%H:%M:%S</code> ）。例如：23:59:59。
<code>%X</code>	以当前区域设置的格式显示的时间。 <code>9:30 AM</code> 的美国英语格式为 <code>9:30:00</code> 。
<code>%Z</code>	时区缩写。例如，EST 是美国东部标准时间的缩写。
	与 UTC 的时区偏移，以小时和分钟表示： <code>+hhmm</code> 或 <code>-hhmm</code> 。例如，比 UTC 早 5 个小时的时间值表示为 <code>-0500</code> ，这是美国东部标准时间。

%z	<p>示例：</p> <ul style="list-style-type: none"> 使用 %z 指定小时和分钟，例如，-0500 使用 %:z 指定以冒号分隔的小时和分钟，例如，-5:00 使用 %::z 指定以冒号分隔的小时、分钟和秒，例如，-05:00:00 使用 %:::z 仅指定小时，例如，-05
%%	代表 "%" 文字字符。

日期变量

变量	描述
%F	相当于 %Y-%m-%d (ISO 8601 日期格式)。
%x	以当前区域设置的格式显示的日期。例如：美国英语日期 7/13/2019。

指定天数和周数

变量	描述
%A	完整的星期名称。(Sunday, ..., Saturday)
%a	缩写的星期名称。(Sun, ..., Sat)
%d	以十进制表示的月中的某一日，包括前导零。(01 到 31)
%e	与 %d 相似，以十进制表示的月中的某一日，但前导零由空格代替。(1 到 31)
%j	以十进制表示的一年中的某一日，包括前导零。(001 到 366)
%V (或 %U)	一年中的某一周。%V 变量从 1 开始计数，这是最常见的起始编号。%U 变量从 0 开始计数。
%w	以十进制表示的星期。(0 = 星期日, ..., 6 = 星期六)

指定月份

变量	描述
%b	缩写的月份名称。(一月、二月等等)
%B	完整的月份名称。(一月份、二月份等等)
%m	以十进制表示的月份。(01 至 12)。可使用前导零，但不做强制要求。

指定年份

变量	描述
%y	以十进制表示的年份，不包含世纪。(00 至 99)。可使用前导零，但不做强制要求。
%Y	以十进制表示的年份，包含世纪。例如：2020。

示例

下表列出了一些时间格式字符串的结果：

时间格式字符串	结果
%Y-%m-%d	2019/12/31
%y-%m-%d	19-12-31
%b %d, %Y	Feb 11, 2020
%d%b '%y = %Y-%m-%d	23 Apr '20 = 2020-04-23

下表列出了使用时间变量的搜索的结果：

搜索示例	结果
host="www1" eval WeekNo = strftime(_time, "%V")	创建一个名为 WeekNo 的字段，并返回对应于 _time 字段中的日期的周的数字值。
... eval mytime=strftime(_time, "%Y-%m-%dT%H:%M:%S.%Q")	创建一个名为 mytime 的字段，并返回转换后的 _time 字段中的时间戳值。这些值以 UNIX 格式存储，并使用指定的格式（ISO 8601 格式）进行转换。例如：2020-04-13T14:00:15.000。

时间调节器

使用时间调节器来自定义搜索的时间范围或更改搜索结果中时间戳的格式。

搜索时间和字段

当 Splunk 软件处理某一事件时，该事件的时间戳保存为默认字段 _time。此时间戳（事件发生的时间）以 UNIX 时间表示法保存。使用相对时间调节器 earliest 或 latest 执行搜索时，将查找以时间戳开始、结束或位于指定时间戳之间的所有事件。

例如，当您搜索 earliest=@d 时，搜索查找从午夜起具有 _time 值的所有事件。本示例使用日期格式变量 @d。请参阅“日期和时间格式变量”。

时间调节器和时间范围挑选器

当您在 SPL 语法中使用时间调节器时，该时间会覆盖时间范围挑选器中指定的时间。

例如，假设您的搜索在时间范围挑选器中使用了 yesterday。您将时间调节器 earliest=-2d 添加到您的搜索语法中。搜索使用时间调节器中指定的时间，而忽略时间范围挑选器中的时间。由于搜索未指定 latest 时间调节器，因此默认值 now 用于 latest。

有关更多信息，请参阅《搜索手册》中的“在搜索中指定时间调节器”。

时间范围和子搜索

从时间范围挑选器中选择的时间范围会适用于基本搜索或子搜索。

但是，直接在基本搜索中指定的时间范围不会应用于子搜索。同样地，直接在子搜索中指定的时间范围仅适用于该子搜索。该时间范围不适用于基本搜索或任何其他子搜索。

例如，如果时间范围挑选器设置为过去 7 天，并且子搜索包含 earliest=2d@d，则最早时间调节器仅适用于子搜索，而过去 7 天适用于基本搜索。

根据索引时间进行搜索

您还可以选择根据事件建立索引的时间来搜索事件。UNIX 时间保存在 _indexetime 字段中。与 earliest 和 latest（用于 _time 字段）相似，您可以使用相对时间调节器 _index_earliest 和 _index_latest 基于 _indexetime 搜索事件。例如，如果您想要搜索在前一小时建立索引的事件，则使用：_index_earliest=-h@h _index_latest=@h。

使用诸如 _index_earliest 和 _index_latest 等基于索引时间的调节器时，您的搜索还必须包含一个检索事件的事件时间窗口。换句话说，可能基于非索引时间窗口以及索引时间窗口排除事件数据块。要确保基于索引时间检索每个事件，您必须使用所有时间运行您的搜索。

时间调节器的列表

使用 earliest 和 latest 修饰符指定自定义和相对时间范围。您可以指定精确时间（如 earliest="10/5/2016:20:00:00"）或相对时间（earliest=-h 或 latest=@w6）。

指定相对时间时，您可以使用 now 修饰符来代表当前时间。

修饰符	语法	描述
-----	----	----

earliest	<code>earliest=[+ -] <time_integer><time_unit>@<time_unit></code>	指定搜索的时间范围的 earliest_time。 使用 earliest=1 指定 UNIX epoch 时间 1，其中 UTC 为 1970 年 1 月 1 日，上午 12:00:01。 使用 earliest=0 指定数据中最早的事件。
_index_earliest	<code>_index_earliest=[+ -] <time_integer><time_unit>@<time_unit></code>	指定搜索的时间范围的 earliest_indextime。
_index_latest	<code>_index_latest=[+ -] <time_integer><time_unit>@<time_unit></code>	指定搜索的时间范围的 latest_indextime。
latest	<code>latest=[+ -] <time_integer><time_unit>@<time_unit></code>	指定搜索的 _time 范围的最晚时间。
now	now() 或 now	指当前时间。如果设置为 earliest, now() 为搜索的开始时间。
时间	time()	在实时搜索中, time() 为当前机器时间。

有关自定义搜索窗口的更多信息，请参阅《搜索手册》中的“在搜索中指定实时时间范围窗口”。

如何指定相对时间调节器

可使用表示时间量（整数和单位）的字符串在搜索中定义相对时间。您也可以指定一个“对齐到”时间单位，由时间单位后面的 @ 符号表示。

使用时间调节器的语法是 `[+|-]<time_integer><time_unit>@<time_unit>`

指定相对时间调节器的步骤是：

1. 表示当前时间的时间偏移。
2. 使用数字和单位定义时间量。
3. 指定一个“对齐到”时间单位。此时间单位表示您的时间量将舍入到的最早或最晚时间。

表示时间偏移

在字符串的前面加上加号 (+) 或减号 (-) 来表示与当前时间的偏移。

定义时间量

使用数字和单位定义时间量。下表列出了所支持的时间单位。

时间单位	有效的单位缩写
subseconds	微秒 (us)、毫秒 (ms)、百分之一秒 (cs) 或十分之一秒 (ds)
秒	s, sec, secs, second, seconds
分	m, min, minute, minutes
时	h, hr, hrs, hour, hours
天	d, day, days
周	w, week, weeks
月	mon, month, months
季度	q, qtr, qtrs, quarter, quarters
年	y, yr, yrs, year, years

例如，要在一小时前开始搜索，可使用以下时间调节器中的任意一个。

```
earliest=-h
```

或

```
earliest=-60m
```

指定单个时间量时，将隐含数字：'s' 等同于 '1s'，'m' 等同于 '1m'，'h' 等同于 '1h'，等等。

只有在指标搜索是针对启用了毫秒级时间戳分辨率的指标索引展开时，才能使用诸如 ms 之类的亚秒级时间刻度。

有关启用指标索引以便为毫秒级时间戳精度的指标数据点建立索引的更多信息，请参阅：

- 《Splunk Cloud 管理员手册》中的“管理 Splunk Cloud 索引”（如果您使用的是 Splunk Cloud）。
- 《管理索引器和索引器群集》中“创建自定义索引”（如果您使用的是 Splunk Enterprise）。

指定一个对齐到时间单位

您可以指定一个“对齐到”时间单位。此时间单位表示您的时间量将舍入到的最早或最晚时间。用 "@" 符号将时间量与“对齐到”时间单位分开。

- 您可以使用前面列出的任何时间单位。例如：
 - @w, @week 和 @w0 表示星期日
 - @month 表示一个月的开始
 - @q, @qtr 或 @quarter 表示最近季度的开始（1 月 1 日、4 月 1 日、7 月 1 日或 10 月 1 日）。
- 您可以指定星期几：w0（星期日）、w1、w2、w3、w4、w5 和 w6（星期六）。对于星期日，可以指定 w0 或 w7。
- 您还可以指定与对齐到时间之间的偏移，或者与时间调节器结合使用，以获得更具体的相对时间定义。例如，@d-2h 对齐到今天开始（凌晨 12 点或午夜），然后应用 2h 的时间偏移，结果是昨天晚上 10 点。
 - Splunk 平台始终应用偏移，然后再对齐。换句话说，先应用 @ 符号左侧再应用右侧。
- 当对齐到最早或最晚时间时，Splunk 软件始终向后对齐或向下舍入到最晚时间，而不晚于指定时间。例如，如果为 11:59:00，您要“对齐到”小时，则将对齐到 11:00 而非 12:00。
- 如果在“对齐到”时间量之前，您未指定时间偏移，Splunk 软件会将时间解释为“当前时间对齐到”指定时间量。例如，若当前的时间为周五 11:59 PM，您可以使用 @w6 “对齐至周六”，由此生成的时间为上一个周六 12:01 A.M.。

示例

1. 一直运行搜索

如果想要搜索自 UNIX 时间起的事件，使用 earliest=1。

当使用 earliest=1 且 latest=now()，搜索会一直运行。

```
...earliest=1 latest=now()
```

指定 latest=now() 不会返回未来事件。

要返回未来事件，请指定 latest=<a_big_number>。未来事件是包含晚于当前时间 now() 的时间戳的事件。

2. 从本周起始日开始搜索事件

```
earliest=@w0
```

3. 从上一完整工作周开始搜索事件

```
earliest=-5d@w1 latest=@w6
```

4. 搜索确切日期作为边界

用边界进行搜索，例如，从 11 月 15 日下午 8 点到 11 月 22 日下午 8 点，可使用时间格式 %m/%d/%Y:%H:%M:%S。

```
earliest="11/15/2017:20:00:00" latest="11/22/2017:20:00:00"
```

5. 使用固定的日期时间格式指定多个时间窗口

您可以使用时间格式 %m/%d/%Y:%H:%M:%S 指定多个时间窗口。例如：使用以下语法找出特定日期下午 5-6 点或 7-8 点的事件。

```
(earliest="1/22/2018:17:00:00" latest="1/22/2018:18:00:00") OR (earliest="1/22/2018:19:00:00" latest="1/22/2018:20:00:00")
```

6. 使用相对时间格式指定多个时间窗口

您可以使用时间调节器指定多个时间窗口并使用相对时间对齐。例如，要查找过去 24 小时内的事件，但忽略午夜到凌晨 1 点的事件，使用以下语法：

```
((earliest=-24h latest<@d) OR (earliest>=@d+1h))
```

其他时间调节器

以下搜索时间调节器仍然有效，只是在将来的版本中可能会被删除并且不再支持其功能。

修饰符	语法	描述
daysago	daysago=<int>	搜索在过去整天数内发生的事件。
enddaysago	enddaysago=<int>	设置在当前时间之前的整天数的结束时间。
endhoursago	endhoursago=<int>	设置在当前时间之前的整小时数的结束时间。
endminutesago	endminutesago=<int>	设置在当前时间之前的整分钟数的结束时间。
endmonthsago	endmonthsago=<int>	设置在当前时间之前的整月数的结束时间。
endtime	endtime=<string>	搜索指定时间之前（不包含指定时间）的事件。可使用 时间格式 来指定如何设置时间戳的格式。
endtimeu	endtimeu=<int>	搜索特定 UNIX 时间之前的事件。
hoursago	hoursago=<int>	搜索在过去整小时数内发生的事件。
minutesago	minutesago=<int>	搜索在过去整分钟数内发生的事件。
monthsago	monthsago=<int>	搜索在过去整月数内发生的事件。
searchtimespandays	searchtimespandays=<int>	在指定的天数范围内进行搜索，以整数表示。
searchtimespanhours	searchtimespanhours=<int>	在指定的小时数范围内进行搜索，以整数表示。
searchtimespanminutes	searchtimespanminutes=<int>	在指定的分钟数范围内进行搜索，以整数表示。
searchtimespanmonths	searchtimespanmonths=<int>	在指定的月数范围内进行搜索，以整数表示。
startdaysago	startdaysago=<int>	对在当前时间之前的指定天数范围进行搜索。
starthoursago	starthoursago=<int>	对在当前时间之前的指定小时数范围进行搜索。
startminutesago	startminutesago=<int>	对在当前时间之前的指定分钟数范围进行搜索。
startmonthsago	startmonthsago=<int>	对在当前时间之前的指定月数范围进行搜索。
starttime	starttime=<timestamp>	在指定日期和时间到目前为止的时间范围内进行搜索，包含指定时间。
starttimeu	starttimeu=<int>	搜索从特定 UNIX 时间开始的事件。
timeformat	timeformat=<string>	为 starttime 和 endtime 调节器设置时间格式。默认情况下： timeformat=%m/%d/%Y:%H:%M:%S

搜索命令

abstract

描述

生成搜索结果文本的摘要、汇总或简单表示。原始文本由汇总代替。

摘要由评分机制生成。即使事件大于选定的 `maxlines`，那些在相邻行上具有更多文本术语和术语的事件仍优先于术语较少的事件。如果某行包含一个搜索术语，则其邻近行也是部分匹配，并且可能会返回这些行，以提供上下文。当所选行之间存在间隙时，这些行带有省略号前缀（...）。

若某一事件的文本其行数较少，或行数与 `maxlines` 的一样，则不会发生更改。

语法

```
abstract [maxterms=<int>] [maxlines=<int>]
```

可选参数

maxterms

语法: `maxterms=<int>`

描述: 要匹配的最大术语数。接受的值是 1 到 1000。

maxlines

语法: `maxlines=<int>`

描述: 要匹配的最大行数。接受的值是 1 到 500。

示例

示例 1：显示搜索结果的汇总，每个搜索结果最多对应 5 行。

```
... |abstract maxlines=5
```

另请参阅

`highlight`

accum

描述

对 `field` 为数字的所有事件而言，`accum` 命令计算这些数字的累计值或总和。累计值既可以返回至同一个字段，也可以返回至您指定的 `newfield`。

语法

```
accum <field> [AS <newfield>]
```

必要参数

field

语法: `<string>`

描述: 待计算累计总和的字段的名称。该字段必须包含数字值。

可选参数

newfield

语法: `<string>`

描述: 要存放结果的新字段的名称。

基本示例

1. 创建一个字段的累计值

此示例使用搜索教程中的示例数据，但应该可以与任意格式的 Apache Web 访问日志配合使用。要在您自己的 Splunk 实例中尝试此示例，您必须下载示例数据并遵循操作以将教程数据导入 Splunk。当运行搜索时，请使用时间范围所有时间。

以下搜索从网页访问日志文件中查找事件，这些事件代表成功查看策略游戏的次数。此搜索按每个产品的 ID 返回事件计数。

```
sourcetype=access_* status=200 categoryId=STRATEGY | chart count AS views by productId
```

统计选项卡中显示的结果如下所示：

productId	views
DB-SG-G01	1796
DC-SG-G02	1642
FS-SG-G03	1482
PZ-SG-G05	1300

您可以使用 `accum` 命令生成查看次数的累计值，并在名为 "TotalViews" 的新字段中显示该累计值。

```
sourcetype=access_* status=200 categoryId=STRATEGY | chart count AS views by productId | accum views as TotalViews
```

统计选项卡中显示的结果如下所示：

productId	views	TotalViews
DB-SG-G01	1796	1796
DC-SG-G02	1642	3438
FS-SG-G03	1482	4920
PZ-SG-G05	1300	6220

另请参阅

`autoregress`, `delta`, `streamstats`, `trendline`

addcoltotals

描述

`addcoltotals` 命令将新的结果附加到搜索结果集的末尾。结果包含每个数字字段的总和，或者您也可以指定要汇总哪些字段。结果显示在“统计”选项卡中。若已指定了 `labelfield` 参数，则名称已指定的列将添加至统计结果表。

语法

```
addcoltotals [labelfield=<field>] [label=<string>] [<wc-field-list>]
```

可选参数

<wc-field-list>

语法：<field> ...

描述：有效字段名称的列表，以空格进行分隔。`addcoltotals` 命令仅计算您所指定列表中各字段的总和。您可以使用星号 (*) 作为通配符来指定具有相似名称的字段列表。例如，如果要指定所有以“value”开头的字段，则可以使用通配符，例如 `value*`。

默认值：计算所有字段的总和。

`labelfield`

语法：`labelfield=<fieldname>`

描述：指定要添加到结果集中的字段名称。

默认值：none

标签

语法: `label=<string>`

描述: 和 `labelfield` 参数一起使用, 即可在摘要事件中添加标签。若未使用 `labelfield` 参数, `label` 参数将不会产生任何效果。

默认值: Total

基本示例

1. 计算所有字段的总和

计算所有字段的总和, 并将总和放在名为 "change_name" 的汇总事件中。

```
... | addcoltotals labelfield=change_name label=ALL
```

2. 为两个特定字段添加列总计

为表中的两个特定字段添加列总计。

```
sourcetype=access_* | table userId bytes avgTime duration | addcoltotals bytes duration
```

3. 为与字段名称模式匹配的字段创建总计

过滤两个名称模式的字段, 和得到其中之一的总计。

```
... | fields user*, *size | addcoltotals *size
```

4. 指定列总计的字段名称

增添一个现有值总计的图表。

```
index=_internal source="metrics.log" group=pipeline | stats avg(cpu_seconds) by processor | addcoltotals labelfield=processor
```

延伸示例

1. 生成列总计

此示例使用搜索教程中的示例数据, 但应该可以与任意格式的 Apache Web 访问日志配合使用。要在您自己的 Splunk 实例中尝试此示例, 您必须下载示例数据并遵循操作以将教程数据导入 Splunk。当运行搜索时, 请使用时间范围所有时间。

以下搜索从网页访问日志文件中查找事件, 这些事件代表成功查看策略游戏的次数。此搜索按每个产品的 ID 返回事件计数。

```
sourcetype=access_* status=200 categoryId=STRATEGY | chart count AS views by productId
```

统计选项卡中显示的结果如下所示:

productId	views
DB-SG-G01	1796
DC-SG-G02	1642
FS-SG-G03	1482
PZ-SG-G05	1300

您可以使用 `addcoltotals` 命令生成查看次数的总计值, 并将总计值显示于该列的底部。

```
sourcetype=access_* status=200 categoryId=STRATEGY | chart count AS views by productId | addcoltotals
```

统计选项卡中显示的结果如下所示:

productId	views
DB-SG-G01	1796
DC-SG-G02	1642

FS-SG-G03	1482
PZ-SG-G05	1300
	6220

您可以在结果中添加一个字段来标记总计值。

```
sourcetype=access_* status=200 categoryId=STRATEGY | chart count AS views by productId | addcoltotals labelfield="Total views"
```

统计选项卡中显示的结果如下所示：

productId	views	Total views
DB-SG-G01	1796	
DC-SG-G02	1642	
FS-SG-G03	1482	
PZ-SG-G05	1300	
	6220	总计

另请参阅

命令

```
addtotals  
stats
```

addinfo

描述

将包含与搜索相关的全局通用信息的字段添加至每个事件。此命令主要是摘要索引的内部用组件。

语法

```
addinfo
```

使用 `addinfo` 命令时，下列字段将添加至所有字段。

字段	描述
info_min_time	用来限制搜索的最早时间。
info_max_time	用来限制搜索的最晚时间。
info_sid	生成事件的搜索的 ID。
info_search_time	搜索运行的时间。

用法

`addinfo` 命令属于可分配的流命令。请参阅“命令类型”。

示例

1. 添加信息到每个事件

将搜索的相关信息添加到每个事件中。

```
... | addinfo
```

2. 确定哪个检测信号比预期的晚

您可以用这个示例追踪主机上的检测信号、转发器、索引器上的 `tcpin_connection` 或任一数量的系统部件。本例使用主机。

您在名为 `expected_hosts` 的查找文件中有一个主机名称列表。您想要搜索主机上晚于预期时间范围的检测信号。您可以使用 `addinfo` 命令将信息添加到每个事件，这有助于您评估时间范围。

```
... | stats latest(_time) AS latest_time BY host | addinfo | eval latest_age = info_max_time - latest_time | fields - info_* | inputlookup append=t expected_hosts | fillnull value=9999 latest_age | dedup host | where latest_age > 42
```

使用 `stats` 命令计算主机的最晚检测信号。`addinfo` 命令会将信息添加到每个事件。此搜索将使用 `info_max_time`，这是搜索的最后时间边界。用 `eval` 命令新建名为 `latest_age` 的字段并计算关系到时间范围结束日期的检测信号时间。允许的时间范围是 `-11m@m` 到 `-m@m`。这是从分钟开始的前 11 分钟到前 1 分钟。如果您指定 `latest=null` 或所有时间，那么搜索无效，因为 `info_max_time` 可能设为无穷大。

使用查找文件 `expected_hosts` 将主机列表附加到结果。您可以使用此列表确定哪个主机未发送预期时间范围内的检测信号。对于 `latest_age` 字段中有 `null` 值的任何主机，填写该字段值为 9999。用 `dedup` 命令删除任何重复主机事件。使用 `where` 命令过滤结果并返回超出 42 秒的任何检测信号。

在这一示例中，您可以使用 `tstats` 命令（而不是 `stats` 命令）以改善搜索的性能。

另请参阅

`search`

addtotals

描述

`addtotals` 命令为每个搜索结果计算所有数字字段的算术总和。结果将显示在“统计”选项卡中。

您可指定要计算总和的字段列表，而不计算每个数字字段。总和将显示于新字段中。

若 `col=true`，则 `addtotals` 命令会计算总列数，这会在末尾添加一笔代表每个字段总和的新结果。`labelfield`，如果指定，则会将该字段添加到此汇总事件中，其值由 ‘label’ 选项设置。或者，若不使用 `addtotals col=true` 命令，您可以使用 `addcoltotals` 命令来计算摘要事件。

语法

```
addtotals [row=<bool>] [col=<bool>] [labelfield=<field>] [label=<string>] [fieldname=<field>] [<field-list>]
```

必要参数

无。

可选参数

field-list

语法：`<field> ...`

描述：一个或多个以空格分隔的数字字段。只有在 `<field-list>` 中指定的字段才计算其总和。若未指定 `<field-list>`，则会计算所有数字字段的总和。

用法：可在字段名称中使用通配符。例如，若字段名称为 `count1`、`count2` 和 `count3`，您可以指定 `count*` 来表示所有以 ‘count’ 开头的字段。

默认值：会计算所有数字字段的总和。

row

语法：`row=<bool>`

描述：指定是否为每个事件计算 `<field-list>` 的总和。类似于计算一个表格中每列的总计。总和将显示于新字段中。该字段的默认名称为 `Total`。若想为该字段指定不同的名称，请使用 `fieldname` 参数。

用法：由于默认值是 `row=true`，请指定 `row` 参数但仅限于您不希望事件总计显示 `row=false` 时。

默认值：`true`

col

语法：`col=<bool>`

描述：指定是否在事件列表底部添加新事件，也称为汇总事件。汇总事件显示事件中每个字段的总和，类似于计算一个表格中每列的总计。

默认值：`false`

fieldname

语法: `fieldname=<field>`

描述: 用于指定字段名称, 此字段包含每个事件 `field-list` 的计算总和。`fieldname` 参数仅在 `row=true` 时有效。

默认值: Total

labelfield

语法: `labelfield=<field>`

描述: 为汇总事件标签指定一个字段。`labelfield` 参数仅在 `col=true` 时有效。

* 要使用结果集中现有的字段, 请指定 `labelfield` 参数的字段名称。例如, 若字段名称为 IP, 请指定 `labelfield=IP`。

* 若您的结果集中没有字段可以与 `labelfield` 匹配, 请使用 `labelfield` 值添加新的字段。

默认值: none

标签

语法: `label=<string>`

描述: 为汇总事件指定一个行标签。

* 若 `labelfield` 参数为结果集中的现有字段, `label` 值会出现在显示的那一行中。

* 若 `labelfield` 参数新建字段, `label` 将出现在摘要事件行的新字段中。

默认值: Total

用法

`addtotals` 命令是可分配的流命令, 除非用于计算列总计。当用于计算列总计时, `addtotals` 命令属于转换命令。请参阅“命令类型”。

示例

1: 计算每个事件的所有数字字段值的总和

此示例使用列出每个产品和季度数字销量的事件, 例如:

产品	季度	销售	配额
ProductA	QTR1	1200	1000
ProductB	QTR1	1400	1550
ProductC	QTR1	1650	1275
ProductA	QTR2	1425	1300
ProductB	QTR2	1175	1425
ProductC	QTR2	1550	1450
ProductA	QTR3	1300	1400
ProductB	QTR3	1250	1125
ProductC	QTR3	1375	1475
ProductA	QTR4	1550	1300
ProductB	QTR4	1700	1225
ProductC	QTR4	1625	1350

使用图表命令汇总数据

要按产品为每个季度汇总数据, 请运行此搜索:

```
source="addtotalsData.csv" | chart sum(sales) BY products quarter
```

在此示例中, `BY` 子句中有两个通过 `chart` 命令指定的字段。

- `products` 字段指的是 `<row-split>` 字段。
- `quarter` 字段指的是 `<column-split>` 字段。

统计选项卡中显示的结果如下所示:



产品	QTR1	QTR2	QTR3	QTR4
ProductA	1200	1425	1300	1550
ProductB	1400	1175	1250	1700
ProductC	1650	1550	1375	1625

要添加为每行生成总数的列，请运行此搜索：

```
source="addtotalsData.csv" | chart sum(sales) BY products quarter | addtotals
```

统计选项卡中显示的结果如下所示：

产品	QTR1	QTR2	QTR3	QTR4	Total
ProductA	1200	1425	1300	1550	5475
ProductB	1400	1175	1250	1700	5525
ProductC	1650	1550	1375	1625	6200

使用 stats 命令计算总计

如果您所需要的只是每个产品的总计，更为简单的解决方案是使用 stats 命令：

```
source="addtotalsData.csv" | stats sum(sales) BY products
```

统计选项卡中显示的结果如下所示：

产品	总计（销售）
ProductA	5475
ProductB	5525
ProductC	6200

2. 指定包含每个事件总和的字段的名称

您可以不接受由 addtotals 命令添加的默认名称，指定字段的名称。

```
... | addtotals fieldname=sum
```

3. 使用通配符指定要计算总和的字段的名称

计算以 amount 开始或字段名中包含文本 size 的字段的总和。将总和保存在名为 TotalAmount 的字段中。

```
... | addtotals fieldname=TotalAmount amount* *size*
```

4. 计算指定字段的总和

在这一示例中，行计算关闭，列计算打开。将只计算一个字段，即 sum(quota) 的总计。

```
source="addtotalsData.csv" | stats sum(quota) by quarter| addtotals row=f col=t labelfield=quarter sum(quota)
```

- labelfield 参数指定在哪个字段中显示总计标签。默认标签为总计。

统计选项卡中显示的结果如下所示：

季度	总计（配额）
QTR1	3825
QTR2	4175
QTR3	4000

QTR4	3875
Total	15875

5. 计算字段总数并向总计添加自定义标签

计算每个季度和产品的总和并计算总计。

```
source="addtotalsData.csv" | chart sum(sales) by products quarter| addtotals col=t labelfield=products label="Quarterly Totals" fieldname="Product Totals"
```

- `labelfield` 参数指定在哪个字段中显示总计标签，在此示例中是产品。
- `label` 参数用于指定 `labelfield` 的标签季度总计，而不是使用默认标签总计。
- `fieldname` 参数用于指定总计行数的标签产品总计。

统计选项卡中显示的结果如下所示：

产品	QTR1	QTR2	QTR3	QTR4	产品总计
ProductA	1200	1425	1300	1550	5475
ProductB	1400	1175	1250	1700	5525
ProductC	1650	1550	1375	1625	6200
季度总计	4250	4150	3925	4875	17200

另请参阅

`stats`

analyzefields

描述

将 `field` 作为离散随机变量，此命令将分析所有数字字段来决定这些字段中的每一个字段 `predict classfield` 值的能力。它将决定目标 `classfield` 中的值与其他字段中数字值之间关系的稳定性。

作为一个报表命令，`analyzefields` 使用所有输入结果，并为输入结果中的每个数字字段生成一行。该行中的各值表示 `analyzefields` 命令预测 `classfield` 的值的性能。对于每个事件，如果基于最高 z 概率的数字字段的条件分布符合实际类别，则此事件视为准确。最高的 z 概率基于 `classfield`。

语法

```
analyzefields classfield=<field>
```

您可以使用缩写 `af` 代表 `analyzefields` 命令。

`analyzefields` 命令将返回一个有五列的表格。

字段	描述
<code>field</code>	输入搜索结果中数字字段的名称。
<code>count</code>	搜索结果中该字段的出现次数。
<code>cocur</code>	该字段的同现。在包含 <code>classfield</code> 的结果中，此为该结果中亦包含 <code>field</code> 的比例。 <code>cocur</code> 为 1，前提是 <code>field</code> 存在于所有包含 <code>classfield</code> 的事件中。
<code>acc</code>	使用字段中的值预测 <code>classfield</code> 值的准确度。此为准确预测与包含 <code>field</code> 的事件总数之间的比例。此参数仅适用于数字字段。
<code>balacc</code>	平衡准确度为预测每个 <code>classfield</code> 值的准确度未加权值。仅适用于数值字段。

必要参数

classfield

语法: classfield=<field>

描述: 为了获得最佳结果, 尽管可以进行多类分析, classfield 仍应有两个唯一值。

示例

示例 1:

分析数值字段, 以预知 "is_activated" 的值。

```
... | analyzefields classfield=is_activated
```

另请参阅

anomalousvalue

anomalies

描述

使用 anomalies 命令查找异常或意外的事件或字段。

anomalies 命令为每个事件分配一个 unexpectedness 分数, 并将该分数存放于名为 unexpectedness 的新字段中。是否将该事件视为异常取决于 threshold 值。将 threshold 值与 unexpectedness 分数进行比较。若 unexpectedness 分数大于 threshold 值, 则事件被视为意外或异常。

在搜索中使用 anomalies 命令后, 在“搜索和报表”窗口中查找感兴趣的字段列表。选择 unexpectedness 字段即可查看事件中各值的相关信息。

一个事件的 unexpectedness 分数是基于该事件 (X) 与先前事件集 (P) 的相似度进行计算的。

计算 unexpectedness 分数的公式为:

$$\text{unexpectedness} = [s(P \text{ and } X) - s(P)] / [s(P) + s(X)]$$

在此公式中, $s(\cdot)$ 是衡量数据相似度或一致度的指标。此公式提供了一种方式来计算添加的 X 对事件集相似度的影响程度, 同时还将对不同的事件大小进行规范化。

语法

要求的语法以粗体表示。

```
anomalies
[threshold=<num>]
[labelonly=<bool>]
[normalize=<bool>]
[maxvalues=<num>]
[field=<field>]
[denylist=<filename>]
[denylistthreshold=<num>]
[by-clause]
```

可选参数

threshold

语法: threshold=<num>

描述: 代表预期或正常事件上限值的数字。如果某个事件计算出的 unexpectedness 值大于此阈值, 则将此事件视为意外或异常事件。

默认值: 0.01

labelonly

语法: labelonly=<bool>

描述: 指定输出结果集是包含所有事件, 还是仅包含超过阈值的事件。unexpectedness 字段会附加到所有事件。若 labelonly=true, 则不移除任何事件。若 labelonly=false, 则会从输出结果集中移除 unexpectedness 分数低于阈值的事件。

默认值: false

normalize
语法: normalize=<bool>
描述: 指定是否将字段中的数字文本规范化。用于算法时，字段中从 0 到 9 的所有字符都认为是相同的。尽管数字的位置和数量仍然有明显区别。若一字段中包含不应进行规范化而应被视为类别的数值型数据，请设置 normalize=false。
默认值: true

maxvalues
语法: maxvalues=<num>
描述: 指定在确定字段值的 unexpectedness 时要包括的先前事件的滑动事件集的大小。默认情况下，此计算采用先前的 100 次事件做对比。如果当前事件编号为 1,000，则此计算采用第 900 至 999 号事件的值。如果当前事件编号为 1,500，则此计算采用第 1,400 至 1,499 号事件的值。您可以指定介于 10 至 10,000 之间的一个编号。调高 maxvalues 的值会线性增加每个事件的 CPU 总成本。大的数值会占用很长的搜索运行时间。
默认值: 100

field
语法: field=<field>
描述: 在确定事件的 unexpectedness 时要分析的字段。
默认值: _raw

denylist
语法: denylist=<filename>
描述: 包含需忽略的预期事件列表的 CSV 文件名。与 denylist 中的事件相似的传入事件将被视为非异常事件，或预期，并为其分配 unexpectedness 分数 0.0。CSV 文件必须位于搜索头上的 \$SPLUNK_HOME/var/run/splunk/ 目录中。如果您用的是 Splunk Cloud 而且想要配置拒绝列表文件，请提交支持工单。

denystthreshold
语法: denystthreshold=<num>
描述: 指定相似度分数阈值，以将传入事件与拒绝列表中的事件匹配。若传入事件的相似度分数大于 denystthreshold，则该事件将被标记为意外事件。
默认值: 0.05

by-clause
语法: by <fieldlist>
描述: 用于指定一个字段列表，以隔离异常检测的结果。对于每个指定字段值的组合，将以完全隔离的方式对待具有这些值的事件。

示例

1. 指定要忽略的事件的拒绝列表文件

以下示例显示有趣事件，忽略拒绝列表“boring events”（无趣事件）中的所有事件。以降序排列事件列表，unexpectedness 字段中的值最高者优先列出。

```
... | anomalies denylist=boringevents | sort -unexpectedness
```

2. 找到交易中的异常

此示例使用交易找到看起来不寻常的时间范围。

```
... | transaction maxpause=2s | anomalies
```

3. 按数据来源识别异常

单独在每个数据来源中寻找异常。一个数据来源中的一种模式不影响它在另一个数据来源中异常的事实。

```
... | anomalies by source
```

4. 识别异常时指定阈值

这一示例显示如何使用 threshold 值调整对异常的搜索。从使用默认 threshold 值的搜索开始。

```
index=_internal | anomalies BY group | search group=*
```

此搜索查找 _internal 索引中的事件，并计算具有相同 group 值的事件集的 unexpectedness 分数。

- 用于计算每个唯一 group 值的 unexpectedness 分数的滑动事件集仅包括有着相同 group 值的事件。
- search 命令用于显示只包括 group 字段的事件。

unexpectedness 和 group 字段出现在感兴趣的字段列表中。单击字段名称，然后单击是，将字段移动到已选字段列表中。字段移动，还出现在搜索结果中。结果应该看起来像下图所示。

列表 ▾ 格式 每页 20 个 ▾		
隐藏字段	所有字段	i 时间 事件
选定字段		> 18/05/31 05-31-2018 15:31:58.524 +0100 INFO Metrics - group=pipeline, name=indexerpipe, processor=r=indexin, cpu_seconds=0.132, executes=29201, cumulative_hits=4369780 group = pipeline host = SplunkMaster source = /opt/splunk/var/log/splunk/metrics.log sourcetype = splunkd unexpectedness = 0.010526
# group 4 # host 1 # source 1 # sourcetype 1 # unexpectedness 26		> 18/05/31 05-31-2018 15:28:21.526 +0100 INFO Metrics - group=per_sourcetype_thruput, series="splunk_resource_usage", kbps=0, eps=0.446202040902832, eps=1.03226765461563, kb=13.8330078125, ev=32, avg_age=1.125, max_age=3 group = per_sourcetype_thruput host = SplunkMaster source = /opt/splunk/var/log/splunk/metrics.log sourcetype = splunkd unexpectedness = 0.011737
感兴趣的字段		> 18/05/31 05-31-2018 15:27:50.527 +0100 INFO Metrics - group=per_sourcetype_thruput, series="fs_notification", kbps=0, eps=0.032260009482507186, kb=0, ev=1, avg_age=0, max_age=0 group = per_sourcetype_thruput host = SplunkMaster source = /opt/splunk/var/log/splunk/metrics.log sourcetype = splunkd unexpectedness = 0.024706
# avg_age 14 # component 1 # cpu_seconds 6 # date_hour 1 # date_mday 1 # date_minute 7 # date_month 1 # date_second 9 # date_wday 1 # date_year 1 # date_zone 1 # eps 14 # ev 14 # executes 6 # index 1 # kb 14 # kbs 14		> 18/05/31 05-31-2018 15:27:50.526 +0100 INFO Metrics - group=per_source_thruput, series="fschange_monitor", kbps=0, eps=0.032260009482507186, kb=0, ev=1, avg_age=0, max_age=0 group = per_source_thruput host = SplunkMaster source = /opt/splunk/var/log/splunk/metrics.log sourcetype = splunkd unexpectedness = 0.028802
		> 18/05/31 05-31-2018 15:27:50.526 +0100 INFO Metrics - group=per_source_thruput, series="/opt/splunk/var/log/splunkd.log", kbps=214.5017806678409, eps=935.2821949168483, kb=6649.

首个事件中的键值对包括 group=pipeline、name=indexerpipe、processor=indexer、cpu_seconds=0.022 等。

借助默认的 threshold，即 0.01，您可以看到有些事件可能非常相似。下一个搜索将稍微提高 threshold：

```
index=_internal | anomalies threshold=0.03 by group | search group=*
```

列表 ▾ 格式 每页 20 个 ▾		
隐藏字段	所有字段	i Time Event
选定字段		> 3/16/18 03-16-2018 22:54:24.180 -0700 INFO Metrics - group=pipeline, name=fschangemanager, processor=fschangemanager, cpu_seconds=0, executes=1, cumulative_hits=71 group = pipeline host = docs-unix-17f source = /opt/splunk-nightlight/var/log/splunk/metrics.log sourcetype = splunkd unexpectedness = 0.031949
# group 9 # host 1 # source 1 # sourcetype 1 # unexpectedness 91		> 3/16/18 03-16-2018 22:44:04.181 -0700 INFO Metrics - group=pipeline, name=fschangemanager, processor=fschangemanager, cpu_seconds=0, executes=1, cumulative_hits=78 group = pipeline host = docs-unix-17f source = /opt/splunk-nightlight/var/log/splunk/metrics.log sourcetype = splunkd unexpectedness = 0.030794
感兴趣的字段		> 3/16/18 03-16-2018 22:29:05.182 -0700 INFO Metrics - group=thruput, name=idxsummary, series="_audit", ev=668, kb=92.7470703125, st_count=1, h_count=1, s_count=1, min_time=1521269191, max_time=1521264249 group = thruput host = docs-unix-17f source = /opt/splunk-nightlight/var/log/splunk/metrics.log sourcetype = splunkd unexpectedness = 0.085174
# component 1 # cpu_seconds 1 # cumulative_hits 85 # date_hour 13 # date_mday 1 # date_minute 45 # date_month 1 # date_second 55 # date_wday 1 # date_year 1 # date_zone 1 # executes 8 # index 1 # linecount 1 # log_level 1 # message 100+		> 3/16/18 03-16-2018 22:23:55.180 -0700 INFO Metrics - group=search_concurrency, user=splunk-system-user, active_hist_searches=2, active_realtime_searches=0 group = search_concurrency host = docs-unix-17f source = /opt/splunk-nightlight/var/log/splunk/metrics.log sourcetype = splunkd unexpectedness = 0.032184
		> 3/16/18 03-16-2018 22:22:53.180 -0700 INFO Metrics - group=pipeline, name=fschangemanager, processor=fschangemanager, cpu_seconds=0, executes=1, cumulative_hits=68 group = pipeline host = docs-unix-17f source = /opt/splunk-nightlight/var/log/splunk/metrics.log sourcetype = splunkd unexpectedness = 0.031447

通过使用更高的 threshold 值，时间戳和键值显示每个事件之间的更多区别。

另外，您可能不想隐藏非异常事件。您可以在事件中再添加一个字段，用于显示您是否对此事件感兴趣。一种方式是使用 eval 命令来实现：

```
index=_internal | anomalies threshold=0.03 labelonly=true by group | search group=* | eval threshold=0.03 | eval score;if(unexpectedness>=threshold, "anomalous", "boring")
```

此搜索使用 labelonly=true，以便让干扰事件继续保留在结果列表中。eval 命令用于定义名为 threshold 的字段，并将其设置为阈值。此操作必须显式完成，因为 threshold 属性（属于 anomalies 命令）不是一个字段。

第二个 eval 命令用于定义另一个新字段 score；根据 unexpectedness 与 threshold 值的比较结果，该新字段将被判断为“异常”或“干扰”。下图显示结果的截图。

列表 ▾			格式	每页 20 个 ▾	< 上一个	1	2	3	4	5	6	7	8	... 下一步 >
<隐藏字段 选定字段 <code>a group 26</code> <code>a host 1</code> <code>a score 2</code> <code>a source 1</code> <code>a sourcetype 1</code> <code># unexpectedness 100+</code> 感兴趣的字段 <code>a component 1</code> <code># current_size 6</code> <code># date_hour 1</code> <code># date_mday 1</code> <code># date_minute 2</code> <code># date_month 1</code> <code># date_second 2</code> <code># date_wday 1</code> <code># date_year 1</code> <code># date_zone 1</code> <code># ev 39</code> <code>a index 1</code> <code># kb 41</code> <code># linecount 1</code> <code>a log_level 1</code> <code>a message 100+</code>	i	时间	事件											

另请参阅

`anomalousvalue`, `cluster`, `kmeans`, `outlier`

anomalousvalue

描述

`anomalousvalue` 命令为所有事件的每个字段计算异常分数（相对于其他事件中此字段的值而言）。对于数值字段，它会根据出现频率或与平均值的标准偏差数来发现或汇总数据中的异常值。

对于确定为异常的字段，会在下面的方案中添加一个新字段。若此字段为数字，例如 `size`，则该新字段将为 `Anomaly_Score_Num(size)`。若此字段并非数字，例如 `name`，则该新字段将为 `Anomaly_Score_Cat(name)`。

语法

`anomalousvalue <av-options>... [action] [pthresh] [field-list]`

必要参数

无。

可选参数

`<av-options>`

语法： `minsupcount=<int> | maxanofreq=<float> | minsupfreq=<float> | minnormfreq=<float>`
描述： 指定一个或多个选项，以控制考虑哪些字段以判别异常。

av-option 参数的描述

`maxanofreq`

语法： `maxanofreq=<float>`

描述： 最大异常频率以 0 到 1 之间的一个浮点值来表示。如果某个字段太频繁地出现异常，则省略该字段。若该字段的异常频率与其总频率之间的比例大于 `maxanofreq` 值，则不考虑该字段。

默认值： 0.05

`minnormfreq`

语法： `minnormfreq=<float>`

描述： 最小正常频率以 0 到 1 之间的一个浮点值来表示。如果某个字段未经常性出现异常，则省略该字段。如果字段发生异常的次数与字段出现的总数之间的比率小于 ρ ，则不考虑该字段。

默认值： 0.01

`minsupcount`

语法： `minsupcount=<int>`

描述：最小支持计数必须是正整数。丢弃在输入结果集中出现次数很少的字段。如果在输入事件中字段出现的次数小于 N 次，则不考虑该字段。
默认值：100

minsupfreq

语法：minsupfreq=<float>

描述：最小支持频率以 0 到 1 之间的一个浮点值来表示。丢弃出现频率很低的字段。minsupfreq 参数将检查该字段出现的频率与事件总数之间的比例。如果该比率小于 ρ ，则不考虑该字段。

默认值：0.05

action

语法：action=annotate | filter | summary

描述：指定是返回异常分数 (annotate)，筛选出非异常值的事件 (filter)，还是返回异常统计信息 (summary)。
默认值：filter

操作参数的描述

annotate

语法：action=annotate

描述：annotate 操作把新字段添加到包含异常值的事件。添加的字段为 Anomaly_Score_Cat(field) 或 Anomaly_Score_Num(field)，也可以两个字段都添加。

过滤器

语法：action=filter

描述：filter 操作返回含异常值的事件。未包含异常值的事件会被删除。如 action=annotate 中所述，返回的事件都会添加批注。

summary

语法：action=summary

描述：summary 操作返回一个表格，其中汇总了每个生成字段的异常统计信息。此表格包含该字段所含的事件个数、占异常事件的比例、所进行的测试类型（类别型或数字型）等。

输出字段	描述
fieldname	字段名。
count	字段出现的次数。
distinct_count	字段中唯一值的数量。
mean	字段值的计算平均值。
catAnoFreq%	类别字段的异常频率。
catNormFreq%	类别字段的正常频率。
numAnoFreq%	数字字段的异常频率。
stdev	字段值的标准偏差。
supportFreq%	字段的支持频率。
useCat	使用类别异常检测。类别异常检测用于查找罕见值。
useNum	使用数字异常检测。数字异常检测用于检测与平均值相差很大的值。此异常检测基于高斯分布。
isNum	无论该字段是否为数字型。

field-list

语法：<field> ...

描述：要考虑的字段的“列表”。

默认值：如果未提供字段列表，则考虑所有字段。

pthresh

语法：pthresh=<num>

描述：将值视为异常所必须达到的可能性阈值（十进制数）。

默认值：0.01。

用法

默认情况下，最多返回 50,000 个结果。此最大值受 limits.conf 文件中 [anomalousvalue] 段落的 maxresultrows 设置控制。

提高此限制会导致占用更多内存。

只有有着文件系统访问权限的用户，如系统管理员，才能编辑配置文件。不要更改或复制默认目录中的配置文件。默认目录中的文件必须保持原样并位于其原始位置。在本地目录进行更改。

请参阅“如何编辑配置文件”。

基本示例

1. 仅返回搜索结果中不常见的值

```
... | anomalousvalue
```

这与运行以下搜索相同：

```
...| anomalousvalue action=filter pthresh=0.01
```

2. 返回来自主机 “reports”的不常见值

```
host="reports" | anomalousvalue action=filter pthresh=0.02
```

延伸示例

1. 返回每个数字字段的异常统计信息的汇总

此搜索使用从 USGS 地震网站下载的近期地震数据。该数据是一个逗号分隔的 ASCII 文本文件，其中包含每次记录的地震的震级（mag）、坐标（经度、纬度）、区域（地点）等。

您可以从 **USGS 地震源** 下载当前 CSV 文件，然后将文件上载到 Splunk 实例。此示例使用过去 30 天的所有地震数据。

搜索地震数据中的异常值。

```
source="all_month.csv" | anomalousvalue action=summary pthresh=0.02 | search isNum=YES
```

事件 (9,890)	模式	统计信息 (21)	可视化										
每页 20 个	格式	预览	< 上一个 1 2 下一步 >										
catAnoFreq%	catNormFreq%	count	distinct_count	fieldname	isNum	mean	numAnoFreq%	stdev	supportFreq%	useCat	useNum		
0.0000	4.1667	9890	24	date_hour	YES	11.184833	0.0000	6.946614	100.0000	YES	YES		
0.0000	4.0000	9890	25	date_mday	YES	16.585743	0.0000	6.946190	100.0000	YES	YES		
0.0000	1.6667	9890	60	date_minute	YES	29.251668	0.0000	17.407434	100.0000	YES	YES		
0.0000	1.6667	9890	60	date_second	YES	29.072700	0.0000	17.203107	100.0000	YES	YES		
0.0000	100.0000	9890	1	date_year	YES	2018.000000	0.0000	0.000000	100.0000	NO	NO		
0.0000	100.0000	9890	1	date_zone	YES	0.000000	0.0000	0.000000	100.0000	NO	NO		
0.0000	0.0376	9890	2660	depth	YES	15.168215	3.9939	36.531770	100.0000	NO	YES		
0.0000	0.1553	9841	644	depthError	YES	2.094915	3.1244	5.520545	99.5046	NO	YES		
0.0000	0.0168	7873	5941	dmin	YES	0.328978	1.7998	1.497277	79.6057	NO	YES		
0.0000	0.0984	7887	1016	gap	YES	118.472556	3.5693	68.065646	79.7472	NO	YES		
0.0000	0.2000	7189	500	horizontalError	YES	1.247435	4.5703	2.489247	72.6896	NO	YES		
0.0000	0.0132	9890	7597	latitude	YES	35.184659	2.2851	17.931700	100.0000	NO	YES		

返回带多个小数位的数字结果。使用字段格式图标（形状像铅笔）以启用数字格式设置，指定要显示的小数精确度。

事件 (9,890) 模式 统计信息 (21) 可视化

每页 20 个 ▾ 格式 预览 ▾

catAnoFreq%	catNormFreq%	distinct_count	x	numAnoFreq%	stdev
0.0000	4.17			0.0000	6.946614
0.0000	4.00			0.0000	6.946190
0.0000	1.67			0.0000	17.407434
0.0000	1.67			0.0000	17.203107
0.0000	100.00			0.0000	0.000000
0.0000	100.00			0.0000	0.000000
0.0000	0.04			3.9939	36.531770
0.0000	0.16	9841	644 depthError	YES	2.094915
0.0000	0.02	7873	5941 dmin	YES	0.328978
0.0000	0.10	7887	1016 gap	YES	118.472556
0.0000	0.20	7189	500 horizontalError	YES	1.247435
0.0000	0.01	9890	7597 latitude	YES	35.184659
				2.2851	17.931700

另请参阅

`analyzefields`, `anomalies`, `cluster`, `kmeans`, `outlier`

anomalydetection

描述

这是一种流式和报表命令，通过计算每个事件的概率并检测异常小的概率来识别异常事件。可能性定义为事件中每个独立字段值的频率之乘积。

- 对于类别字段，值 X 的频率是指出现次数 X 除以总事件数。
- 对于数字字段，首先会为所有值构建一个直方图，然后会计算值 X 的频率，即包含 X 的数据桶的大小除以总事件数。

`anomalydetection` 命令包含现有 `anomalousvalue` 和 `outlier` 命令的功能，并提供一种基于直方图的异常检测方法。

语法

`anomalydetection [<method-option>] [<action-option>] [<pthresh-option>] [<cutoff-option>] [<field-list>]`

可选参数

<method-option>

语法: `method = histogram | zscore | iqr`

描述: 选择一种异常检测方式。`method=zscore` 时，像 `anomalousvalue` 命令一样运行。`method=iqr` 时，像 `outlier` 命令一样运行。请参阅“用法”。

默认值: `method=histogram`

<action-option>

`Method=histogram` 或 `method=zscore` 的语法: `action = filter | annotate | summary`

`Method=iqr` 的语法: `action = remove | transform`

描述: 操作项和默认值取决于您指定的方法。所有方法各项操作的详细描述请参阅下文。

<pthresh-option>

语法: `pthresh=<num>`

描述: 与 `method=histogram` 或 `method=zscore` 一起使用。将值视为异常所必须达到的可能性阈值（十进制数）。

默认值: 若 `method=histogram`，命令将在分析期间计算每个数据集的 `pthresh`。对于 `method=zscore`，默认值是 0.01。若您在 `method=iqr` 时使用，将返回一个无效参数错误。

<cutoff-option>

语法: `cutoff=<bool>`

描述: 设置异常数量的上限阈值。此选项仅应用于直方图方法。若 `cutoff=false`, 算法将使用没有修改过的公式 `threshold = 1st-quartile - 1.5 * IRQ`。若 `cutoff=true`, 算法将修改公式, 以减少异常的数量。

默认值: true

<field-list>

语法: <string> <string> ...

描述: 字段名称的列表。

直方图操作

<action-option>

语法: `action=annotate | filter | summary`

描述: 指定是要返回含额外字段的所有事件 (annotate), 过滤出具有异常值的事件 (filter), 还是汇总异常统计信息 (summary)。

默认值: filter

若 `action=filter`, 命令将返回异常事件并过滤掉其他事件。每一个返回的事件都包含四个新字段。若 `action=annotate`, 命令将返回所有原始事件, 这些事件在 `action=filter` 时添加了相同的四个新字段。

字段	描述
<code>log_event_prob</code>	事件可能性的自然对数。
<code>probable_cause</code>	最清楚的解释了事件异常原因的字段名称。字段本身不会造成异常, 而往往是因为有些字段值很少出现, 从而使得事件可能性变得很低。
<code>probable_cause_freq</code>	<code>probable_cause</code> 字段值的频率。
<code>max_freq</code>	事件中所有字段值的最大频率。

若 `action=summary`, 命令将返回包含六个字段的单一事件。

输出字段	描述
<code>num_anomalies</code>	异常事件的数量。
<code>thresh</code>	分隔异常事件的事件概率阈值。
<code>max_logprob</code>	所有日志的最大值 (<code>event_prob</code>)。
<code>min_logprob</code>	所有日志的最小值 (<code>event_prob</code>)。
<code>1st_quartile</code>	所有日志的第一四分位值 (<code>event_prob</code>)。
<code>3rd_quartile</code>	所有日志的第三四分位值 (<code>event_prob</code>)。

Zscore 操作

<action-option>

语法: `action=annotate | filter | summary`

描述: 指定是要返回异常分数 (annotate), 过滤出具有异常值的事件 (filter), 还是汇总异常统计信息 (summary)。

默认值: filter

若 `action=filter`, 命令将返回包含异常值的事件, 并丢弃其他事件。为保留的事件添加批注, 如 `annotate` 操作。

若 `action=annotate`, 命令将把新字段 `Anomaly_Score_Cat(field)` 和 `Anomaly_Score_Num(field)` 添加到包含异常值的事件。

若 `action=summary`, 命令将返回一个表格, 其中汇总了每个生成字段的异常统计信息。此表格包含该字段所含的事件个数、占异常事件的比例、所进行的测试类型 (类别型或数字型) 等。

IQR 操作

<action-option>

语法: `action=remove | transform`

描述: 指定如何处理离群值。`remove` 操作将移除包含无关数字值的事件。`transform` 操作通过把无关值截成离群值阈值来转换事件。若 `mark=true`, `transform` 操作将在值前添加前缀 "000"。

缩写: `remove` 的缩写为 `rm`。`transform` 的缩写为 `tf`。

默认值: `action=transform`

用法

anomalydetection 命令是一个流命令。请参阅“命令类型”。

zscore 方法

指定 method=zscore 时，anomalydetection 命令像 anomalousvalue 命令一样实施。将 anomalydetection 命令与 method=zscore 一起使用时，您可以指定 anomalousvalue 命令的语法组件。请参阅 anomalousvalue 命令。

iqr 方法

指定 method=iqr 时，anomalydetection 命令像 outlier 命令一样实施。当您用 anomalydetection 命令指定 method=iqr 时，您可以指定 outlier 命令的语法组件。例如：您可以指定离群值选项 <action>、<mark>、<param> 和 <uselower>。请参阅离群值命令。

示例

示例 1：仅返回异常事件

这两个搜索返回相同结果。第二个搜索中指定的参数为默认值。

```
... | anomalydetection  
... | anomalydetection method=histogram action=filter
```

示例 2：返回一个简短摘要，说明异常事件的个数

返回说明异常事件个数的简短摘要，以及其他部分统计数据，如用于检测的阈值。

```
... | anomalydetection action=summary
```

示例 3：返回包含异常值的事件

这一示例指定 method=zscore 以返回异常值。搜索使用 filter 操作以筛选没有异常值的事件。事件必须在被认为是异常值之前必须符合可能性阈值 pthresh。

```
... | anomalydetection method=zscore action=filter pthresh=0.05
```

示例 4：返回离群值

此示例使用 outlier 命令中的离群值选项。本示例中，缩写 tf 用于转换操作。

```
... | anomalydetection method=iqr action=tf param=4 uselower=true mark=true
```

另请参阅

analyzefields, anomalies, anomalousvalue, cluster, kmeans, outlier

append

描述

将子搜索结果附加到当前结果。append 命令仅对历史数据有效，若用于实时搜索则不会产生正确的结果。

有关何时使用 append 命令的更多信息，请参阅《搜索手册》中的“有关事件分组和相关性”主题中的流程图。

如果您对 SQL 很熟悉，但对 SPL 较为陌生，请参阅“面向 SQL 用户的 Splunk SPL”。

语法

```
append [<subsearch-options>...]<subsearch>
```

必要参数

subsearch

语法: [subsearch]

描述: 一种辅助搜索，可用于指定要附加的事件源。子搜索必须用方括号括起来。请阅读《搜索手册》中的“关于子搜索”。

可选参数

subsearch-options

语法: extendtimerange=<boolean> | maxtime=<int> | maxout=<int> | timeout=<int>

描述: 控制子搜索的处理方式。

子搜索选项

extendtimerange

语法: extendtimerange=<boolean>

描述: 指定是否将子搜索时间范围包含在整个搜索的时间范围内。当子搜索的时间范围超过主搜索的时间范围之后，使用 extendtimerange 参数。当转换命令（如 chart、timechart 或 stats）跟在搜索中的 append 命令之后，且搜索使用基于时间的数据箱时，使用此参数。

默认值: false

maxtime

语法: maxtime=<int>

描述: 在自动完成之前，子搜索最多使用的时间，单位为秒。

默认值: 60

maxout

语法: maxout=<int>

描述: 从子搜索中输出的结果行的最大数量。

默认值: 50000

timeout

语法: timeout=<int>

描述: 等待子搜索完全完成的最长等待时间，单位为秒。

默认值: 60

用法

append 命令是一个流命令。请参阅“命令类型”。

示例

1: 使用 append 命令添加列总计。

此搜索使用从 USGS 地震网站下载的近期地震数据。该数据是一个逗号分隔的 ASCII 文本文件，其中包含每次记录的地震的震级 (mag)、坐标 (经度、纬度)、区域 (地点) 等。

您可以从 USGS 地震源下载当前 CSV 文件，然后将文件上载到 Splunk 实例。此示例使用过去 30 天的所有地震数据。

统计昨天在 California 市区和周边发生的地震数量，然后计算地震的总数。

```
source=usgs place=*California* | stats count by magType | append [search index=usgs_* source=usgs place=*California* | stats count]
```

此示例使用子搜索对加利福尼亚地区的所有地震进行计数 (place="*California")，然后使用主要搜索根据搜索的震级类型对地震的数量进行计数。

stats 命令无法同时统计事件的总数量和指定字段的事件总数。子搜索计算已发生的地震的总数。使用 append 命令可将统计结果添加到前一个搜索的结果中。

由于两个搜索都使用 count 字段，子搜索的结果将在计数列的最后一行中列出。

统计选项卡中显示的结果如下所示：

magType	count
H	123
MbLg	1

Md	1565
Me	2
Ml	1202
Mw	6
ml	10

此搜索演示如何使用 `append` 命令，以类似于使用 `addcoltotals` 命令的方法添加列总数。

2. 计算购买项目的不同客户的数量。附加每种产品类型购买数量最大的顾客。

此示例使用搜索教程中的示例数据。要在您自己的 Splunk 实例中尝试此示例，您必须下载示例数据并遵循操作以将教程数据导入 Splunk。当运行搜索时，请使用时间范围昨天。

统计昨天在 Buttercup Games 网上商店购物的不同客户的人数，然后再根据客户购买的产品类型（配饰、T 恤和游戏类型）细分人数统计结果。此外，还将列出每种产品购买数量最大的顾客以及这位顾客购买了多少该产品。

```
sourcetype=access_* action=purchase | stats dc(clientip) BY categoryId | append [search sourcetype=access_* action=purchase | top 1 clientip BY categoryId] | table categoryId, dc(clientip), clientip, count
```

本示例首先搜索购物事件 (`action=purchase`)。然后将搜索结果通过管道符传递给 `stats` 命令，再使用 `dc()` 或 `distinct_count()` 函数统计在店内购物的不同用户的数量。运用 `BY` 子句根据不同的产品类别 (`categoryId`) 细分用户数量。

本示例包含了作为 `append` 命令参数的子搜索。

```
...[search sourcetype=access_* action=purchase | top 1 clientip BY categoryId]
```

运用子搜索来搜索购物事件，并根据 `clientip` 统计每个产品类型的最大客户。使用 `append` 命令可将这些结果添加到前一个搜索的结果中。

此示例中，`table` 命令仅用于显示产品类别 (`categoryId`)、每种产品类型的购买用户的不同统计结果 (`dc(clientip)`)、某种产品类型的最大实际购买用户 (`clientip`) 以及该用户购买的每种产品的数量 (`count`)。

categoryId	dc(clientip)	clientip	count
ACCESSORIES	37		
ARCADE	58		
NULL	8		
SHOOTER	31		
SIMULATION	34		
SPORTS	13		
STRATEGY	74		
TEE	38		
ACCESSORIES	91.208.184.24		3
ARCADE	211.166.11.101		4
NULL	87.194.216.51		2
SHOOTER	87.194.216.51		2
SIMULATION	211.166.11.101		2
SPORTS	95.163.78.227		1
STRATEGY	76.169.7.252		3
TEE	87.194.216.51		2

您会发现，尽管字段值相同，`append` 命令仅将子搜索的结果添加到前一个搜索的末尾。它不允许您对输出进行操作或重新进行格式设置。

3. 使用 `append` 命令，确定访问此 Web 服务器的唯一 IP 地址的数量。

使用 `append` 命令，以及 `stats`、`count` 和 `top` 命令，确定访问此 Web 服务器的唯一 IP 地址的数量。找到每种类型的页面请求中，最常访问此 Web 服务器的用户。

此示例使用搜索教程中的示例数据。要在您自己的 Splunk 实例中尝试此示例，您必须下载示例数据并遵循操作以将教程数据导入 Splunk。当运行搜索时，请使用时间范围昨天。

统计访问 Web 服务器的不同 IP 地址的数量，同时还要找出就每种页面请求类型而言访问 Web 服务器最多的用户 (`method`)。

```
sourcetype=access_* | stats dc(clientip), count by method | append [search sourcetype=access_* | top 1 clientip by method]
```

Web 访问事件通过管道符传递给 `stats` 命令，再使用 `dc()` 或 `distinct_count()` 函数统计访问该站点的不同用户的数量。使用 `count()` 函数统计访问该站点的总次数。这些数量由页面请求进行分隔 (`method`)。

利用子搜索找出使用每种页面请求类型最频繁的用户 (`method`)。使用 `append` 命令将子搜索的结果添加到表格底部。

统计选项卡中显示的结果如下所示：

方法	dc(clientip)	count	clientip	百分比
GET	173	2666		
POST	168	1727		
GET		83	87.194.216.51	3.113278
POST		64	87.194.216.51	3.705848

前两行是第一个搜索的结果。最后两行是子搜索的结果。两个结果集共用 `method` 和 `count` 字段。

4. 指定子搜索运行的最长时间和子搜索结果行的最大数量

使用 `append` 命令，确定访问此 Web 服务器的唯一 IP 地址的数量。找到每种类型的页面请求中，最常访问此 Web 服务器的用户。

此示例使用搜索教程中的示例数据，但应该可以与任意格式的 Apache Web 访问日志配合使用。从搜索教程中的本主题下载数据集，并按照说明将其上载到 Splunk 部署。当运行此搜索时，请使用时间范围昨天。

统计访问 Web 服务器的不同 IP 地址的数量，同时还要找出就每种页面请求类型而言访问 Web 服务器最多的用户 (`method`)。将子搜索限制为 30 秒，子搜索结果的最大数量为 1000。

```
sourcetype=access_* | stats dc(clientip), count by method | append maxtime=30 maxout=1000 [search sourcetype=access_* | top 1 clientip by method]
```

5. 使用 `extendtimerange` 参数

使用 `extendtimerange` 参数确保用于搜索的时间范围同时包括主搜索和子搜索的时间范围。

```
index=_internal earliest=11/20/2017:00:00:00 latest=11/30/2017:00:00:00 | append extendtimerange=true [search index=_audit earliest=11/1/2017:00:00:00 latest=11/25/2017:00:00:00] | timechart span=1d count
```

用于搜索的时间范围从子搜索的最早时间 11/1/2017:00:00:00 开始，到主搜索的最晚时间 11/30/2017:00:00:00 结束。

另请参阅

`appendcols`、`appendpipe`、`join`、`set`

`appendcols`

描述

将子搜索结果的字段附加到输入搜索结果中。以非下划线字符（_）开头的子搜索的外部字段不会计入当前结果。第一个子搜索结果与第一个主结果合并，第二个子搜索结果与第二个主结果合并，依此类推。

语法

```
appendcols [override= <bool> | <subsearch-options>...]<subsearch>
```

必要参数

subsearch

描述：添加到主搜索的辅助搜索。请参阅《[搜索手册](#)》中的“子搜索如何工作”以了解更多信息。

可选参数

override

语法：override=<bool>

描述：若 override 参数为 false，且若一字段同时出现在子搜索结果和主搜索结果中，则将使用主搜索结果。若 override=true，则将使用子搜索结果。

默认值：override=false

subsearch-options

语法：maxtime=<int> | maxout=<int> | timeout=<int>

描述：这些选项控制着如何执行子搜索。

子搜索选项

maxtime

语法：maxtime=<int>

描述：在自动完成之前，子搜索最多使用的时间，单位为秒。

默认值：60

maxout

语法：maxout=<int>

描述：从子搜索中输出的结果行的最大数量。

默认值：50000

timeout

语法：timeout=<int>

描述：等待子搜索完全完成的最长等待时间，单位为秒。

默认值：60

示例

示例 1：

搜索 "404" 事件并将每个事件中的字段附加到之前的搜索结果。

```
... | appendcols [search 404]
```

示例 2：

本搜索使用 appendcols 计算某个字段在特定服务器上出现的次数，并使用该值以计算其他字段。

```
specific.server | stats dc(userID) as totalUsers | appendcols [ search specific.server AND "text" | addinfo | where _time >= info_min_time AND _time <=info_max_time | stats count(field) as variableA ] | eval variableB = exact(variableA/totalUsers)
```

- 首先，本搜索会使用 stats 计算变量 "totalUsers" 的特定服务器和名称上的单个用户数量。
- 然后，此搜索使用 appendcols 搜索服务器，并计算某个字段在该特定服务器上出现的次数。该计数被重命名为 "VariableA"。addinfo 命令用于限制 info_min_time 和 info_max_time 范围内的本子搜索。
- eval 命令用于定义 "variableB"。

结果是带字段 totalUsers、variableA 和 variableB 的表格。

另请参阅

[append](#)、[appendpipe](#)、[join](#)、[set](#)

appendpipe

描述

向搜索结果添加子管道的结果。与子搜索不同，子管道开始是不运行的。搜索到达 `appendpipe` 命令时，子管道运行。`appendpipe` 命令用于附加转换命令的输出，如 `chart`、`timechart`、`stats` 和 `top`。

语法

```
appendpipe [run_in_preview=<bool>] [<subpipeline>]
```

可选参数

`run_in_preview`

语法: `run_in_preview=<bool>`

描述: 指定是否在预览中显示 `appendpipe` 命令的影响。设置为 `FALSE` 时，将运行搜索并在预览中显示结果，相当于 `appendpipe` 命令不再是搜索的一部分。但当搜索结束后，结果中将包含 `appendpipe` 命令的影响。

默认值: `True`

`subpipeline`

语法: `<subpipeline>`

描述: 搜索中 `appendpipe` 命令之前所发生的命令中的一组应用于搜索结果的命令。

用法

`appendpipe` 会很有用，因为它在您构建表格或图表时提供了汇总、总计或整个数据集的描述行。此命令在您需要其他计算原本的结果时，也很有用。

示例

示例 1:

跨所有用户附加每个操作的小计值。

```
index=_audit | stats count by action user | appendpipe [stats sum(count) as count by action | eval user = "TOTAL - ALL USERS"] | sort action
```

统计选项卡中显示的结果如下所示：

<code>action</code>	用户	<code>count</code>
accelerate_search	管理员	209
accelerate_search	buttercup	345
accelerate_search	can-delete	6
accelerate_search	TOTAL - ALL USERS	560
add	n/a	1
add	TOTAL - ALL USERS	1
change_authentication	管理员	50
change_authentication	buttercup	9
change_authentication	can-delete	24
change_authentication	TOTAL - ALL USERS	83

另请参阅

`append`, `appendcols`, `join`, `set`

arules

描述

`arules` 命令会查找字段值之间的关联关系。此命令将返回一个具有以下列的表格：“给定字段”、“隐含字段”、“强度”、“给定字段”支持和“隐含字段”支持。给定和隐含字段值是您提供的字段值。“强度”值指示给定字段值和隐含字段值之间的关系。

实现在 *Michael Hahsler, Bettina Gruen and Kurt Hornik (2012)*. *Arules* 中介绍的 `arules` 算法：挖掘“关联规则”和“频繁项目集”。R 软件包版本 1.0-12。本算法类似于用于在线购买网址的算法，这基于其他客户已经查看或购买的商品建议相关商品。

语法

`arules [<arules-option>...] <field-list>...`

必要参数

`field-list`

语法：`<field> <field> ...`

描述：字段名称的列表。必须至少指定两个字段。

可选参数

`<arules-option>`

语法：`<support> | <confidence>`

描述：`arules` 命令的选项。

`arules options`

`support`

语法：`sup=<int>`

描述：指定支持限制。所计算的支持级别小于该值的关联将不包含于输出结果中。支持选项必须是正整数。

默认值：3

`confidence`

语法：`conf=<float>`

描述：指定置信限制。含置信（表达为 Strength 字段）的关联将不包含于输出结果中。该值必须介于 0 到 1 之间。

默认值：.5

用法

`arules` 命令是一个流命令。请参阅“命令类型”。

示例

示例 1：搜索字段相关的可能性。

```
... | arules field1 field2 field3
```

示例 2：

```
... | arules sup=3 conf=.6 field1 field2 field3
```

另请参阅

`associate, correlate`

`associate`

描述

`associate` 命令识别字段间的相关性。此命令将尝试通过基于值来计算熵的变化以找到字段对之间的关系。此熵表示知道某个字段的值是否有助于预测另一字段的值。

从信息论的角度来说，熵是一种用于度量与随机变量相关联的不确定性的方式。因此，如果某字段只有一个唯一值，那么它的熵为零。如果此字段有多个值，那么这些值的分布越均匀，则熵就越高。

`associate` 命令使用香农熵（基于对数 2）。单位为 bits。

语法

`associate [<associate-options>...] [field-list]`

必要参数

无。

可选参数

`associate-option`

语法: `supcnt | supfreq | improv`

描述: `associate` 命令的选项。请参阅 **Associate-options** 部分。

`field-list`

语法: `<field>...`

描述: 一个或多个字段的列表。不能在字段列表中使用通配符字符。如果已指定字段列表，则分析仅限于这些字段。

默认值: 分析所有字段。

Associate-options

`supcnt`

语法: `supcnt=<num>`

描述: 指定 "reference key=reference value" 组合必须出现的次数下限。必须是非负整数。

默认值: 100

`supfreq`

语法: `supfreq=<num>`

描述: 指定 "reference key=reference value" 组合的最低频率，以占事件总数的比例表示。

默认值: 0.1

`improv`

语法: `improv=<num>`

描述: 指定“目标键”的限制或最小熵改进。计算得出的熵改进必须大于或等于此限制。

默认值: 0.5

输出表格中的列

`associate` 命令将输出一个表格，各表格列内包含下列字段。

字段	描述
<code>Reference_Key</code>	一个字段对中第一个字段的名称。
<code>Reference_Value</code>	一个字段对中第一个字段的值。
<code>Target_Key</code>	一个字段对中第二个字段的名称。
<code>Unconditional_Entropy</code>	目标键的熵。
<code>Conditional_Entropy</code>	当参考键为参考值时，目标键的熵。
<code>Entropy_Improvement</code>	无条件熵与条件熵之差。
<code>Description</code>	汇总基于熵计算的字段值之间关系的消息。Description 以文本方式表示结果。采用以下格式：“当 'Reference_Key' 包含值 'Reference_Value' 时，'Target_Key' 的熵从 Unconditional_Entropy 降为 Conditional_Entropy。”
<code>Support</code>	指定参考字段为参考值这种情况相对于事件总数的频率。例如，相对于事件总数，字段 A 等于值 X 的频率。

示例

1. 分析网络访问日志文件中字段之间的关系

此示例使用搜索教程中的示例数据。要在您自己的 Splunk 实例中尝试此示例，您必须下载示例数据并遵循操作以将教程数据导入 Splunk。当运行搜索时，请使用时间范围昨天。

此示例演示分析 Web 访问日志中字段之间关系的一种方式。

```
sourcetype=access_* status!=200 | fields method, status | associate improv=0.05 | table Reference_Key, Reference_Value, Target_Key, Top_Conditional_Value, Description
```

该搜索的第一部分检索返回的状态不是 200 的 Web 访问事件。Web 访问数据包含许多字段。您可以使用 `associate` 命令查看数据中所有字段与值对之间的关系。要简化此示例，可以将搜索限制为两个字段：`method` 和 `status`。

因为 `associate` 命令向输出添加了许多列，此搜索使用 `table` 命令仅显示选择的列。

统计选项卡中显示的结果如下所示：

Reference_Key	Reference_Value	Target_Key	Top_Conditional_Value	描述
method	POST	status	503 (17.44% -> 33.96%)	当 'method' 有值 'POST' 时，'status' 的熵从 2.923 减至 2.729。
status	400	method	GET (76.37% -> 83.45%)	当 'status' 有值 '400' 时，'method' 的熵从 0.789 减至 0.647。
status	404	method	GET (76.37% -> 81.27%)	当 'status' 有值 '404' 时，'method' 的熵从 0.789 减至 0.696。
status	406	method	GET (76.37% -> 81.69%)	当 'status' 有值 '406' 时，'method' 的熵从 0.789 减至 0.687。
status	408	method	GET (76.37% -> 80.00%)	当 'status' 有值 '408' 时，'method' 的熵从 0.789 减至 0.722。
status	500	method	GET (76.37% -> 80.73%)	当 'status' 有值 '500' 时，'method' 的熵从 0.789 减至 0.707。

在结果中，您可以看到结果中有一个方法和五个状态值。

从结果的第一行中可以看出，若 `method=POST`，这些事件的 `status` 字段为 503。`associate` 命令得出结论：如果 `method=POST`，`Top_Conditional_Value` 很可能有 33% 的时间是 503。

`Reference_Key` 和 `Reference_Value` 与 `Target_Key` 存在关联。

`Top_Conditional_Value` 字段说明了三项内容：

- 给定 `Reference_Value` 的最常见值
- 数据集中该字段的 `Reference_Value` 的频率
- 在该参考键中具有特定 `Reference_Value` 的事件的 `Target_Key` 中最常见关联值的频率。

该字段的格式为 "`CV (FRV% -> FCV%)`"，其中 `CV` 代表条件值，`FRV` 是参考值出现频率的百分比值，`FCV` 为特定参考值情况下对应条件值出现频率的百分比值。

2. 返回彼此之间至少有 3 个引用的结果

返回彼此关联的结果（彼此之间至少有 3 个引用）。

```
index=_internal sourcetype=splunkd | associate supcnt=3
```

3. 分析来自主机的事件

分析来自主机 "reports" 的所有事件并返回彼此关联的结果。

```
host="reports" | associate supcnt=50 supfreq=0.2 improv=0.5
```

另请参阅

`arules`, `correlate`, `contingency`

audit

描述

返回本地审计索引中存储的审计线索信息。此命令在检查间隙和篡改的同时会验证带签名的审计事件。

语法

audit

示例

示例 1：查看“审计”索引中的信息。

```
index="_audit" | audit
```

autoregress

描述

通过将 *field* 的一个或多个先前值复制到每个事件中，让这些事件为自动回归或移动平均值的计算做好准备。

刚开始的一些事件由于之前的值不存在，会缺少之前值的增强功能。

语法

```
autoregress <field> [AS <newfield>] [ p=<int> | p=<int>-<int> ]
```

必要参数

field

语法：`<string>`

描述：字段的名称。最有效的具有数字值的字段。

可选参数

p

语法：`p=<int> | p=<int>-<int>`

描述：指定从哪些之前的事件中复制值。可指定两种形式：单个整数和数字范围。若为单个整数，如 3，autoregress 命令会将第三个先前事件的字段值复制到新的字段。若为数字范围，autoregress 命令会从先前事件的范围内复制字段值。例如，若您指定了一个数字范围，如 `p=2-4`，则第二个、第三个和第四个先前事件的字段值将被复制到新的字段中。

默认值：1

newfield

语法：`<field>`

描述：若将 `p` 设置为单个整数，`newfield` 参数将指定一个可把单个字段值复制到其中的字段名称。若 `p` 设置为一个数字范围则无效。

若未指定 `newfield` 参数，单个或多个值将被复制到名为 `<field>_p<num>` 的字段中。例如，若 `p=2-4` 且 `field=count`，字段名称将为 `count_p2`、`count_p3` 和 `count_p4`。

用法

autoregress 命令属于中央流命令。请参阅“命令类型”。

示例

示例 1：

对于每个事件，将 'ip' 字段前面的第三个值复制到 'old_ip' 字段中。

```
... | autoregress ip AS old_ip p=3
```

示例 2：

对于每个事件，复制前面第 2、3、4 和 5 个 'count' 字段值。

```
... | autoregress count p=2-5
```

由于并未指定 `new field` 参数，值将被复制到 '`count_p2`'、'`count_p3`'、'`count_p4`' 和 '`count_p5`' 字段中。

示例 3:

计算现有事件和之前四个事件的事件大小的移动平均值。因为空字段总和被视为空，所以此搜索会省略字段会出错的初始事件的 `moving_average`。

```
... | eval rawlen=len(_raw) | autoregress rawlen p=1-4 | eval moving_average=(rawlen + rawlen_p1 + rawlen_p2 + rawlen_p3 + rawlen_p4) /5
```

另请参阅

`accum`, `delta`, `streamstats`, `trendline`

awssnsalert

`awssnsalert` 命令与 Splunk Add-on for AWS 结合使用。

此命令的相关信息，请参阅 *Splunk Add-on for AWS* 中的“使用 `awssnsalert` 搜索命令”。

bin

描述

通过调整 `<field>` 值将连续数字值放入离散集合或数据桶，这样特定集合中的所有项目都具有同样的值。

`chart` 和 `timechart` 命令会自动呼叫 `bin` 命令。仅将 `bin` 命令用于处理 `chart` 和 `timechart` 命令无法处理的统计运算。如果您打算将所有事件导出为 CSV 或 JSON 文件格式，请勿使用 `bin` 命令。

语法

```
bin [<bin-options>...]<field> [AS <newfield>]
```

必要参数

field

语法: `<field>`

描述: 指定一个字段名称。

可选参数

bin 选项

语法: `bins` | `minspan` | `span` | `<start-end>` | `aligntime`

描述: 离散化选项。请参阅本主题中的“Bin 选项”部分了解每个选项的语法和描述。

newfield

语法: `<string>`

描述: 字段的新名称。

Bin 选项

bins

语法: `bins=<int>`

描述: 设置离散为数据箱的最大数量。

minspan

语法: `minspan=<span-length>`

描述: 指定最小的跨度粒度，以自动使用来自数据时间范围的推断跨度。

span

语法: `span = <log-span> | <span-length>`

描述: 使用基于时间的跨度长度或基于对数的跨度设置每个数据箱的大小。

<start-end>

语法: start=<num> | end=<num>

描述: 设置数字型数据箱的最小和最大范围。将分析字段的数据并确定开始和结束值。如果未指定跨度值，则会使用开始和结束参数。

仅可以使用开始或结束参数来扩展范围，而不能缩减范围。例如，如果字段代表秒数，值的范围为 0–59。如果您指定跨度值为 10，则会以 10 为增量来计算数据箱。即，数据箱为 0–9、10–19、20–29，以此类推。如果未指定跨度值，但指定 end=1000，则会基于实际开始值并以 1000 为结束值来计算数据箱。

若您设置 end=10 且各值 >10，结束参数将不会产生任何效果。

aligntime

语法: aligntime=(earliest | latest | <time-specifier>)

描述: 将数据箱时间和基本 UTC 时间 (epoch 0) 以外的时间对齐。仅当执行基于时间离散化时 aligntime 选项有效。如果 span 是以日期、月份或年份为单位，请忽略。

跨度选项

log-span

语法: [<num>] log[<num>]

描述: 设置为基于对数的跨度。第一个数字为系数，第二个数字为底。若提供第一个数字，该数字必须是大于等于 1.0 且小于底的实数。若提供底，则底必须是大于 1.0 的实数（严格大于 1）。

示例: span=2 log10

span-length

语法: <int>[<timescale>]

描述: 每个数据箱的跨度。若基于 _time 字段离散化或与 timescale 一起使用，此将被视为时间范围。否则，这是一个绝对的数据箱长度。

<timescale>

语法: <sec> | <min> | <hr> | <day> | <month> | <subseconds>

描述: 时间刻度单位。若基于 _time 字段离散化。

默认值: sec

时间刻度	语法	描述
<sec>	s sec secs second seconds	时间刻度（秒）。
<min>	m min mins minute minutes	时间刻度（分钟）。
<hr>	h hr hrs hour hours	时间刻度（小时）。
<day>	d day days	时间刻度（天）。
<month>	mon month months	时间刻度（月）。
<subseconds>	us ms cs ds	单位为微秒 (us)、毫秒 (ms)、百分之一秒 (cs) 或十分之一秒 (ds) 的时间刻度。

用法

bucket 命令为 bin 命令的别名。

bin 命令通常属于数据集处理命令。如果使用命令指定 span 参数，那么 bin 命令是流命令。请参阅“命令类型”。

亚秒级数据箱时间跨度

亚秒级 span 时间刻度（由秒 (ds)、百分之一 (cs)、毫秒 (ms) 或微秒 (us) 组成的时间跨度）应为一秒内可均分的数字。例如，1s = 1000ms。这意味着有效的毫秒 span 值为 1、2、4、5、8、10、20、25、40、50、100、125、200、250 或 500ms。另外，不允许设置 span = 1000ms。改用 span = 1s。

示例

示例 1:

返回每 5 分钟跨度内每个 "host" 的平均 "thruput"。

```
... | bin _time span=5m | stats avg(thrput) by _time host
```

示例 2:

数据箱在 10 个数据箱中搜索结果，并返回每个数据箱原始事件的计数。

```
... | bin size bins=10 | stats count(_raw) by size
```

示例 3:

以大于所需值的结束值新建数据箱，确保包含了所有可能值。

```
... | bin amount end=1000
```

示例 4:

将时间数据箱对齐到上午 3 点（本地时间）。将跨度设为 12 小时。数据箱将表示上午 3 点到下午 3 点，然后是下午 3 点到第二天上午 3 点，依次类推。

```
... | bin _time span=12h aligntime=@d+3h
```

示例 5:

将数据箱对齐到特定的 UTC 时间 1500567890。

```
... | bin _time aligntime=1500567890
```

另请参阅

`chart`, `timechart`

bucket

`bucket` 命令为 `bin` 命令的别名。请参阅 `bin` 命令了解语法信息和示例。

bucketdir

描述

用更高级别的分组替换字段值，例如，用目录替换文件名。

返回 `maxcount` 事件，具体做法为：获取传入事件并将多个来源汇总到目录中，此过程中优先选取内含文件较多但事件较少的目录。具有路径的字段为 `PATHFIELD`（如 `source`），各字符串由分隔字符分隔。默认为 `pathfield=source; sizefield=totalCount; maxcount=20; countfield=totalCount; sep="/" 或 "\\"`（具体字符取决于操作系统）。

语法

```
bucketdir pathfield=<field> sizefield=<field> [maxcount=<int>] [countfield=<field>] [sep=<char>]
```

必要参数

pathfield

语法：`pathfield=<field>`

描述：指定一个具有路径值的字段名。

sizefield

语法：`sizefield=<field>`

描述：指定一个定义数据桶大小的数值字段。

可选参数

countfield

语法：`countfield=<field>`

描述：指定一个描述事件计数的数值字段。

maxcount

语法: maxcount=<int>

描述: 指定数据桶的事件总数。

sep

语法: <char>

描述: 分隔符。指定为正斜杠 "/" 或双反斜杠 "\\", 具体取决于操作系统。

用法

bucketdir 命令是一个流命令。请参阅“命令类型”。

示例

示例 1:

返回 10 个最佳来源和目录。

```
... | top source | bucketdir pathfield=source sizefield=count maxcount=10
```

另请参阅

cluster, dedup

cefout

cefout 命令与 Splunk App for CEF 结合使用。

如需此命令的相关信息，请参阅部署和使用 *Splunk App for CEF* 中的“技术实施”。

chart

描述

chart 命令是一种转换命令，将以表格格式返回结果，之后可以使用返回的结果，以柱状图、折线图、面积图或饼图等图表显示数据。请参阅《仪表板和可视化》手册中的“可视化参考信息”。

使用 chart 命令时必须指定一个统计函数。请参阅“统计和图表函数”。

语法

要求的语法以粗体表示。

```
chart
[<chart-options>]
[agg=<stats-agg-term>]
( <stats-agg-term> | <sparkline-agg-term> | "("<eval-expression>"")" )...
[ BY <row-split> <column-split> ] | [ OVER <row-split> ] [BY <column-split>]
[<dedup_splitvals>]
```

必要参数

使用 chart 命令时必须包括其中一个参数。

stats-agg-term

语法: <stats-func> (<eval-ed-field> | <wc-field>) [AS <wc-field>]

描述: 统计聚合函数。请参阅“Stats 函数”选项。该函数可应用于 Eval 表达式、字段或字段组。使用 AS 子句将结果放入您已指定其名称的新字段内。可以在字段名称中使用通配符字符。

sparkline-agg-term

语法: <sparkline-agg> [AS <wc-field>]

描述: 迷你图聚合函数。使用 AS 子句将结果放入您已指定其名称的新字段内。可以在字段名称中使用通配符字符。请参阅“Sparkline 选项”。

Eval-expression

语法: <eval-math-exp> | <eval-concat-exp> | <eval-compare-exp> | <eval-bool-exp> | <eval-function-call>

描述：代表目标字段值的文字、字段、运算符以及函数的组合。有关更多信息，请参阅“评估函数”。请参阅“用法”。

为使这些计算正常运行，值对于运算类型而言必须有效。例如，除了加法之外，如果值不是数字，则算术运算将不会生成有效的结果。如果两个操作数都是字符串，则可以连接它们。如果使用句点对值进行连接，则无论这两个值实际是何种类型，此搜索都会将其视为字符串。

可选参数

agg
语法：agg=<stats-agg-term>
描述：指定一个聚合器或函数。有关 stats 函数的列表以及相应的描述和示例，请参阅“统计和图表函数”。

chart-options
语法：cont | format | limit | sep
描述：可指定用于优化结果的选项。请参阅本主题中的“图表选项”部分。
默认值：

column-split
语法：<field> [<tc-options>]... [<where-clause>]
描述：指定一个字段，可在结果表格中用作列。根据默认设置，若结果是可视化的，各列将变成图表中的数据系列。若字段为数字，将使用 tc-options 参数应用离散化。请参阅本主题中的 tc 选项和 where 子句部分。
默认值：默认情况下，所包含的列数限定为 10，可以更改包含 <where-clause> 的列数。

加入 column-split 字段后，输出内容将为表格，其中每列代表 split-by 字段的非重复值。与之不同的是，by-clause 中的每行代表 group-by 字段值的单个唯一组合。更多信息请参阅本主题中的“用法”部分。

dedup_splitvals
语法：dedup_splitvals=<boolean>
描述：指定是否删除多值 BY 子句中的重复值。
默认值：false

row-split
语法：<field> [<bin-options>]...
描述：您指定的字段变成结果表格中的第一列。字段值变成结果表格中的行标签。在图表中，字段名称用于标记 X 轴。字段值变成 X 轴值。请参阅本主题中的“Bin 选项部分”。
默认值：无。

图表选项

cont
语法：cont=<bool>
描述：指定 Bin 是否连续。若 cont=false 则重绘 X 轴，以使 x 值数据桶的非连续序列在输出时相邻显示。若 cont=true 则未含值的数据桶将显示统计结果为 0 或为空值。
默认值：true

format
语法：format=<string>
描述：当多个数据系列由 split-by 字段进行分隔时，用于构建输出字段名称。format 优先于 sep 并允许您使用 stats 聚合器和函数 (\$AGG\$) 以及 split-by-field 的值 (\$VAL\$) 指定参数化表达式。

limit
语法：limit=(top | bottom) <int>
描述：只有在指定 column-split 时有效。使用 limit 选项指定应出现于输出中的结果数量。若设置了 limit=N，则会根据每个系列的总和以及您所选择的前缀保留前或后 N 个值。若 limit=0，则将返回所有结果。如果您选择不提供前缀，则 Splunk 软件会提供前几个结果。
默认值：top 10

sep
语法：sep=<string>
描述：当多个数据系列由 split-by 字段进行分隔时，用于构建输出字段名称。这相当于把 format 设置为 \$AGG\$<sep>\$VAL\$。

Stats 函数选项

stats-func
语法：语法取决于您使用的函数。请参阅“用法”。
描述：可与 chart 命令一起使用的统计和图表函数。每次调用 chart 命令时都可以使用一个或多个函数。但是，只能使用一个 BY 子句。

迷你图选项

迷你图是内联图表，位于搜索结果的表单元格内，显示与每一行的主键相关联的基于时间的趋势。

sparkline-agg

语法: sparkline (count(<wc-field>), <span-length>) | sparkline (<sparkline-func>(<wc-field>), <span-length>)

描述: 迷你图说明符，它将获取某一个字段上的聚合函数的第一个参数和一个可选的时间跨度说明符。如果未使用时间跨度说明符，将根据搜索的时间范围选择合适的时间跨度。若迷你图的范围未限制到某一字段，仅允许使用计数聚合函数。可以在字段名称中使用通配符字符。

span-length

请参阅本主题中的“跨度选项”部分。

sparkline-func

语法: c() | count() | dc() | mean() | avg() | stdev() | stdevp() | var() | varp() | sum() | sumsq() | min() | max() | range()

描述: 用来生成迷你图值的聚合函数。每个迷你图值都是通过将此聚合应用到落入每个特定时间数据桶范围内的事件而生成。

迷你图的大小由 `limits.conf` 文件中的设置定义。`sparkline_maxsize` 设置用于定义为迷你图提供的最大元素数。

更多信息请参阅《搜索手册》中的“为您的搜索结果添加迷你图”。

Bin 选项

Bin 选项控制将搜索结果分隔或离散化为多少及多大的数据箱，即数据箱的数量和大小。

语法: bins | span | <start-end> | aligntime

描述: 离散化选项。

默认值: `bins=300`

bins

语法: `bins=<int>`

描述: 设置离散化为数据箱的最大数量。例如，如果 `bin=300`，则搜索会查找可生成不超过 300 个非重复数据箱的最小数据箱大小。

默认值: 300

span

语法: `span=<log-span>` | `span=<span-length>`

描述: 使用基于时间的跨度长度或基于对数的跨度设置每个数据箱的大小。请参阅本主题中的“跨度选项”部分。

<start-end>

语法: `end=<num>` | `start=<num>`

描述: 设置数字型数据箱的最小和最大范围。在 `[start, end]` 范围之外的数据将遭丢弃。

aligntime

语法: `aligntime=(earliest | latest | <time-specifier>)`

描述: 将数据箱时间和基本 UNIX 时间 (epoch 0) 以外的时间对齐。仅当执行基于时间离散化时 `aligntime` 选项有效。如果 `span` 是以日期、月份或年份为单位，请忽略。

跨度选项

<log-span>

语法: `[<num>] log [<num>]`

描述: 设置为基于对数的跨度。第一个数字为系数，第二个数字为底。若提供第一个数字，该数字必须是大于等于 1.0 且小于底的实数。若提供底，则底必须是大于 1.0 的实数（严格大于 1）。

span-length

语法: `[<timescale>]`

描述: 基于时间的跨度长度。

语法: `<int>`

描述: 每个数据箱的跨度。如果使用 `timescale`，则将其用作时间范围。否则，这是一个绝对的数据桶“长度”。

<timescale>

语法: `<sec> | <min> | <hr> | <day> | <month> | <subseconds>`

描述: 时间刻度单位。

时间刻度	语法	描述
<sec>	s sec secs second seconds	时间刻度（秒）。
<min>	m min mins minute minutes	时间刻度（分钟）。
<hr>	h hr hrs hour hours	时间刻度（小时）。
<day>	d day days	时间刻度（天）。
<month>	mon month months	时间刻度（月）。
<subseconds>	us ms cs ds	单位为微秒 (us)、毫秒 (ms)、百分之一秒 (cs) 或十分之一秒 (ds) 的时间刻度。

tc 选项

Timechart 选项是 <column-split> 参数的一部分，用于控制按字段拆分搜索结果的行为。有一些选项可控制将搜索结果分隔为多少及多大的数据箱，即数据箱的数量和大小。有一些选项可控制当事件不包含 split-by 字段时以及出现不符合 <where-clause> 条件的事件时会发生的情况。

tc 选项

语法：<bin-options> | usenull=<bool> | useother=<bool> | nullstr=<string> | otherstr=<string>
描述：用于控制拆分方式字段行为的选项。

bin 选项

请参阅本主题中的“Bin 选项部分”。

nullstr

语法：nullstr=<string>
描述：为不包含 split-by 字段的事件指定数据系列的字段名称。仅当 usenull 选项设置为 true 时，nullstr 选项才适用。
默认值：NULL

otherstr

字符串：otherstr=<string>
描述：指定不符合 <where-clause> 条件的数据系列的字段名称。仅当 useother 选项设置为 true 时，otherstr 选项才适用。
默认值：OTHER

usenull

语法：usenull=<bool>
描述：用于控制是否为不包含 split-by 字段的事件创建一个系列。
默认值：true

useother

语法：useother=<bool>
描述：指定是否应该为图表中未包含的数据系列（因为这些系列未满足 <where-clause> 的条件）添加一个系列。
默认值：true

Where 子句

<where-clause> 是 <column-split> 参数的一部分。

Where 子句

语法：<single-agg> <where-comp>
描述：指定在 tc-by-clause 中给定某字段时包含特定数据系列的条件。此选项最常见的用法是，在系列选择中选择峰值而不是分布总和。默认值将根据曲线下的区域查找前十个系列。或者，可将总和替换为最大值，以查找具有十个最高峰值的系列。这与 where 命令无关。

single-agg

语法：count | <stats-func>(<field>)
描述：应用到一个单一字段的单一聚合，包括已评估字段。不允许使用通配符。必须指定此字段，除非使用的是以整体形式应用于事件的 count 聚合函数。

<stats-func>

请参阅本主题中的“统计函数”部分。

<where-comp>

语法：<wherein-comp> | <wherethresh-comp>

描述：<where-clause> 的条件。

<wherein-comp>

语法：(in |notin) (top | bottom)<int>

描述：<where-clause> 的分组条件。聚合系列值在或不在某些最大值或最小值分组内。

<wherethresh-comp>

语法：(< | >) <num>

描述：<where-clause> 的阈值。聚合系列值必须大于或小于指定的数字阈值。

用法

chart 命令是一个转换命令。请参阅“命令类型”。

评估表达式

可以将 chart 命令和 eval 表达式搭配使用。除非您指定了 split-by 子句，否则必须重命名 Eval 表达式。

支持的函数

您可以将 stats 命令与一系列函数结合使用。有关使用函数的一般信息，请参阅“统计和图表函数”。

- 若需按类别排列的统计函数列表，请参阅“按类别排列的函数列表”
- 若需按字母顺序排列的统计列表，请参阅“按字母顺序排列的函数列表”

函数和内存用法

就内存而言，某些函数本身就比其他函数占用更多内存。例如，distinct_count 函数需要的内存比 count 函数大得多。values 和 list 函数也会消耗大量的内存。

如果您使用的是未结合 split-by 字段或结合按字段的低基数 split-by 的 distinct_count 函数，考虑用 estdc 函数替换 distinct_count 函数（估计非重复计数）。estdc 函数可降低内存使用和减少运行时间。

将统计函数应用于所有可用字段

对于未与一个或多个字段配对的函数，有些统计命令（如 stats）会将其视为已与隐式通配符配对并进行相应处理，因此这些命令会将函数应用于所有可用字段。例如，| stats sum 被视为 | stats sum(*)。

chart 命令仅在与 count 函数搭配使用时才允许这种行为。如果没有为 count 指定字段，则 chart 会将其应用于搜索返回的所有事件。如果要将其他函数应用于所有字段，则必须使用显式通配符：| chart sum(*)。

X 轴

您可以指定图表的 X 轴上要跟踪的字段。使用 by 字段指定 X 轴变量，必要时对该变量进行离散化处理。如果必要，还要将图表字段转换为数字型量。

timechart 命令会生成一个以 _time 字段作为 X 轴的图表，与其不同的是，chart 命令会产生一个以任意字段作为 X 轴的表格。

您还可以在 over 关键字之后，及任意 by 及其后面的 split-by 子句之前指定 X 轴字段。limit 和 agg 选项可以轻松指定系列筛选。若提供了显式的 Where 子句，则忽略 limit 和 agg 选项。

使用 row-split 字段与 column-split 字段

加入 column-split 字段后，输出内容将为表格，其中每列代表 column-split 字段的非重复值。与之不同的是，stats 命令中的每行代表 group-by 字段值的单个唯一组合。默认情况下，所包含的列数限定为 10，可以更改包含 where-clause 的列数。

使用 chart 和 timechart 命令既无法指定一函数内的相同字段，也无法将其指定为 row-split 字段。

例如，不能运行此搜索。sum 函数和 row-split 参数中已指定了字段 A。

... | chart sum(A) by A span=log2

与 row-split 参数中一样，您必须指定一个不同的字段。

不过，您也可以通过使用 eval 表达式来规避这个问题。例如：

... | eval A1=A | chart sum(A) by A1 span=log2

亚秒级数据箱时间跨度

亚秒级 span 时间刻度（由秒 (ds)、百分之一 (cs)、毫秒 (ms) 或微秒 (us) 组成的时间跨度）应为一秒内可均分的数字。例如， $1\text{s} = 1000\text{ms}$ 。这意味着有效的毫秒 span 值为 1、2、4、5、8、10、20、25、40、50、100、125、200、250 或 500ms。另外，不允许设置 span = 1000ms。改用 span = 1s。

基本示例

1. 按每个 foo 值的 max(delay) 绘制图表

返回每个 foo 值的 max(delay)。

```
... | chart max(delay) OVER foo
```

2. 针对由 bar 的值拆分的每个 foo 值的 max(delay) 绘制图表

返回由 bar 的值拆分的每个 foo 值的 max(delay)。

```
... | chart max(delay) OVER foo BY bar
```

3. 针对每个非重复 "host" 和 "user" 对的平均值与最大 "delay" 的比绘制图表

返回每个非重复 "host" 和 "user" 对的平均 "size" 与最大 "delay" 的比。

```
... | chart eval(avg(size)/max(delay)) AS ratio BY host user
```

4. 针对最大 "delay" 与 "size" 的积绘制图表，并将 "size" 拆分为数据箱

返回最大 "delay" 与 "size" 的积，其中 "size" 拆分为最多 10 个大小相同的数据箱。

```
... | chart max(delay) BY size bins=10
```

5. 按每个非重复 "host" 的平均 "size" 绘制图表

返回每个非重复 "host" 的平均 "size"。

```
... | chart avg(size) BY host
```

6. 针对以日期和小时数分组的事件数量绘制图表

返回的事件数量按日期和一天内的小时数分组，采用间隔为每 7 天和每一半天数的每 24 小时一组。跨度立即应用于命令之前的字段。

```
... | chart count BY date_mday span=3 date_hour span=12
```

7. 将图表时间数据箱对齐到本地时间

将时间数据箱对齐到上午 5 点（本地时间）。将跨度设为 12 小时。数据箱将表示上午 5 点到下午 5 点，然后是下午 5 点到第二天上午 5 点，依次类推。

```
... | chart _time span=12h aligntime=@d+5h
```

8. 在多值 BY 字段中，删除重复的值

对于每个唯一的 mvfield 值，用图表记录 field 的平均值。删除重复的 mvfield 中的值。

```
... | chart avg(field) BY mvfield dedup_splitval=true
```

延伸示例

1. 用图表命令指定 <row-split> 和 <column-split> 值

此示例使用列出每个产品和季度数字销量的事件，例如：

产品	季度	销售
----	----	----

ProductA	QTR1	1, 200
ProductB	QTR1	1, 400
ProductC	QTR1	1, 650
ProductA	QTR2	1, 425
ProductB	QTR2	1, 175
ProductC	QTR2	1, 550
ProductA	QTR3	1, 300
ProductB	QTR3	1, 250
ProductC	QTR3	1, 375
ProductA	QTR4	1, 550
ProductB	QTR4	1, 700
ProductC	QTR4	1, 625

要按产品为每个季度汇总数据，请运行此搜索：

```
source="addtotalsData.csv" | chart sum(sales) BY products quarter
```

在此示例中，BY 子句中有两个通过 chart 命令指定的字段。

- products 字段指的是 <row-split> 字段。图表中，此字段形成 X 轴。
- quarter 字段指的是 <column-split> 字段。图表中，此字段形成数据系列。

统计选项卡中显示的结果如下所示：

产品	QTR1	QTR2	QTR3	QTR4
ProductA	1, 200	1, 425	1, 300	1, 550
ProductB	1, 400	1, 175	1, 250	1, 700
ProductC	1, 650	1, 550	1, 375	1, 625

单击可视化选项卡以图表形式查看结果。

请查看 addtotals 命令以了解为每个产品添加列数总计的示例。

2. 针对每个 Web 服务器上发生的不同页面请求的数量绘制图表

此示例使用搜索教程中的示例数据，但应该可以与任意格式的 Apache Web 访问日志配合使用。要在您自己的 Splunk 实例中尝试此示例，您必须下载示例数据并遵循操作以将教程数据导入 Splunk。当运行搜索时，请使用时间范围所有时间。

针对每个 Web 服务器上发生的不同页面请求（GET 和 POST）的数量绘制图表。

```
sourcetype=access_* | chart count(eval(method="GET")) AS GET, count(eval(method="POST")) AS POST by host
```

本示例使用 eval 表达式来指定 stats 命令要统计的不同字段值。第一个子句使用 count() 函数来统计包含 method 字段值 GET 的 Web 访问事件。然后，使用 AS 关键字，代表这些结果的字段重命名为 GET。

第二个子句对 POST 事件执行相同的操作。然后两种类型的事件计数都由 web 服务器分隔，使用有 host 字段的 BY 子句。

统计选项卡中显示的结果如下所示：

host	GET	POST
www1	8, 431	5, 197
www2	8, 097	4, 815

www3	8,338	4,654
------	-------	-------

单击可视化选项卡。如需要，将结果格式设置为柱状图。此图表根据 host 值显示每个事件类型（GET 或 POST）的事件总数。



3. 以按持续时间显示的交易数量来绘制图表

此示例使用搜索教程中的示例数据。要在您自己的 Splunk 实例中尝试此示例，您必须下载示例数据并遵循操作以将教程数据导入 Splunk。当运行搜索时，请使用时间范围所有时间。

创建一个基于持续时间（秒）显示交易数的图表。

```
sourcetype=access_* status=200 action=purchase | transaction clientip maxspan=10m | chart count BY duration span=log2
```

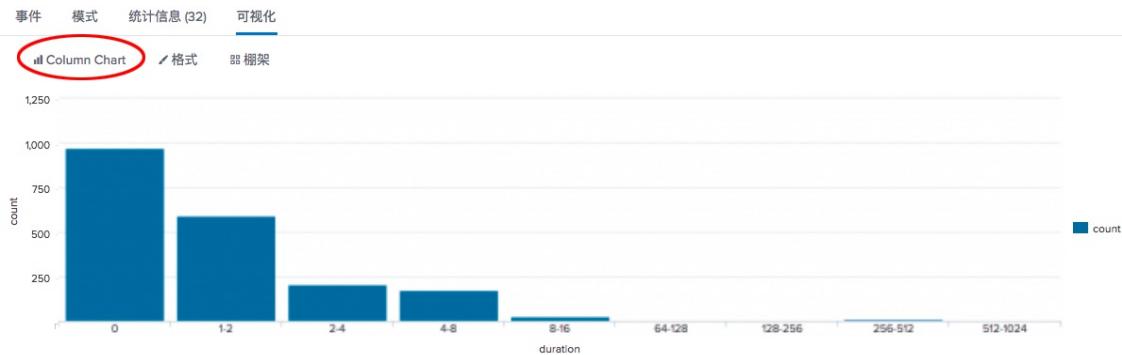
本搜索使用 `transaction` 命令将交易定义为共享 `clientip` 字段且时间范围在十分钟内的事件。`transaction` 命令创建了名为 `duration` 的新字段，其为交易内第一个事件和最后一个事件之间的时间戳差异。（由于 `maxspan=10s`，`duration` 值应小于该值。）

之后，各交易将通过管道符传递给 `chart` 命令。`count()` 函数用于统计交易的数量，并按每笔交易的持续时间分隔统计结果。由于持续时间以秒计算且您预期会有很多值，本搜索将使用 `span` 参数把持续时间存储到大小为 `log2` (`span=log2`) 的数据箱中。

统计选项卡中显示的结果如下所示：

duration	计数
0	970
1-2	593
2-4	208
4-8	173
8-16	26
64-128	3
128-256	3
256-512	12
512-1024	2

单击可视化选项卡。如需要，将结果格式设置为柱状图。



在此数据集中，大部分交易都在 0 到 2 秒内完成。

4. 创建一个基于交易的持续时间显示交易中的平均事件数的图表

此示例使用搜索教程中的示例数据。要在您自己的 Splunk 实例中尝试此示例，您必须下载示例数据并遵循操作以将教程数据导入 Splunk。当运行搜索时，请使用时间范围所有时间。

新建一个基于交易的持续时间显示交易中的平均事件数的图表。

```
sourcetype=access_* status=200 action=purchase | transaction clientip maxspan=30m | chart avg(eventcount) by duration span=log2
```

transaction 命令向结果 duration 和 eventcount 添加两个字段。eventcount 字段追踪单个交易中的事件数量。

在此搜索中，交易传递到 chart 命令。avg() 函数用于计算每个期间的平均事件数。因为持续时间以秒为单位，预期会有许多值，搜索使用 span 通过基数为 2 的算法将数据箱持续时间分为数据箱。

使用字段格式选项以启用数字格式化。

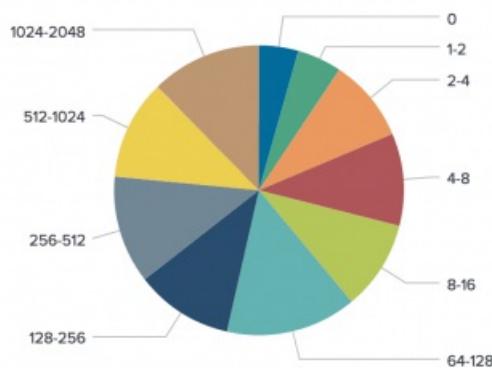
The screenshot shows the Splunk search interface with a search bar containing the command: sourcetype=access_* status=200 action=purchase | transaction clientip maxspan=30m | chart avg(eventcount) by duration span=log2. Below the search bar, there are tabs for '事件' (Events), '模式' (Mode), '统计信息 (438)' (Statistics (438)), and '可视化' (Visualization). The '可视化' tab is selected. A modal dialog box titled '字段格式选项' (Field Format Options) is open, with the '数字格式' (Number Format) dropdown set to '已禁用' (Disabled). The main search results page shows a chart with two columns: 'duration' and 'avg(eventcount)'. The 'avg(eventcount)' column contains numerical values like 2.818950177935943, 1.83571296124606, etc.

单击可视化选项卡并将显示形式更改为饼图。

Pie Chart

格式

器棚架



饼图的每个扇形代表事件交易的持续时间。您可以悬停在扇形区以查看平均值。

5. 针对客户购买绘制图表

此示例使用搜索教程中的示例数据。要在您自己的 Splunk 实例中尝试此示例，您必须下载示例数据并遵循操作以将教程数据导入 Splunk。当运行搜索时，请使用时间范围昨天。

针对昨天在 Buttercup Games 在线商店买东西的人数以及购买的实际商品绘制图表。

```
sourcetype=access_* status=200 action=purchase | chart dc(clientip) OVER date_hour BY categoryId usenull=f
```

此搜索获取购物事件，并通过管道符将其传递给 chart 命令。dc() 或 distinct_count() 函数用于统计非重复访问者（由 clientip 字段表示）的数量。之后，以一天内的小时时段为单位绘制该数量的图表，并根据购物的 category_id 进行细分。此外，由于它们是数字值，此搜索将使用 usenull=f 参数排除未含值的字段。

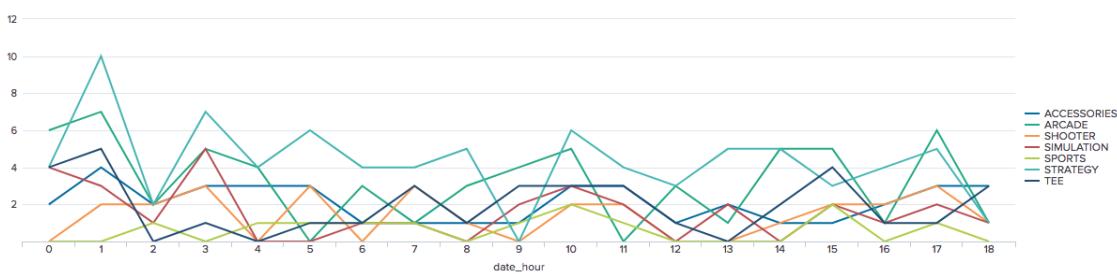
统计选项卡中显示的结果如下所示：

date_hour	ACCESSORIES	ARCADE	SHOOTER	SIMULATION	SPORTS	STRATEGY	TEE
0	2	6	0	4	0	4	4
1	4	7	2	3	0	10	5
2	2	2	2	1	1	2	0
3	3	5	3	5	0	7	1
4	3	4	0	0	1	4	0
5	3	0	3	0	1	6	1

单击可视化选项卡。如需要，将结果格式设置为折线图。

事件 模式 统计信息 (19) 可视化

Line Chart 格式 器棚架



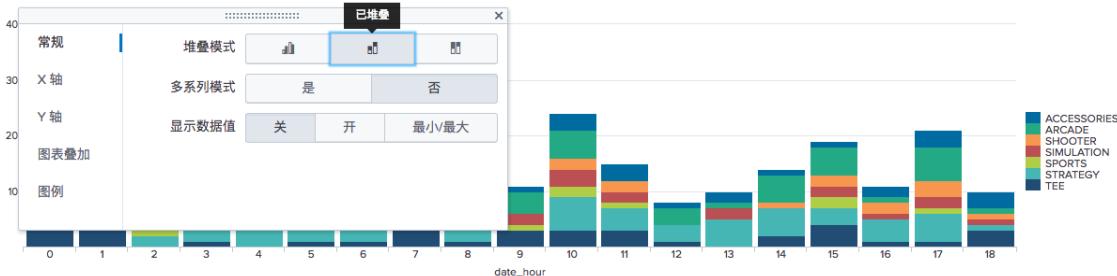
每条折线都代表在 Buttercup Games 在线商店售出的一种不同类型的产品。每条折线的高度表示在对应小时时段内购买该产品的不同顾客的数量。总体来说，街机游戏看起来是网上商店最畅销的商品。

您可将报表格式化为堆叠柱状图，这将显示一天中的每个小时的总购买量。

1. 将图表类型修改为柱状图。
2. 使用格式菜单并在一般选项卡上选择堆叠。

事件 模式 统计信息 (19) 可视化

Column Chart 格式 器棚架



6. 针对地震数量和每次地震的震级绘制图表

此示例使用从 USGS 地震网站下载的近期地震数据。该数据是一个逗号分隔的 ASCII 文本文件，其中包含每次记录的地震的震级 (mag)、坐标（经度、纬度）、区域（地点）等。

您可以从 [USGS 地震源](#) 下载当前 CSV 文件并作为输入添加。

创建列出发生在阿拉斯加及周边地区的地震数和每次地震的震级的图表。使用时间范围所有时间运行搜索。

```
source=all_month.csv place=*alaska* mag>=3.5 | chart count BY mag place useother=f | rename mag AS Magnitude
```

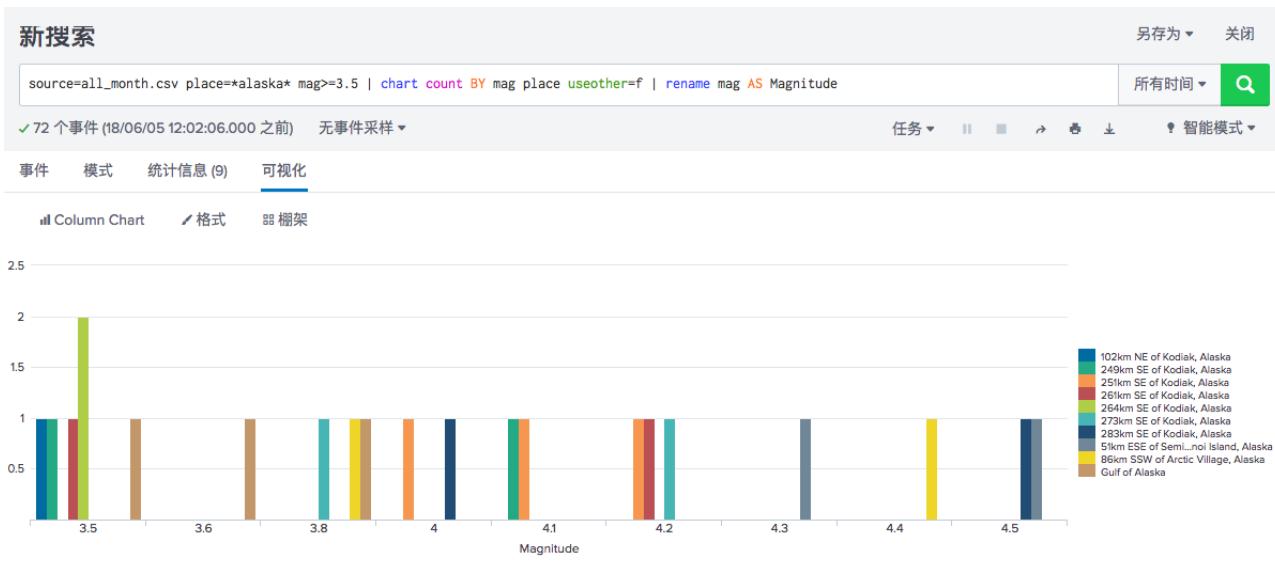
此搜索将统计在阿拉斯加各个区域发生的地震的数量。之后，统计结果将根据地震震级，按照每个 place 进行细分。由于 place 值并非数字，此搜索使用 useother=f 参数排除不匹配的事件。

统计选项卡中显示的结果如下所示：

震级	阿拉斯加奇里科夫岛东偏北 145km	阿拉斯加科迪亚克东南 225km	阿拉斯加科迪亚克东南 250km	阿拉斯加科迪亚克东南 252km	阿拉斯加科迪亚克东南 254km	阿拉斯加科迪亚克东南 255km	阿拉斯加科迪亚克东南 259km	阿拉斯加科迪亚克东南 264km	阿拉斯加科迪亚克东南 265km	阿拉斯加湾
3. 5	1	1	0	1	0	1	0	0	2	2
3. 6	0	0	1	0	0	0	0	1	0	1
3. 7	0	0	0	0	1	0	0	0	0	2
3. 8	0	1	0	0	0	0	1	1	0	3
3. 9	0	0	1	0	1	0	0	0	0	0
4	0	0	0	0	1	1	0	0	0	1

4.1	0	0	0	0	0	0	0	0	0	1
4.2	0	0	0	1	0	0	0	0	0	1
4.3	0	0	0	0	0	0	0	0	0	1
4.4	0	0	0	0	0	0	1	0	0	1
4.6	1	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	1

单击可视化选项卡查看图表结果。此图表按震级显示地震的数量。



另请参阅

命令

```
timechart
bin
sichart
```

博客

搜索命令优于 stats、chart 和 timechart

cluster

描述

`cluster` 命令根据事件之间的相似度对事件进行分组。除非您指定了不同的字段，否则 `cluster` 将根据 `_raw` 字段的内容对事件进行分组。默认的分组方法为：把事件细分为若干术语 (`match=termList`)，然后再计算事件之间的矢量。若您希望命令可以更有效地区分事件分组的标准，请为 `t` 设置较高的阈值。

`cluster` 命令的结果会向每个事件附加两个新字段。您可以使用 `countField` 和 `labelField` 参数来指定这些字段的命名，两个参数的默认值分别为 `cluster_count` 和 `cluster_label`。`cluster_count` 值为群集中事件的数量，或群集的大小。群集中的每个事件都分配到了所属群集的 `cluster_label` 值。例如，如果搜索返回 10 个群集，则会用 1 到 10 对这些群集进行标记。

语法

```
cluster [slc-options]...
```

可选参数

slc-options

语法: `t=<num> | delims=<string> | showcount=<bool> | countfield=<field> | labelfield=<field> |`

`field=<field> | labelonly=<bool> | match=(termlist | termset | ngramset)`

描述：用于配置简单日志群集 (SLC) 的选项。

SLC 选项

`t`

语法：`t=<num>`

描述：设置群集阈值，用于控制群集化的灵敏度。该值必须大于 0.0 且小于 1.0。阈值越接近 1，事件就必须越相似，这样才能考虑将其加入同一个群集中。

默认值：0.8

`delims`

语法：`delims=<string>`

描述：配置用于将原始字符串标签化的分隔符集。默认情况下，除了 0-9、A-Z、a-z 和 '_' 之外，任何符号都可以作为分隔符。

`showcount`

语法：`showcount=<bool>`

描述：如果 `showcount=false`，每个索引器在搜索头上进行群集化之前会对自己的事件进行群集化。如果 `showcount=false`，则事件计数未添加到事件。如果 `showcount=true`，则将记录每个群集的事件统计并在每个事件旁边加注计数。

默认值：`showcount=false`

`countfield`

语法：`countfield=<field>`

描述：若 `showcount=true` 为 `true`，要将群集大小写入其中的字段的名称。群集大小是指群集中事件的计数。

默认值：`cluster_count`.

`labelfield`

语法：`labelfield=<field>`

描述：要将群集数写入其中的字段名称。由于事件被分组成群集，因此对每个群集进行计数并用数字标记。

默认值：`cluster_label`

`field`

语法：`field=<field>`

描述：每个事件中要分析的字段的名称。

默认值：`_raw`

`labelonly`

描述：`labelonly=<bool>`

语法：选择是否保留传入事件并将其所属群集加注到每个事件中 (`labelonly=true`)，或仅输出群集字段为新事件 (`labelonly=false`)。如果 `labelonly=false`，则会输出一个群集列表，包含描述性事件及所包含的事件数量。

默认值：`false`

`match`

语法：`match=(termlist | termset | ngramset)`

描述：选择用于确定事件之间相似性的方法。`termlist` 把字段细分为若干词，并要求术语顺序完全相同。`termset` 允许使用未经排序的术语组。`ngramset` 比较三字母组（由 3 个字符组成的子字符串）组。`ngramset` 作用于较大字段值时明显变慢，作用于简短的非文字字段，如 `punct` 时效果最好。

默认值：`termlist`

用法

`cluster` 命令是一个流命令或数据集处理命令，具体取决于使用命令指定的参数类型。请参阅“命令类型”。

使用 `cluster` 命令查找数据中的常见或罕见事件。例如，如果您正在调查 IT 问题，可使用 `cluster` 命令来查找异常事件。在本例中，异常事件是指那些没有被归到大群集或包含很少事件的群集中的事件。或者，如果您要搜索错误，可使用 `cluster` 命令来查看大约有多少种不同类型错误以及数据中有哪些常见错误类型。

示例

示例 1

快速返回您的 Splunk 部署出现错误的任何内容。您的角色必须有访问内部索引的相应功能。

```
index=_internal source=*splunkd.log* log_level!=info | cluster showcount=t | table cluster_count _raw | sort -cluster_count
```

本搜索利用了 Splunk 软件在 `_internal` 索引中记录有关它操作的内容。它将返回所有日志，其中 `log_level` 是 DEBUG、WARN、ERROR、FATAL，并集合到一起。然后按每个群集的事件计数将群集排序。

统计选项卡中显示的结果如下所示：

cluster_count	raw
303010	03-20-2018 09:37:33.806 -0700 ERROR HotDBManager - Unable to create directory /Applications/Splunk/var/lib/splunk/_internaldb/db/hot_v1_49427345 because No such file or directory
151506	03-20-2018 09:37:33.811 -0700 ERROR pipeline - Uncaught exception in pipeline execution (indexer) - getting next event
16390	04-05-2018 08:30:53.996 -0700 WARN SearchResultsMem - Failed to append to multival. Original value not converted successfully to multival.
486	03-20-2018 09:37:33.811 -0700 ERROR BTreecP - failed: failed to mkdir /Applications/Splunk/var/lib/splunk/fishbucket/splunk_private_db/snapshot.tmp: No such file or directory
216	03-20-2018 09:37:33.814 -0700 WARN DatabaseDirectoryManager - idx=_internal Cannot open file='/Applications/Splunk/var/lib/splunk/_internaldb/db/.bucketManifest99454_1652919429_tmp' for writing bucket manifest (No such file or directory)
216	03-20-2018 09:37:33.814 -0700 ERROR SearchResultsWriter - Unable to open output file: path=/Applications/Splunk/var/lib/splunk/_internaldb/db/.bucketManifest99454_1652919429_tmp error=No such file or directory

示例 2

搜索不能群集到大组中的事件。

```
... | cluster showcount=t | sort cluster_count
```

此代码将返回事件群集，并使用 `sort` 命令根据群集大小（即 `cluster_count` 的值）升序排列各群集。由于它们不能群集到大组中，因此可以将其视为罕见或不常见事件。

示例 3

将相似的错误事件群集并搜索最常出现的错误类型。

```
error | cluster t=0.9 showcount=t | sort - cluster_count | head 20
```

此代码将在索引中搜索包含 "error" 一词的事件，如果这些事件相似，则将其群集。使用 `sort` 命令可根据群集大小（即 `cluster_count`）降序排列各事件，因此，最大的群集将最先显示。然后，使用 `head` 命令显示最大的二十个群集。既然已经找到数据中最常见的错误类型，您现在就可以深入调查以找出这些错误的根本原因。

示例 4

使用 `cluster` 命令可以查看数据概览。如果您有大量数据，那么可以基于一段较短的时间范围（例如，15 分钟或 1 小时）运行以下搜索，或将其限定为某个来源类型或索引。

```
... | cluster labelonly=t showcount=t | sort - cluster_count, cluster_label, _time | dedup 5 cluster_label
```

此搜索会基于相似性将事件归组在一起并显示来自每个群集的少数几个事件，以帮助您更多地了解您的数据。它使用 `labelonly=t` 将每个事件保留在群集内，并为事件附加 `cluster_label`。使用 `sort` 命令可以依次按照大小（即 `cluster_count`）、`cluster_label` 和事件的索引时间戳（即 `_time`）降序排列搜索结果。然后再使用 `dedup` 命令显示每个群集中的前五个事件，并运用 `cluster_label` 区分每个群集。

另请参阅

[anomalies](#), [anomalousvalue](#), [kmeans](#), [outlier](#)

cofilter

描述

使用此命令确定 `field1` 和 `field2` 值一起出现的次数。

此命令一键式执行合作式筛选分析，以提供参考建议。若用户字段（field1）和商品字段（field2）已给定，则会找出每件商品的常见程度。即，此命令会计算出总和（A 买了 X 和 Y），其中 A 指唯一的用户，X 和 Y 指两种不同的商品。

语法

```
cofilter <field1> <field2>
```

必要参数

field1

语法：<field>

描述：字段的名称。

field2

语法：<field>

描述：字段的名称。

用法

cofilter 命令是一个转换命令。请参阅“命令类型”。

示例

示例 1

查找 user 和 item 的协同过滤。必须先指定 user 字段然后再指定 item 字段。输出每对商品的事件，包含：第一个商品及其受欢迎程度，第二个商品及其受欢迎程度，以及这对商品的受欢迎程度。

让我们先来看一个会创建少量结果的简单搜索：

```
| makeresults | eval user="a b c a b c a b c" | makemv user | mvexpand user | streamstats count
```

统计选项卡中显示的结果如下所示：

_time	计数	用户
2020/2/19 21:17:54	1	a
2020/2/19 21:17:54	2	b
2020/2/19 21:17:54	3	c
2020/2/19 21:17:54	4	a
2020/2/19 21:17:54	5	b
2020/2/19 21:17:54	6	c
2020/2/19 21:17:54	7	a
2020/2/19 21:17:54	8	b
2020/2/19 21:17:54	9	c

带取模（%）运算符的 eval 命令用于创建 item 字段：

```
| makeresults | eval user="a b c a b c a b c" | makemv user | mvexpand user | streamstats count | eval item = count % 5
```

结果形式如下所示：

_time	计数	项目	用户
2020/2/19 21:17:54	1	1	a
2020/2/19 21:17:54	2	2	b

2020/2/19 21:17:54	3	3	c
2020/2/19 21:17:54	4	4	a
2020/2/19 21:17:54	5	0	b
2020/2/19 21:17:54	6	1	c
2020/2/19 21:17:54	7	2	a
2020/2/19 21:17:54	8	3	b
2020/2/19 21:17:54	9	4	c

将 `cofilter` 命令添加到搜索中，以确定对于每一对 `item` 值，出现了多少个 `user` 值。

```
| makeresults | eval user="a b c a b c a b c" | makemv user | mvexpand user | streamstats count | eval item = count % 5 | cofilter user item
```

结果形式如下所示：

项目 1	项目 1 用户数	项目 2	项目 2 用户数	值对数
1	2	2	2	1
1	2	3	2	1
1	2	4	2	2
2	2	3	2	1
2	2	4	2	1
2	2	0	1	1
3	2	4	2	1
3	2	0	1	1

另请参阅

`associate`, `correlate`

collect

描述

将搜索结果添加到指定的摘要索引中。在调用 `collect` 命令前必须创建摘要索引。

您不需要知道如何使用 `collect` 来创建和使用摘要索引，但是它可以提供帮助。有关摘要索引的概述，请参阅《知识管理器手册》中的“使用摘要索引提高报表效率”。

语法

```
collect index=<string> [<arg-options>...]
```

必要参数

index

语法： `index=<string>`

描述： 添加了事件的摘要索引的名称。此索引必须先于事件添加之前创建。此索引非自动创建。

可选参数

arg-options

语法： `addtime=<bool> | file=<string> | spool=<bool> | marker=<string> | output_format [raw | hec] | testmode=<bool> | run_in_preview=<bool> | host=<string> | source=<string> | sourcetype=<string>`

描述：collect 命令的可选参数。有关每个选项的说明，请参见 **arg-options** 部分。

arg-options

addtime

语法：addtime=<bool>

描述：使用此选项可指定是否为每个事件加上时间字段作为前缀。某些命令返回没有 raw 字段的结果（例如 stats、chart、timechart 命令）。如果您指定 addtime=false，则 Splunk 软件将对摘要行中按随机顺序出现的字段应用一般日期检测。如果您指定 addtime=true，则 Splunk 软件使用搜索时间范围 info_min_time。该时间范围通过 sistats 命令或 _time 添加。Splunk 软件将基于找到的第一个字段添加时间字段：info_min_time、_time 或 now()。

当 output_format=hec 时，此选项无效。

默认值：true

file

语法：file=<string>

描述：您想要将事件写入其内的文件名。通过指定 file=\$timestamp\$ 或 file=\$random\$，您可以在文件名中使用时间戳或任何数字。

用法：和 "index=" 一起使用时，需要把 ".stash" 添加到文件名的末尾。否则，将把数据添加到主索引。

默认值：<random-number>_events.stash

host

语法：host=<string>

描述：要为事件指定的主机的名称。

当 output_format=hec 时，此选项无效。

marker

语法：marker=<string>

描述：要附加到写出的每个事件的字符串（通常为键值对）。每个键值对必须以逗号和空格分隔开。

如果此值包含空格或逗号，则必须使用转义引号。例如，若键值对为 search_name=vpn starts and stops，您必须将其更改 为 search_name=\"vpn starts and stops\"。

当 output_format=hec 时，此选项无效。

output_format

语法：output_format=[raw | hec]

描述：指定摘要索引的输出格式。如果设置为 raw，则使用传统的非结构化日志样式摘要索引 stash 输出格式。

如果设置为 hec，则会生成 HTTP 事件收集器 (HEC) JSON 格式的输出：

- 为 stash 文件建立索引后，所有字段都会自动建立索引。
- 写入 var/spool/splunk 路径的文件以 .stash_hec 而非 .stash 结尾。
- 允许原始数据的数据来源、来源类型和主机直接用于摘要索引中。请勿将这些字段重新映射到 extract_host/extracted_sourcetype/... 路径。
- 原始数据中的 index 和 splunk_server 字段将被忽略。
- 当 output_format=hec 时，不能使用 addtime、host、marker、source 或 sourcetype 选项。

默认：raw

run_in_preview

语法：run_in_preview=<bool>

描述：控制在预览生成期间是否启用 collect 命令。通常，您不希望将预览结果插入摘要索引中，因此可设置 run-in-preview=false。在某些情况下，比如在搜索中使用了自定义搜索命令时，您可能需要启用此选项，以确保生成正确的可索引摘要的预览。

默认值：false

spool

语法：spool=<bool>

描述：若设置为 true，摘要索引文件将写入 Splunk 后台打印目录，在此目录中，系统将自动为文件创建索引。若设置为 false，文件将写入 \$SPLUNK_HOME/var/run/splunk 目录。除非进一步执行了某种形式的自动化或管理，否则文件将留在此目录中。如果有 Splunk Enterprise，则可以使用此命令排除摘要索引的故障，方法是将输出文件转储到磁盘的相应位置，其中它不会作为数据进行插入。

默认值：true

source

语法：source=<string>

描述：要为事件指定的数据来源的名称。

当 output_format=hec 时，此选项无效。

sourcetype

语法：sourcetype=<string>

描述：要为事件指定的来源类型的名称。通过在存储外部指定源类型，您将使用许可证。

当 output_format=hec 时，此选项无效。

默认值: stash

testmode

语法: testmode=<bool>

描述: 在测试模式和实模式之间切换。在测试模式中，不会将结果写入新索引中，但会修改搜索结果，就像已经发送到索引一样。

默认值: false

用法

事件被写入一个名称格式为 `random-num_events.stash` 的文件中，除非覆盖，否则在 Splunk 部署所监视的目录中。若事件包含 `_raw` 字段，则保存该字段。若事件不含 `_raw` 字段，则通过将所有字段连接成以逗号分隔的 `key=value` 对列表来创建该字段。

`collect` 命令也适用于时间范围为所有时间的实时搜索。

未含时间戳的事件

若您将 `collect` 命令应用于未含时间戳的事件，该命令将使用搜索范围内最早（或最小）的时间为所有事件指定一个时间。例如，如果您在过去四小时使用 `collect` 命令（范围：-4h 到 +0h），该命令会分配一个时间戳，该时间戳早于搜索启动时间四小时。时间戳应用于没有时间戳的所有事件。

如果您使用具有时间范围为所有时间的 `collect` 命令，并且事件没有时间戳，则当前系统时间用于时间戳。

有关不带时间戳摘要索引数据的更多信息，请参阅《知识管理器手册》中的“使用摘要索引以提高报表效率”。

将事件复制到不同索引

您可以使用 `collect` 命令将搜索结果复制到其他索引中。构建返回您想要复制的数据的搜索，并通过管道将结果传递到 `collect` 命令。例如：

```
index=foo | ... | collect index=bar
```

此搜索将结果写入到 `bar` 索引。源类型更改为 `stash`。

您可以用 `collect` 命令指定源类型。但是，指定源类型计入许可证数量，就像您重新索引数据一样。

示例

1. 将 "download" 事件置于名为 "download count" 的索引中

```
eventtype=download | collect index=downloadcount
```

2. 收集有关 VPN 连接和断连的统计信息

您想按国家收集有关 VPN 连接和断连的每小时的统计信息。

```
index=mysummary | geoip REMOTE_IP | eval country_source;if(REMOTE_IP_country_code=="US","domestic","foreign") | bin _time span=1h | stats count by _time,vpn_action,country_source | addinfo | collect index=mysummary marker="summary_type=vpn,summary_span=3600, summary_method=bin, search_name=\"vpn starts and stops\""
```

`addinfo` 命令确保搜索结果中包含指定何时运行该搜索以填充这些特定索引值的字段。

另请参阅

命令

```
overlap  
sichart  
sirare  
sistats  
sitimechart  
sitop  
tscollect
```

问答

有什么问题吗？请访问 Splunk Answers，查看 Splunk 社区有哪些与使用 `collect` 命令相关的问题和解答。

Concurrency

描述

Concurrency 能测量事件数量，这些事件具有与每个事件的启动重叠的间隔。或者，此测量代表每个特定事件启动时正在运行的事件总数，包括该事件本身。此命令不会计量整个时间跨度内与特定事件重叠的事件总数。

语法

```
concurrency duration=<field> [start=<field>] [output=<field>]
```

必要参数

duration

语法: duration=<field>

描述: 代表时间跨度的字段。此字段必须是与 `start` 字段具有相同单位的数字。例如，`transaction` 命令生成的持续时间字段以秒为单位（请参阅示例 1），该字段可与 `_time` 的默认值（亦以秒为单位）结合使用。

可选参数

start

语法: start=<field>

描述: 代表开始时间的字段。

默认值: `_time`

output

语法: output=<field>

描述: 要写入生成的并发事件数的字段。

默认值: "concurrency"

用法

如果 `X.start` 在 `Y.start` 与 `(Y.start + Y.duration)` 之间，则事件 `X` 与事件 `Y` 是并发事件。

如果您的事件具有代表事件完成的时间和代表完成前时间的跨度，您将需要从 `concurrency` 命令前的启动时间减去持续时间：

```
... | eval new_start = start - duration | concurrency start=new_start duration=duration
```

限制

对于重叠项目的数量有限制。如果跟踪到的最大并发数超过在 `limits.conf` 中 `[concurrency]` 段落的 `max_count`，在 UI/搜索结果中将显示一个警告，并且这些值将被钳制，这使得它们可能为不正确的值。此限制默认为 1000000 或一千万。

基本示例

1. 确定重叠 HTTP 请求的数量

确定每个 http 请求开始时，访问 `splunkd` 的浏览器中未完成的重叠 HTTP 请求的数量。

这依赖于下列事实，已记录消息的时间戳是收到请求的时间，并且 '`spent`' 字段是处理请求花费的毫秒数。一如既往，您必须是 '`admin`' 用户，或为了访问 `_internal` 索引已经改变了您的角色方案。

```
index=_internal sourcetype=splunkd_ui_access | eval spent_in_seconds = spent / 1000 | concurrency duration=spent_in_seconds
```

2. 计算并发事件的数量

计算每个事件的并发事件数并以字段 '`foo`' 形式发出：

```
... | concurrency duration=total_time output=foo
```

3. 使用现有字段指定开始时间和持续时间

使用 '`et`' 字段作为开始时间，'`length`' 作为持续时间，计算并发事件的数量。

```
... | concurrency duration=length start=et
```

延伸示例

1. 对在同一时间发生的交易计数

此示例使用搜索教程中的示例数据。要在您自己的 Splunk 实例中尝试此示例，您必须下载示例数据并遵循操作以将教程数据导入 Splunk。当运行搜索时，请使用时间范围所有时间。

使用持续时间或交易跨度统计同时发生的其他交易的数量。

```
sourcetype=access_* | transaction JSESSIONID clientip startswith="view" endswith="purchase" | concurrency duration=duration | eval duration=tosstring(duration,"duration")
```

- 此搜索将有着相同 JSESSIONID 和 clientip 值的事件分组为交易。如果事件包含字符串 "view"，事件是交易的开始。如果事件包含字符串 "purchase"，事件是交易的最后事件。
- transaction 命令返回一个名为持续时间的字段。
- 然后再通过通配符把各交易传递至 concurrency 命令，该命令将根据时间戳和交易的 duration 统计同时发生的事件数量。
- 该搜索还会使用 eval 命令和 tostring() 函数把 duration 字段值的格式重新设置为可读性更高的格式，即 HH:MM:SS。

<隐藏字段	所有字段	i 时间	事件
选定字段			
a host 3			
a source 3			
a sourcetype 1			
感兴趣的字段			
a action 5			
a bytes 100+			
a categoryid 8			
a clientip 100+			
# closed_tn 1			
# concurrency 4			
# date_hour 24			
# date_mday 8			
# date_minute 60			
a date_month 2			
a date_second 60			
a date_wday 7			
# date_year 1			
a date_zone 1			
a duration 14			
# eventcount 12			
# field_match_sum 12			
a file 14			
a ident 1			
a index 1			

要查看每个交易中 JSESSIONID、clientip、并发和持续时间字段的值：

- 在感兴趣的字段列表中，单击字段名称。
- 在信息框中，为已选单击是。
- 选择感兴趣的字段列表中的下一个字段。信息框自动刷新。为已选单击是。
- 对您想要在结果列表中显示的每个字段重复这些步骤。结果应如下图所示：

i	时间	事件
>	18/05/28 18:27:50.000	74.125.19.106 - - [28/May/2018:18:27:50] "GET /oldlink?itemId=EST-12&JSESSIONID=SD10SL4FF4ADFF4976 HTTP 1.1" 400 1224 "http://www.buttercupgames.com/cart.do?action=view&itemId=EST-12" "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/536.5 (KHTML, like Gecko) Chrome/19.0.1084.46 Safari/536.5" 774 74.125.19.106 - - [28/May/2018:18:27:51] "POST /cart/error.do?msg=CreditDoesNotMatch&JSESSIONID=SD10SL4FF4ADFF4976 HTTP 1.1" 200 2934 "http://www.buttercupgames.com/cart.do?action=purchase&itemId=EST-26" "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/536.5 (KHTML, like Gecko) Chrome/19.0.1084.46 Safari/536.5" 866
		JSESSIONID = SD10SL4FF4ADFF4976 clientip = 74.125.19.106 concurrency = 1 duration = 00:00:01 host = www3 source = tutorialdata.zip:/www3/access.log sourcetype = access_combined_wcookie
>	18/05/28 18:44:42.000	175.44.24.82 - - [28/May/2018:18:44:42] "POST /cart.do?action=view&itemId=EST-21&productId=WC-SH-G04&JSESSIONID=SD7SL9FF5ADFF5066 HTTP 1.1" 200 675 "http://www.buttercupgames.com/oldlink?itemId=EST-21" "Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; WOW64; Trident/5.0; BOIE9;ENUS)" 142 175.44.24.82 - - [28/May/2018:18:44:44] "POST /cart.do?action=purchase&itemId=EST-12&JSESSIONID=SD7SL9FF5ADFF5066 HTTP 1.1" 200 2399 "http://www.buttercupgames.com/cart.do?action=addtocart&itemId=EST-12&categoryId=STRATEGY&productId=DC-SG-G02" "Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; WOW64; Trident/5.0; BOIE9;ENUS)" 594
		JSESSIONID = SD7SL9FF5ADFF5066 clientip = 175.44.24.82 concurrency = 1 duration = 00:00:02 host = www2 source = tutorialdata.zip:/www2/access.log sourcetype = access_combined_wcookie

2. 对在同一时间出现的购买计数

此示例使用搜索教程中的示例数据。要在您自己的 Splunk 实例中尝试此示例，您必须下载示例数据并遵循操作以将教程数据导入 Splunk。当运行搜索时，请使用时间范围所有时间。

使用每个购买的时间来统计在同一时间发生的不同购买的数量。

```
sourcetype=access_* action=purchase | delta _time AS timeDelta p=1 | eval timeDelta=abs(timeDelta) | concurrency duration=timeDelta
```

- 本搜索使用 `delta` 命令和 `_time` 字段计算一个购物事件 (`action=purchase`) 与上一个购物事件之间的时间。
- 本搜索将此时间上的更改重命名为 `timeDelta`。
- `timeDelta` 的某些值为负数。由于 `concurrency` 命令无法处理负值，因此使用 `eval` 命令将 `timeDelta` 重新定义为其绝对值 (`abs(timeDelta)`)。
- 然后再把 `timeDelta` 用作 `duration` 来计算并发事件。

< 隐藏字段	所有字段	i	时间	事件
选定字段 <i>a clientip</i> 100+ # concurrency 2 <i>a host</i> 3 <i>a JSESSIONID</i> 100+ <i>a source</i> 3 <i>a sourcetype</i> 1 # timeDelta 100+			> 18/05/28 18:22:21.000	209.160.24.63 - [28/May/2018:18:22:21] "POST /cart.do?action=purchase&itemId=EST-21&JSESSIONID=SD0SL6FF7ADFF4953 HTTP 1.1" 200 486 "http://www.buttercupgames.com/cart.do?action=addtocart&itemId=EST-21&categoryId=STRATEGY&produ ctId=FS-SG-G03" "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/536.5 (KHTML, like Gecko) Chrome/19.0.1084.46 Saf ari/536.5" 293 JSESSIONID = SD0SL6FF7ADFF4953 clientip = 209.160.24.63 concurrency = 1 host = www1 source = tutorialdata.zip:/www1/access.log sourcetype = access_combined_wcookie timeDelta = 1
感兴趣的字段 <i>a action</i> 1 # bytes 100+ <i>a categoryId</i> 8 # date_hour 24 # date_mday 8 # date_minute 60 <i>a date_month</i> 2 # date_second 60			> 18/05/28 18:22:22.000	209.160.24.63 - [28/May/2018:18:22:22] "POST /cart/success.do?msg=CreditDoesNotMatch&JSESSIONID=SD0SL6FF7ADFF4953 HTTP 1.1" 200 3280 "h ttp://www.buttercupgames.com/cart.do?action=purchase&itemId=EST-21" "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/536.5 (KHTML, like Gecko) Chrome/19.0.1084.46 Safari/536.5" 952 JSESSIONID = SD0SL6FF7ADFF4953 clientip = 209.160.24.63 concurrency = 1 host = www1 source = tutorialdata.zip:/www1/access.log sourcetype = access_combined_wcookie timeDelta = 256
			> 18/05/28 18:26:38.000	112.111.162.4 - [28/May/2018:18:26:38] "POST /cart/error.do?msg=CreditDoesNotMatch&JSESSIONID=SD7SL8FF5ADFF4964 HTTP 1.1" 200 1232 "http://www.buttercupgames.com/cart.do?action=purchase&itemId=EST-18" "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/536.5 (KHTML, like Gecko) Chrome/19.0.1084.52 Safari/536.5" 841 JSESSIONID = SD7SL8FF5ADFF4964 clientip = 112.111.162.4 concurrency = 1 host = www1 source = tutorialdata.zip:/www1/access.log sourcetype = access_combined_wcookie timeDelta = 72

3. 使用连续交易之间的时间计算交易

此示例使用搜索教程中的示例数据。要在您自己的 Splunk 实例中尝试此示例，您必须下载示例数据并遵循操作以将教程数据导入 Splunk。当运行搜索时，请使用时间范围所有时间。

使用连续交易之间的时间来计算在同一时间发生的交易数。

```
sourcetype=access_* | transaction JSESSIONID clientip startswith="view" endswith="purchase" | delta _time AS timeDelta p=1 |  
eval timeDelta=abs(timeDelta) | concurrency duration=timeDelta | eval timeDelta=toString(timeDelta,"duration")
```

- 此搜索将有着相同 `JSESSIONID` 和 `clientip` 值的事件分组为交易。如果事件包含字符串 `"view"`，事件是交易的开始。如果事件包含字符串 `"purchase"`，事件是交易的最后事件。
- `transaction` 命令返回一个名为 **持续时间** 的字段。
- 然后通过管道符把各交易传递给 `delta` 命令，该命令使用 `_time` 字段计算某一笔交易与上一笔交易之间的时间。
- 本搜索将此时间上的更改重命名为 `timeDelta`。
- `timeDelta` 的某些值为负数。由于 `concurrency` 命令无法处理负值，因此使用 `eval` 命令将 `timeDelta` 重新定义为其绝对值 (`abs(timeDelta)`)。
- 然后再把此 `timeDelta` 用作 `duration` 来计算并发交易。

1,712 个事件 (18/06/05 12:09:37:000 之前) 无事件采样 ▾

任务 ▾ 智能模式 ▾

事件 (1,712) 模式 统计信息 可视化

设定时间线的格式 ▾ - 缩小 + 缩放到所选区域 × 取消选择 每列1天

列表 ▾ 格式 每页 20 个 ▾ < 上一个 1 2 3 4 5 6 7 8 ... 下一步 >

所有字段	时间	事件
选定字段 a clientip 100+ # concurrency 2 # duration 14 a host 3 a JSESSIONID 100+ a source 3 a sourcetype 1 a timeDelta 100+	> 18/05/28 18:27:50.000	74.125.19.106 - [28/May/2018:18:27:50] "GET /oldlink?itemId=EST-12&JSESSIONID=SD10SL4FF4ADFF4976 HTTP 1.1" 400 1224 "http://www.buttercupgames.com/cart.do?action=view&itemId=EST-12" "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/536.5 (KHTML, like Gecko) Chrome/19.0.1084.46 Safari/536.5" 774 74.125.19.106 - [28/May/2018:18:27:51] "POST /cart/error.do?msg=CreditDoesNotMatch&JSESSIONID=SD10SL4FF4ADFF4976 HTTP 1.1" 200 2934 "http://www.buttercupgames.com/cart.do?action=purchase&itemId=EST-26" "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/536.5 (KHTML, like Gecko) Chrome/19.0.1084.46 Safari/536.5" 866 JSESSIONID = SD10SL4FF4ADFF4976 clientip = 74.125.19.106 concurrency = 1 duration = 1 host = www3 source = tutorialdata.zip://www3/access.log sourcetype = access_combined_wcookie timeDelta = 00:16:52
感兴趣的字段 a action 5 a bytes 100+ a categoryid 8 # closed_txn 1 # date_hour 24 # date_mday 8	> 18/05/28 18:44:42.000	175.44.24.82 - [28/May/2018:18:44:42] "POST /cart.do?action=view&itemId=EST-21&productId=WC-SH-G04&JSESSIONID=SD7SL9FF5ADFF5066 HTTP 1.1" 200 675 "http://www.buttercupgames.com/oldlink?itemId=EST-21" "Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; WOW64; Trident/5.0; BOIE9;ENUS)" 142 175.44.24.82 - [28/May/2018:18:44:44] "POST /cart.do?action=purchase&itemId=EST-12&JSESSIONID=SD7SL9FF5ADFF5066 HTTP 1.1" 200 2399 "http://www.buttercupgames.com/cart.do?action=addtocart&itemId=EST-12&categoryId=STRATEGY&productId=DC-SG-G02" "Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; WOW64; Trident/5.0; BOIE9;ENUS)" 594 JSESSIONID = SD7SL9FF5ADFF5066 clientip = 175.44.24.82 concurrency = 1 duration = 2 host = www2 source = tutorialdata.zip://www2/access.log sourcetype = access_combined_wcookie timeDelta = 00:15:12

另请参阅

timechart

contingency

描述

在统计中，应变表用于记录和分析两个或更多（通常为分类）变量之间的关系。可以根据应变表计算相关性或无关性的很多指标，如 *phi* 系数或 *Cramer V*。

您可以使用 `contingency` 命令制作应变表，此表在本示例中为您数据中两个字段值的同现矩阵。矩阵中的每个单元格将显示同时包含两个交叉表字段值的事件的计数。这表示该表的第一行和第一列由这两个字段的值组成。表中的每个单元格包含一个数字，该数字代表包含该行列组合中的两个字段值的事件的计数。

如果两个字段之间存在某种关系或模式，您只需通过分析表中的信息即可发现。例如，如果两行中的列值存在明显差异（反之亦然），则两个字段之间存在应变性（它们具有相关性）。如果不存在应变性，则两个字段互不相关。

语法

`contingency [<contingency-options>...]<field1> <field2>`

必要参数

<field1>

语法：<field>

描述：任意字段。无法在字段名称中指定通配符字符。

<field2>

语法：<field>

描述：任意字段。无法在字段名称中指定通配符字符。

可选参数

`contingency-options`

语法：<maxopts> | <mincover> | <usetotal> | <totalstr>

描述：应变表的选项。

应变选项

`maxopts`

语法: maxrows=<int> | maxcols=<int>

描述: 指定要显示的最大行数或列数。若字段的非重复值数量超出此最大值，将忽略最不常用的值。值 0 表示行数或列数的最大限制。此限制来自 maxvalues 设置（位于 limits.conf 文件内的 [ctable] 段落中）。

默认值: 1000

mincover

语法: mincolcover=<num> | minrowcover=<num>

描述: 指定您想在输出表中显示的每列或行数值的百分比。构建完成的表格将包含充足的行或列，以达到显示的值与每行或列总值之间的该比例。若达到这些值，最大行或列将优先。

默认值: 1.0

usetotal

语法: usetotal=<bool>

描述: 指定是否添加行、列和完整总计。

默认值: true

totalstr

语法: totalstr=<field>

描述: 用于行和列总计的字段名。

默认值: TOTAL

用法

contingency 命令是一个转换命令。请参阅“命令类型”。

此命令用于构建两个字段的应变表。若字段中含很多值，您可以使用 maxrows 和 maxcols 参数限制行和列的数量。

Totals

默认情况下，应变表将显示行总计、列总计以及表中显示的所有事件的总计。如果您不想要在结果中显示总计，请使用 contingency 命令包含 usetotal=false 参数。

空值

空字符串 ("") 值将在结果表中表示为 EMPTY_STR。

限制

对于 maxrows 或 maxcols 的值有限制，这意味着对于每个字段超过 1000 个的值将不被使用。

示例

1. 构建最近数据应变表

此搜索使用从 USGS 地震网站下载的近期地震数据。该数据是一个逗号分隔的 ASCII 文本文件，其中包含每次记录的地震的震级 (mag)、坐标 (经度、纬度)、区域 (地点) 等。

您可以从 USGS 地震源下载当前 CSV 文件，然后将文件上载到 Splunk 实例。此示例使用过去 30 天的所有地震数据。当运行搜索时，请使用时间范围所有时间。

您想要构建一个应变表，以查看最近发生地震的震级与深度的关系。从一个简单的搜索开始。

```
source=all_month.csv | contingency mag depth | sort mag
```

Magnitude 和 Depth 字段含大量的值，结果将是一个非常庞大的表。震级值在第一列中。深度值在第一行中。列表按震级排序。

结果将显示在“统计”选项卡中。下表仅显示小部分搜索返回的结果表。

mag	10	0	5	35	8	12	15	11.9	11.8	6.4	5.4	8.2	6.5	8.1	5.6	10.1	9	8.5	9.8	8.7	7.9
-0.81	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
-0.59	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
-0.56	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
-0.45	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

-0.43	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
-------	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

如您所见，地震可能有负的震级。只有发生了与震级和深度相匹配的地震时才会在表中显示计数。

要构建更多可用的应变表，您应重新设置震级和深度字段值的格式。将震级和深度分组到各范围。

```
source=all_month.csv | eval Magnitude=case(mag<=1, "0.0 - 1.0", mag>1 AND mag<=2, "1.1 - 2.0", mag>2 AND mag<=3, "2.1 - 3.0", mag>3 AND mag<=4, "3.1 - 4.0", mag>4 AND mag<=5, "4.1 - 5.0", mag>5 AND mag<=6, "5.1 - 6.0", mag>6 AND mag<=7, "6.1 - 7.0", mag>7, "7.0+") | eval Depth=case(depth<=70, "Shallow", depth>70 AND depth<=300, "Mid", depth>300 AND depth<=700, "Deep") | contingency Magnitude Depth | sort Magnitude
```

此搜索结合使用 `eval` 命令和 `case()` 函数重新定义 `Magnitude` 和 `Depth` 的值，并将它们存储到一个值范围内。例如，将 `Depth` 值重新定义为 "Shallow"、"Mid" 或 "Deep"。使用 `sort` 命令按震级对结果进行排序。否则按行总数对结果进行排序。

统计选项卡中显示的结果如下所示：

震级	Shallow	Mid	Deep	TOTAL
0. 0 - 1. 0	3579	33	0	3612
1. 1 - 2. 0	3188	596	0	3784
2. 1 - 3. 0	1236	131	0	1367
3. 1 - 4. 0	320	63	1	384
4. 1 - 5. 0	400	157	43	600
5. 1 - 6. 0	63	12	3	78
6. 1 - 7. 0	2	2	1	5
TOTAL	8788	994	48	9830

这个月发生了许多地震。震级大的地震是否震源深度也大于震级小的地震呢？并非如此。该表显示最近发生的大部分地震震级都属于浅源范围。震源深度属于中高范围的地震相当少。在本数据集中，深源地震的震源全都属于中等深度。

2. 识别 Splunk 部署中的潜在组件问题

确定是否存在可能导致 Splunk 部署中出现问题的组件。构建应变表以查看 `log_level` 值和 `component` 值之间是否存在关系。使用时间范围所有时间运行搜索并限制返回的列数。

```
index=_internal | contingency maxcols=5 log_level component
```

您的结果应类似如下所示：

新搜索							另存为	关闭
index=_internal contingency log_level component maxcols=5							所有时间	搜索
248,846 个事件 (18/11/03 9:20:15.000 之前) 无事件采样							任务	智能模式
事件	模式	统计信息 (6)	可视化					
每页 20 个	格式	预览						
log_level	指标	PeriodicHealthReporter	root	ArchiveProcessor	LicenseUsage	TOTAL		
INFO	212216	23128	556	548	367	238295		
WARN	0	0	0	0	0	125		
WARNING	0	0	0	0	0	36		
ERROR	0	0	0	0	0	19		
FATAL	0	0	0	0	0	1		
TOTAL	212216	23128	556	548	367	238476		

上述结果显示了可能导致 Splunk 部署中出现问题的组件。`component` 字段有 50 多个值。在此搜索中，`maxcols` 参数用于显示拥有最高值的 5 个组件。

另请参阅

associate, correlate

convert

描述

convert 命令把搜索结果中的字段值转换成数字值。除非您使用 AS 子句，否则新值将替换原始值。

或者，您可以使用评估函数，如 strftime()、strptime() 或 tonumber() 来转换字段值。

语法

```
convert [timeformat=string] (<convert-function> [AS <field>] )...
```

必要参数

<convert-function>

语法：auto() | ctime() | dur2sec() | memk() | mktime() | mstime() | none() | num() | rmcomma() | rmunit()
描述：转换要使用的函数。

可选参数

timeformat

语法：timeformat=<string>
描述：指定转换的时间字段的输出格式。timeformat 选项供 ctime 和 mktime 函数使用。有关格式选项的列表和描述，请参阅《搜索参考》中的“常见的时间格式变量”。
默认值：%m/%d/%Y %H:%M:%S. 请注意，此默认值不符合区域设置。

<field>

语法：<string>
描述：使用您指定的名称新建一个字段，以将转换好的值放入其中。原始字段和值保持不变。

转换函数

auto()

语法：auto(<wc-field>)
描述：使用最佳转换法将字段自动转换为数字。注意，如果无法使用已知转换类型转换某特定字段的所有值，则字段保持不变，不对该字段执行任何转换。可以在字段名称中使用通配符（*）字符。

ctime()

语法：ctime(<wc-field>)
描述：将 epoch 时间转换为可读的 ascii 代码时间。使用 timeformat 选项指定目标代码时间的精确格式。可以在字段名称中使用通配符（*）字符。

dur2sec()

语法：dur2sec(<wc-field>)
描述：将持续时间格式 “[D+]HH:MM:SS” 转换为秒。可以在字段名称中使用通配符（*）字符。

memk()

语法：memk(<wc-field>)
描述：接受正数（整数或浮点数），后跟可选的 “k”、“m” 或 “g”。字母 k 表示 KB，m 表示 MB，g 表示 GB。若未指定任何字母，则假定为 KB。输出字段是表示 KB 数量的数字。负值会导致数据的不一致性。可以在字段名称中使用通配符（*）字符。

mstime()

语法：mstime(<wc-field>)
描述：将可读时间字符串转换为 Epoch 时间。使用 timeformat 选项指定可读时间字符串的精确格式。可以在字段名称中使用通配符（*）字符。

none()

语法：none(<wc-field>)
描述：将 [MM:]SS.SSS 格式转换为秒。可以在字段名称中使用通配符（*）字符。

描述: 如果有其他通配符，指出不应转换匹配字段。可以在字段名称中使用通配符（*）字符。

`num()`

语法: `num(<wc-field>)`

描述: 与 `auto()` 相似，将删除不可转换值之外的值。可以在字段名称中使用通配符（*）字符。

`rmcomma()`

语法: `rmcomma(<wc-field>)`

描述: 删除值中的所有逗号，如 `rmcomma(1,000,000.00)` 会返回 `1000000.00`。可以在字段名称中使用通配符（*）字符。

`rmunit()`

语法: `rmunit(<wc-field>)`

描述: 查找值开头的数字，并删除随后的文本。可以在字段名称中使用通配符（*）字符。

用法

`convert` 命令属于可分配的流命令。请参阅“命令类型”。

基本示例

1. 将所有字段值转换为数字值

使用 `auto` 转换函数将所有字段值转换为数字值。

```
... | convert auto(*)
```

2. 转换字段值，指定字段中的值除外

将每个字段值转换为数字值，字段 `foo` 中的值除外。使用 `none` 转换函数指定要忽略的字段。

```
... | convert auto(*) none(foo)
```

3. 将指定字段的持续时间值更改为秒

将指定字段的持续时间值更改为秒

```
... | convert dur2sec(xdelay) dur2sec(delay)
```

4. 将 `sendmail syslog` 持续时间格式更改为秒

将 `sendmail syslog` 持续时间格式（D+HH:MM:SS）更改为秒。例如，如果 `delay="00:10:15"`，结果值是 `delay="615"`。本示例使用 `dur2sec` 转换函数。

```
... | convert dur2sec(delay)
```

5. 转换包含数字和字符串值的字段值

通过移除字符串值将 `duration` 字段中的值（其中包含数字和字符串值）转换为数字值。例如，如果 `duration="212 sec"`，结果值是 `duration="212"`。本示例使用 `rmunit` 转换函数。

```
... | convert rmunit(duration)
```

6. 将内存值更改为以 KB 为单位

将 `virt` 字段中的所有内存值更改为以 KB 为单位。本示例使用 `memk` 转换函数。

```
... | convert memk(virt)
```

延伸示例

1. 将 UNIX 时间转换成一个更方便阅读的时间格式

将 UNIX 时间转换为显示小时、分钟和秒数的更方便阅读的时间格式。

```
source="all_month.csv" | convert timeformat="%H:%M:%S" ctime(_time) AS c_time | table _time, c_time
```

- `ctime()` 函数将 CSV 文件事件中的 `_time` 值转换为 `timeformat` 参数指定的格式。
- `timeformat="%H:%M:%S"` 参数指示搜索将 `_time` 值的格式设置为 `HH:MM:SS`。
- 将转换的时间 `ctime` 字段重新命名为 `c_time`。
- `table` 命令用于显示原始 `_time` 值和 `ctime` 字段。

统计选项卡中显示的结果如下所示：

<code>_time</code>	<code>c_time</code>
2018-03-27 17:20:14.839	17:20:14
2018-03-27 17:21:05.724	17:21:05
2018-03-27 17:27:03.790	17:27:03
2018-03-27 17:28:41.869	17:28:41
2018-03-27 17:34:40.900	17:34:40
2018-03-27 17:38:47.120	17:38:47
2018-03-27 17:40:10.345	17:40:10
2018-03-27 17:41:55.548	17:41:55

`ctime()` 函数将时间戳更改为非数字值。这对于在报表中显示或增强事件列表中的可读性是非常有用的。

2. 将 `MM:SS.SSS` 形式的时间转换为以秒的形式表示的数字

将 `MM:SS.SSS`（分钟、秒和次秒）形式的时间转换为以秒的形式表示的数字。

```
sourcetype=syslog | convert mstime(_time) AS ms_time | table _time, ms_time
```

- `mstime()` 函数将 `_time` 字段值从分钟和秒钟共同表示转换为只用秒钟来表示。

将转换的时间字段重新命名为 `ms_time`。

- `table` 命令用于显示原始 `_time` 值和转换的时间。

<code>_time</code>	<code>ms_time</code>
2018-03-27 17:20:14.839	1522196414.839
2018-03-27 17:21:05.724	1522196465.724
2018-03-27 17:27:03.790	1522196823.790
2018-03-27 17:28:41.869	1522196921.869
2018-03-27 17:34:40.900	1522197280.900
2018-03-27 17:38:47.120	1522197527.120
2018-03-27 17:40:10.345	1522197610.345
2018-03-27 17:41:55.548	1522197715.548

`mstime()` 函数会将时间戳更改为数字值。如果您希望使用该时间来进行更多的计算，这会非常有用。

3. 将 `HH:MM:SS` 格式的字符串时间转换成数字

将包含 `HH:MM:SS` 格式的时间的字符串字段 `time_elapsed` 转换为数字。按 `user_id` 字段总结 `time_elapsed`。本示例使用 `eval` 命令将已转换的结果从秒转换成分。

```
... | convert num(time_elapsed) | stats sum(eval(time_elapsed/60)) AS Minutes BY user_id
```

另请参阅

命令
Eval
fieldformat
函数
tonumber
strptime

correlate

描述

计算不同字段之间的相关性。

您可以使用 `correlate` 命令查看数据中字段之间的同现概览。结果将以矩阵形式显示，其中两个字段的交叉表是一个单元值。该单元值表示两个字段同时存在于相同事件中的次数的百分比。

结果特定的字段以 `RowField` 字段值命名，而与其相比的字段是其他字段的名称。

注意：此命令会查看搜索结果集中所有字段之间的关系。如果您要分析字段值之间的关系，请参阅 `contingency` 命令，该命令用于统计事件中字段值对的同现数量。

语法

`correlate`

限制

`correlate` 在搜索中考虑的字段数量是有限制的。通过 `limits.conf` 中 `[correlate]` 段落的 `maxfields` 设置此上限。默认值为 1000。

若字段数量超过此默认值，`correlate` 命令将继续处理前 N（例如 1,000）个字段名称的数据，但忽略其他字段的数据。如果发生这种情况，来自搜索或告警的通知将包含一条消息“关联：达到输入字段限制 (N)。可能忽略一些字段。”

对于所有设计好的限制，调整此限制可能会产生明显的内存或 CPU 成本。

示例

示例 1:

查看 `_internal` 索引内所有字段之间的同现相关性。

```
index=_internal | correlate
```

下面是相关结果的快照。

	<code>RowField</code>	<code>abandoned_channels</code>	<code>actions_triggered</code>	<code>active_hist_searches</code>	<code>active_realtime_searches</code>	<code>average_kbps</code>	<code>avg_age</code>
1	<code>abandoned_channels</code>	1.00	0.00	0.00	0.00	0.00	0.00
2	<code>actions_triggered</code>	0.00	1.00	0.00	0.00	0.00	0.00
3	<code>active_hist_searches</code>	0.00	0.00	1.00	1.00	0.00	0.00
4	<code>active_realtime_searches</code>	0.00	0.00	1.00	1.00	0.00	0.00
5	<code>average_kbps</code>	0.00	0.00	0.00	0.00	1.00	0.00
6	<code>avg_age</code>	0.00	0.00	0.00	0.00	0.00	1.00
7	<code>browser</code>	0.00	0.00	0.00	0.00	0.00	0.00
8	<code>bytes</code>	0.00	0.00	0.00	0.00	0.00	0.00

由于 `_internal` 中存在不同类型的日志，您可以预料到不会同时出现很多字段。

示例 2:

计算 Web 访问事件中所有字段之间的同现相关性。

```
sourcetype=access_* | correlate
```

您预计所有 Web 访问事件共享相同的字段：`clientip`、`referer`、`method` 等。但是，由于 `sourcetype=access_*` 包括 `access_common` 和 `access_combined` 两种 Apache 日志格式，因此您应该会发现部分字段百分比小于 1.0。

示例 3:

计算下载事件中所有字段之间的同现相关性。

```
eventtype=download | correlate
```

在您将结果传递至 `correlate` 之前，搜索范围越小，所有字段值对具有 1.0 相关性的可能性就越大。1.0 相关性表示这些值同时出现在所有搜索结果中。对于这些下载事件，您可以根据哪个字段值对的同现相关性小于 1.0 来发现问题。

另请参阅

`associate`, `contingency`

ctable

`ctable` 或 `counttable` 命令是 `contingency` 命令的别名。请参阅 `contingency` 命令了解语法和示例相关信息。

datamodel

描述

检查和搜索数据模型数据集。

使用 `datamodel` 命令为所有或指定的模型及其数据集返回 JSON。您还可以在该数据模型中搜索指定的数据模型或数据集。

数据模型是有关一个或多个数据集语义知识的分层结构搜索时间映射。数据模型将编码构建这些数据集的各种专门搜索所需的域知识。有关更多信息，请参阅《知识管理器手册》中的“关于数据模型”和“设计数据模型”。

`datamodel` 搜索命令允许您从搜索界面中搜索现有的数据模型及其数据集。

`datamodel` 命令属于生成命令，应该是搜索中的第一个命令。生成命令使用前导管道字符。

语法

```
| datamodel [<data model name>] [<dataset name>] [<data model search mode>] [strict_fields=<bool>]  
[allow_old_summaries=<bool>] [summariesonly=<bool>]
```

必要参数

无

可选参数

数据模型名称

语法: <string>

描述: 要搜索数据模型的名称。当仅指定数据模型时，此搜索将为单个数据模型返回 JSON。

数据集名称

语法: <string>

描述: 要搜索数据模型数据集的名称。必须在数据模型名称后指定。此搜索将为单个数据集返回 JSON。

数据模型搜索模式

语法: <data model search result mode> | <data model search string mode>

描述: 您可以使用 `datamodel` 对数据模型或数据模型数据集运行返回结果或搜索字符串的搜索。如果您想这么做，您必须提供 `<data model search mode>`。有两种 `<data model search mode>` 子分类：返回结果的模式和返回搜索字符串的模式。请参阅 `<data model search mode>` 选项。

allow_old_summaries

语法: allow_old_summaries=<bool>

描述: 此参数仅适用于加速的数据模型。当您更改定义数据模型的约束，但是 Splunk 软件没有完全更新摘要以反映该更改时，摘要可能包含一些与旧定义匹配的数据和一些与新定义匹配的数据。默认情况下 `allow_old_summaries = false`，这意味着搜索头不会使用比新的摘要定义更早的摘要目录。这确保 `datamodel` 搜索结果将始终反映您的当前配置。如果设置 `allow_old_summaries = true`, `datamodel` 将使用当前摘要数据和定义更改之前生成的摘要数据。如果您认为旧的摘要数据与结果可靠的新的摘要数据非常相近，那么您可以在搜索中设置 `allow_old_summaries=true`。

默认值: `false`

summariesonly

语法: summariesonly=<bool>

描述: 此参数仅适用于加速的数据模型。如果设为 false, datamodel 搜索会返回所选数据模型的摘要和非摘要数据。如果设为 true, 搜索只针对所选数据模型返回以 TSIDX 格式汇总的数据结果。您可以使用这个参数来识别当前为指定的数据模型汇总了哪些数据, 或者确保特定的数据模型搜索能够有效地运行。

默认值: false

strict_fields

语法: strict_fields=<bool>

描述: 根据返回的字段确定 datamodel 搜索的范围。如果 strict_fields=true, 搜索只返回默认字段和指定的数据模型数据集的约束中包含的字段。如果 strict_fields=false, 搜索会返回数据模型中定义的所有字段, 包括从父级数据模型数据集继承的字段、提取字段、计算字段和从查找衍生的字段。

您还可以为特定数据模型将 strict_fields 设为默认值 false。请参阅《知识管理器手册》中的“设计数据模型”。

默认值: true

<data model search mode> 选项

数据模型搜索结果模式

语法: search | flat | acceleration_search

描述: 在返回结果的数据模型或数据模型数据集上运行搜索的模式。

模式	描述
search	严格按照定义形式返回搜索结果。
flat	返回与 search 相同的结果, 但会从字段名称中删除层次信息。例如, search 模式可能返回名为 dmdataset.server 的字段, flat 模式则会返回名为 server 的字段。
acceleration_search	运行搜索头用于加速数据模型的搜索。此模式仅在仅使用流命令的根事件数据集和根搜索数据集上有效。

数据模型搜索字符串模式

语法: search_string | flat_string | acceleration_search_string

描述: 当数据模型通过相应的 <data model search result mode> 运行 SPL 时, 这些模式返回 Splunk 软件针对数据模型实际运行的搜索的字符串。例如, 如果您选择 acceleration_search_string, 当您通过 acceleration_search 模式运行 SPL 时, Splunk 软件会返回针对数据模型实际使用的搜索字符串。

用法

datamodel 命令属于报表生成命令。请参阅“命令类型”。

生成命令使用前导管道符且应是搜索中的第一个命令。

示例

1. 为所有数据模型返回 JSON

为当前应用上下文可用的所有数据模型返回 JSON。

```
| datamodel
```

i	时间	事件
>	{ [-]	<pre>description: Splunk's Internal Audit Logs record user activity, including searches and configuration changes. displayName: Splunk's Internal Audit Logs - SAMPLE modelName: internal_audit_logs objectNameList: [[+]] objectSummary: { [+] } objects: [[+]] }</pre> <p>显示为原始文本</p>
>	{ [-]	<pre>description: Splunk's Internal Server Logs record information about system usage and performance. displayName: Splunk's Internal Server Logs - SAMPLE modelName: internal_server objectNameList: [[+]] objectSummary: { [+] } objects: [[+]] }</pre> <p>显示为原始文本</p>

2. 为特定数据模型返回 JSON

为具有模型 ID `internal_audit_logs` 的 Splunk 内部审计日志 – 示例数据模型返回 JSON。

| datamodel internal_audit_logs

i	时间	事件
>	{ [-]	<pre>description: Splunk's Internal Audit Logs record user activity, including searches and configuration changes. displayName: Splunk's Internal Audit Logs - SAMPLE modelName: internal_audit_logs objectNameList: [[+]] objectSummary: { [+] } objects: [[+]] }</pre> <p>显示为原始文本</p>

3. 为特定数据集返回 JSON

为 Buttercup Games 的 `Client_errors` 数据集返回 JSON。

| datamodel Tutorial Client_errors

4. 在特定数据集上运行搜索

为 Buttercup Games 的 `Client_errors` 返回 JSON。

| datamodel Tutorial Client_errors search

5. 按特定标准在数据集上运行搜索

针对 404 错误搜索 Buttercup Games 的 `Client_errors` 数据集并计算事件数量。

| datamodel Tutorial Client_errors search | search Tutorial.status=404 | stats count

6. 对于加速的数据模型，显示在选定的时间范围内汇总了哪些数据

教程数据模型加速后，此搜索将 `summariesonly` 参数和 `timechart` 结合使用，以显示在选定的时间范围内为 `Client_errors` 数据集汇总了哪些数据。

```
| datamodel Tutorial summariesonly=true search | timechart span=1h count
```

另请参阅

`pivot`

datamodel simple

`datamodel simple` 命令与 Splunk 常用信息模型加载项结合使用。

如需此命令的相关信息，请参阅《常用信息模型加载项手册》中的“使用 `datamodel simple` 命令”。

dbinspect

描述

在指定索引中返回有关数据桶的信息。如果使用 Splunk Enterprise，此命令可帮助您了解数据驻留的位置，以便您可以根据需要优化磁盘使用情况。索引器群集上的搜索只会从主要数据桶返回结果。

Splunk 索引是 Splunk 软件获取的数据的存储库。由于已为传入数据新建索引且已将传入数据转换为事件，Splunk 软件会新建原始数据和元数据文件（索引文件）。这些文件驻留在按时间组织的目录集中。这些目录称为数据桶。

有关更多信息，请参阅管理索引器和索引器群集中的“索引、索引器和群集”以及“索引器如何存储索引”。

语法

要求的语法以粗体表示。

```
dbinspect  
[index=<wc-string>]...  
<span> | <timeformat>  
[corruptonly=<bool>]  
[cached=<bool>]
```

必要参数

无。

可选参数

`index`

语法： `index=<wc-string>...`
描述： 指定要检查的索引的名称。您可以指定多个索引。对于所有非内部索引，您可以在索引名称中指定星号（*）。
默认值： 默认索引，通常是主索引。

``

语法： `span=<int> | span=<int><timescale>`
描述： 指定数据桶的跨度长度。如果使用 `timescale` 单位（秒、分、小时、天、月或次秒），则将其用作时间范围。否则，这是一个绝对的数据桶“长度”。

使用数据桶跨度调用 `dbinspect` 命令时，将返回一份表格，其中包含每个数据桶的跨度。若未指定 `span`，则返回索引中数据桶的相关信息。未指定数据桶跨度时请参阅“返回的信息”。

`<timeformat>`

语法： `timeformat=<string>`
描述： 设置 `modTime` 字段的时间格式。
默认值： `timeformat=%m/%d/%Y:%H:%M:%S`

`<corruptonly>`

语法： `corruptonly=<bool>`
描述： 指定是否检查每个数据桶以确定是否损坏了任何数据桶，并仅显示损坏的数据桶。若数据桶中的某些文件不正确或丢失（如 `Hosts.data` 或 `tsidx`），则数据桶损坏。数据桶损坏可能返回不正确的数据或提交不可搜索的数据桶。大多数情况下，软件会自动修复损坏的数据桶。

如果 `corruptonly=true`, 将检查每个数据桶并显示以下指示消息。

INFO: The "corruptonly" option will check each of the specified buckets. This search might be slow and will take time.

默认值: false

`cached`

语法: `cached=<bool>`

描述: 如果设置为 `cached=true`, 则 `dbinspect` 命令会从数据桶清单中获取统计信息。如果设置为 `cached=false`, 则 `dbinspect` 命令会检查数据桶本身。对于 `SmartStore` 数据桶, `cached=false` 会检查索引器的数据桶本地副本。但是, 指定 `cached=true` 后系统会代替检查数据桶清单, 其中包含有关驻留在远程存储中的数据桶规范版本的信息。有关更多信息, 请参阅《管理索引器和索引器群集》中的“故障排除 SmartStore”。

默认值: 对于非 `SmartStore` 索引, 默认值为 `false`。对于 `SmartStore` 索引, 默认值为 `true`。

时间刻度单位

用于将 `timescale` 指定为数据桶跨度的选项。

`<timescale>`

语法: `<sec> | <min> | <hr> | <day> | <month> | <subseconds>`

描述: 时间刻度单位。

时间刻度	语法	描述
<code><sec></code>	<code>s sec secs second seconds</code>	时间刻度 (秒)。
<code><min></code>	<code>m min mins minute minutes</code>	时间刻度 (分钟)。
<code><hr></code>	<code>h hr hrs hour hours</code>	时间刻度 (小时)。
<code><day></code>	<code>d day days</code>	时间刻度 (天)。
<code><month></code>	<code>mon month months</code>	时间刻度 (月)。
<code><subseconds></code>	<code>us ms cs ds</code>	单位为微秒 (us)、毫秒 (ms)、百分之一秒 (cs) 或十分之一秒 (ds) 的时间刻度。

未指定跨度时返回的信息

若调用 `dbinspect` 命令但未使用 `span` 参数, 将返回索引中有关数据桶的如下信息。

字段名称	描述
<code>bucketId</code>	字符串由 <code><index>~<id>~<guid></code> 组成, 其中分隔符为波形符。例如, <code>summary~2~4491025B~8E6D~48DA~A90E~89AC3CF2CE80</code> 。
<code>endEpoch</code>	数据桶中上个事件的时间戳, 最大限度接近未来数据桶的时间边缘。指定以 UNIX Epoch 的秒数表示的时间戳。
<code>eventCount</code>	数据桶中的事件数。
<code>guid</code>	托管索引的服务器的全局唯一标识符 (GUID)。这与索引复制相关。
<code>hostCount</code>	数据桶中唯一主机的数量。
<code>id</code>	数据桶的 ID, 该 ID 在生成数据桶的索引器上生成。
<code>index</code>	在您的搜索中指定的索引名称。若要检查所有索引, 您可以指定 <code>index=*</code> , 索引字段会随之而异。
<code>modTime</code>	上一次修改或更新数据桶的时间戳以 <code>timeformat</code> 标记指定的格式表示。
<code>path</code>	数据桶的位置。数据桶 <code>path</code> 的命名约定略有不同, 具体取决于该数据桶在其索引器作为群集对等节点运行时是否滚动到温数据桶: <ul style="list-style-type: none">• 对于非群集数据桶: <code>db_<newest_time>_<oldest_time>_<localid></code>• 对于群集原始数据桶副本: <code>db_<newest_time>_<oldest_time>_<localid>_<guid></code>• 对于群集复制的数据桶副本: <code>rb_<newest_time>_<oldest_time>_<localid>_<guid></code> 有关更多信息, 请参阅《管理索引器和索引器群集》中的“Splunk 如何存储索引”以及“基本群集架构”。
<code>rawSize</code>	每个数据桶中原始数据文件的数据量 (字节)。本值代表压缩前和添加索引文件的数据量。
<code>sizeOnDiskMB</code>	数据桶所占的磁盘空间大小 (MB) 以浮点数表示。本值代表压缩的原始数据文件和索引文件的数据量。
<code>sourceCount</code>	数据桶中唯一来源的数量。

sourceTypeCount	数据桶中唯一来源类型的数量。
splunk_server	在分布式环境中托管索引的 Splunk 服务器名称。
startEpoch	数据桶中首个事件的时间戳（最大限度接近过去数据桶的时间边缘）以 UNIX epoch 的秒数表示。
state	数据桶是温数据桶、热数据桶还是冷数据桶。
corruptReason	说明数据桶损坏的原因。仅当 corruptonly=true 时出现 corruptReason 字段。

用法

`dbinspect` 命令属于生成命令，应该是搜索中的第一个命令。生成命令使用前导管道字符。

访问数据和安全

如果没有数据从您使用 `dbinspect` 命令指定的索引中返回，您可能没有访问该索引的授权。在 Splunk 索引中访问数据的能力受制于赋予每个角色的授权。请参阅《确保 Splunk Enterprise 安全》中的“使用访问权限控制以确保 Splunk 数据安全”。

非可搜索的数据桶副本

对于目标对等节点上的非可搜索的热数据桶副本，不保留 `tsidx` 和其他元数据文件。因为无法报告精确信息，以下字段将显示 NULL：

- eventCount
- hostCount
- sourceCount
- sourceTypeCount
- startEpoch
- endEpoch

示例

1. `dbinspect` 命令的 CLI 使用

使用命令行界面 (CLI) 显示跨度为 1 天的图表。

```
myLaptop $ splunk search "| dbinspect index=_internal span=1d"
```

_time	hot-3	warm-1	warm-2
2015-01-17 00:00:00.000 PST	0		
2015-01-17 14:56:39.000 PST	0		
2015-02-19 00:00:00.000 PST	0	1	
2015-02-20 00:00:00.000 PST	2		1

2. 默认的 `dbinspect` 输出

本地 `_internal` 紴引的默认 `dbinspect` 输出。

```
| dbinspect index=_internal
```

列表	格式	每页 20 个									
bucketId	endEpoch	eventCount	guid	hostCount	id	index	modTime	path	rawSize	sizeOnDiskMB	sourceCount
_internal~0~A11D93E6-	1381275242	555490	A11D93E6-C13C-4192-A832-9C4B3175273F	1	0	_internal	10/08/2013:16:34:20	/Applications/splunk/var/lib/splunk/_internaldb/_db/_1381275242_1379996500_0	102033831	44.742188	13
_internal~1~A11D93E6-	1381275250	6310	A11D93E6-C13C-4192-A832-9C4B3175273F	1	1	_internal	10/08/2013:16:58:07	/Applications/splunk/var/lib/splunk/_internaldb/_db/_1381275250_1381242842_1	1315790	0.847656	10
_internal~2~A11D93E6-	1381246070	7704	A11D93E6-C13C-4192-A832-9C4B3175273F	1	2	_internal	10/08/2013:17:28:07	/Applications/splunk/var/lib/splunk/_internaldb/_db/_1381246070_1381244270_2	1497111	0.964844	10
_internal~3~A11D93E6-	1381753579	293107	A11D93E6-C13C-4192-A832-9C4B3175273F	1	3	_internal	10/14/2013:15:31:57	/Applications/splunk/var/lib/splunk/_internaldb/_db/_1381753579_1381245988_3	50589038	20.003906	12
_internal~4~A11D93E6-	1381940762	171863	A11D93E6-C13C-4192-A832-9C4B3175273F	1	4	_internal	10/16/2013:18:26:21	/Applications/splunk/var/lib/splunk/_internaldb/_db/_1381940762_1381753580_4	30590658	13.128906	11
_internal~5~A11D93E6-	1381941015	1572	A11D93E6-C13C-4192-A832-9C4B3175273F	1	5	_internal	10/16/2013:18:31:15	/Applications/splunk/var/lib/splunk/_internaldb/_db/_1381941015_1381940762_5	309869	0.312500	10
_internal~6~A11D93E6-	1381941741	1967	A11D93E6-C13C-4192-A832-9C4B3175273F	1	6	_internal	10/16/2013:18:42:33	/Applications/splunk/var/lib/splunk/_internaldb/_db/_1381941741_1381941016_6	319026	0.281250	10

此图未显示输出表中的所有列。在您的电脑上，向右滚动屏幕以查看其他列。

3. 确认损坏的数据桶

使用 `corruptonly` 参数显示有关损坏的数据桶的信息，而非所有数据桶的信息。无论有没有 `corruptonly` 参数，显示的输出字段都一样。

```
| dbinspect index=_internal corruptonly=true
```

4. 统计每个 Splunk 服务器的数据桶数量

使用该命令来验证分布式环境中的 Splunk 服务器是否包含于 `dbinspect` 命令中。统计每个服务器的数据桶数量。

```
| dbinspect index=_internal | stats count by splunk_server
```

5. 查找数据桶的索引大小，单位为 GB

使用 `dbinspect` 查找数据桶的索引大小，单位为 GB。对于当前数字，针对近期的时间范围运行此搜索。

```
| dbinspect index=_internal | eval GB=sizeOnDiskMB/1024| stats sum(GB)
```

dbxquery

`Dbxquery` 命令和 `Splunk DB Connect` 结合使用。

关于此命令的信息，请参阅《部署和使用 `Splunk DB Connect`》中的“使用 `dbxquery` 命令执行 SQL 语句和存储过程”。

dedup

描述

删除包含有与指定字段值相同组合的事件。

您可以使用 `dedup` 命令指定要为单个字段中的每个值或多个字段中的每个值组合保留的重复事件数量。`dedup` 按搜索顺序返回事件。对于 **历史搜索**，会先搜索最近的事件。对于 **实时搜索**，会先搜索首先收到的事件，并不一定是最近的事件。

可指定要保留的具有重复值或值组合的事件数量。您可以对字段进行排序，这确定了保留的事件。其他选项将允许您保留事件但会删除重复字段，或保留未包含指定字段的事件。

语法

要求的语法以**粗体**表示。

```
dedup
[<int>]
<field-list>
[keepevents=<bool>]
[keepempty=<bool>]
```

```
[consecutive=<bool>]  
[sortby <sort-by-clause>]
```

必要参数

```
<field-list>  
语法: <string> <string> ...  
描述: 要从中删除重复值的字段名称列表。
```

可选参数

```
consecutive  
语法: consecutive=<bool>  
描述: 如果为 true, 仅删除具有连续值重复组合的事件。  
默认值: false
```

```
keepempty  
语法: keepempty=<bool>  
描述: 如果为 true, 则会保留未包含一个或多个指定字段 (为空) 的事件。  
默认值: false。丢弃任何所选字段为空的所有事件。
```

keepempty=true 参数保留未包含字段列表中的一个或多个字段的所有事件。要保留 N 代表的包含空值的字段值组合的事件, 使用 fillnull 命令来为这些字段提供非空值。例如:

```
... | fillnull value="MISSING" field1 field2 | dedup field1 field2
```

```
keepevents  
语法: keepevents=<bool>  
描述: 如果为 true, 保留所有事件, 但将从第一个包含特定值组合的事件之后的事件中删除所选字段。  
默认值: false。丢弃每个特定组合的首个事件之后的事件。
```

```
<N>  
语法: <int>  
描述: dedup 命令将在您指定 N 后为每个组合保留多个事件。N 的值必须大于 0。如果未指定它的值, 则只保留第一个出现的事件。并会删除所有其他重复事件。
```

```
<sort-by-clause>  
语法: sortby ( - | + ) <sort-field> [(- | +) <sort_field> ...]  
描述: 列出排序所依据的字段及其排序顺序。用破折号 ( - ) 表示降序, 加号 ( + ) 表示升序。您必须为 <sort-by-clause> 中指定的每个字段指定排序顺序。<sort-by-clause> 确定要保留哪些复制事件。事件列表进行排序时, 保留排序列表的重复事件中排在最前面的事件。
```

排序字段选项

```
<sort-field>  
语法: <field> | auto(<field>) | str(<field>) | ip(<field>) | num(<field>)  
描述: 可指定用于排序事件的选项。
```

```
<field>  
语法: <string>  
描述: 要排序的字段的名称。
```

```
auto  
语法: auto(<field>)  
描述: 自动确定如何排序字段的值。
```

```
ip  
语法: ip(<field>)  
描述: 将字段的值解释为 IP 地址。
```

```
num  
语法: num(<field>)  
描述: 将字段的值解释为数字。
```

```
str  
语法: str(<field>)  
描述: 按字典顺序排列字段值。
```

用法

`dedup` 命令是一个流命令或数据集处理命令，具体取决于使用命令指定的参数类型。例如，如果指定 `<sort-by-clause>`，则 `dedup` 命令即为数据集处理命令。排序之前必须收集所有结果。请参阅“命令类型”。

若您要对大量数据执行搜索，不要把 `dedup` 命令用于 `_raw` 字段。若您搜索 `_raw` 字段，则会保留内存中每个事件的文本，从而影响搜索性能。这属于预期行为。此行为适用于基数很高和较大的任何字段。

多值字段

要在多值字段上使用 `dedup` 命令，字段必须匹配所有要重复数据删除的值。

按字典顺序

基于用于编码计算机内存中的项目的值按字典顺序对这些项目进行排序。在 Splunk 软件中，几乎都是使用 UTF-8 进行编码，这是 ASCII 的超集。

- 数字排在字母前面。根据第一位数字对数字进行排序。例如数字 10、9、70、100，按照字典顺序排序为 10、100、70、9。
- 大写字母排在小写字母前面。
- 符号的排序标准不固定。有些符号排在数字值前面。有些符号排在字母前面或后面。

示例

1. 删除一个字段的重复结果

删除带相同 `host` 值的重复搜索结果。

```
... | dedup host
```

2. 删除重复的结果并按升序对结果进行排序

删除带相同 `source` 值的重复搜索结果，并按 `_time` 字段以升序对事件进行排序。

```
... | dedup source sortby +_time
```

3. 删除重复的结果并按降序对结果进行排序

删除带相同 `source` 值的重复搜索结果，并按 `_size` 字段以降序对事件进行排序。

```
... | dedup source sortby -_size
```

4. 保留前 3 个重复的结果

对于带相同 `source` 值的搜索结果，保留前 3 个重复的结果，并删除后续所有重复的结果。

```
... | dedup 3 source
```

5. 保留在多个字段中具有相同值组合的结果

对于带相同 `source` 和 `host` 值的搜索结果，保留前 2 个重复的结果，并删除后续所有重复的结果。

```
... | dedup 2 source host
```

6. 仅删除连续的重复事件

仅删除连续的重复事件。保留非连续的重复事件。在此示例中，带相同 `source` 和 `host` 字段值组合的结果才算重复的结果。

```
... | dedup consecutive=true source host
```

另请参阅

`uniq`

`delete`

描述

使用 `delete` 命令把搜索返回的所有事件标记为已删除。后续搜索将不再返回已标记的事件。删除后，没有用户可以查看此数据，包括具有管理员权限的用户。`delete` 命令不会回收磁盘空间。

删除数据是不可撤消的操作。如果想要重新获取已删除的数据，则必须对适用的数据源重新建立索引。

您无法在事件到达时在实时搜索中运行 `delete` 命令以删除事件。

语法

`delete`

用法

仅具有 `"delete_by_keyword"` 权限的用户才可以访问 `delete` 命令。默认情况下，只有 `"can_delete"` 角色可以删除事件。其他角色没有这个功能，包括管理员角色。建议您新建一个特殊的 `userid`，您可以在想要删除索引数据时使用此用户登录。

若要使用 `delete` 命令，请运行一项返回欲删除事件的搜索。请确保此搜索“仅”返回您想要删除的事件，而不会返回其他事件。确认搜索结果中包含您想要删除的数据后，通过管道符把搜索结果传递给 `delete` 命令。

在受影响的索引中，`delete` 命令不会触发从热数据桶滚动到温数据桶。

`delete` 命令的输出结果为一份表格，内含通过字段 `splunk_server`（索引器或搜索头的名称）删除的事件的数量、索引，以及通过索引 `"__ALL__"` 获取的每个服务器的汇总记录。删除事件的数量位于 `deleted` 字段中。也会发出 `errors` 字段，其值通常为 0。

Delete 命令限制

`delete` 命令不适用于所有情况：

使用集中流命令进行搜索。

在集中流命令之后不能使用 `delete` 命令。例如，您不能使用如下搜索来删除事件：

```
index=myindex ... | head 100 | delete
```

集中流命令包括：`head`、`streamstats`、`dedup` 的一些模式以及群集的一些模式。请参阅“命令类型”。

包含 `index` 字段的事件。

如果除了应用于所有事件的默认 `index` 字段之外，您的事件还包含名为 `index` 的字段。若事件中确实额外包含 `index` 字段，如本示例所示，您可以先使用 `eval` 再调用 `delete`：

```
index=fbus_summary latest=1417356000 earliest=1417273200 | eval index = "fbus_summary" | delete
```

将数据从索引中永久地移除

`delete` 命令不能从您的磁盘空间中移除数据。您必须使用 CLI 中的 `clean` 命令以永久地将数据移除。`clean` 命令移除索引中的所有数据。您无法选择您想要移除的特定数据。请参阅《管理索引器和索引器群集》中的“移除索引和索引数据”。

示例

1. 删除有社会保险号的事件

从 `insecure` 索引中删除包含了看起来像社会保险号的字符串的事件。使用 `regex` 命令识别包含您想要匹配的字符串的事件。

1. 运行以下搜索确保您从 `insecure` 索引中检索正确的数据。

```
index=insecure | regex _raw = "\d{3}-\d{2}-\d{4}"
```

2. 如有必要，调整搜索以检索正确的数据。然后添加 `delete` 命令到搜索末尾以删除事件。

```
index=insecure | regex _raw = "\d{3}-\d{2}-\d{4}" | delete
```

2. 删除包含特定单词的事件

从 `imap` 索引中删除包含 `invalid` 一词的事件。

```
index=imap invalid | delete
```

3. 删除搜索教程事件

删除索引中所有 Splunk 搜索教程事件。

1. 使用 admin 角色以用户身份登录。
2. 单击设置、访问控制并使用 can_delete 角色新建一个用户。
3. 以管理员身份注销并以具有 can_delete 角色的用户身份重新登录。
4. 将时间范围挑选器设为 All time。
5. 运行下列搜索以检索所有搜索教程事件。

```
source=tutorialdata.zip:*
```

6. 确认搜索正在检索正确的数据。
7. 添加 delete 命令到搜索条件的结尾，并再次运行搜索。

```
source=tutorialdata.zip:* | delete
```

事件从索引中移除。

8. 以具有 can_delete 角色的用户身份注销。

delta

描述

使用特定数字字段的值来计算邻近结果之差。对于 <field> 为数字的所有事件，delta 命令将按搜索顺序计算当前事件 <field> 值与前一个事件 <field> 值之间的差值。delta 命令将把差值写入 <newfield>。

语法

要求的语法以粗体表示。

```
delta
<field> [AS <newfield>]
[p=int]
```

必要参数

field

语法: <field-name>
描述: 要分析的字段的名称。如果 <field> 不是数字字段，则不会生成任何输出字段。

可选参数

newfield

语法: <string>
描述: 要将输出写入其中的新字段的名称。
默认值: delta(<field>)

p

语法: p=<int>
描述: 指定当前结果之前有多少结果用于比较当前结果中的 field 值。前面的结果由搜索顺序而定，而并非一定按事件顺序排列。如果 p=1，将当前结果值和当前结果之前的第一个结果中的值相比较。如果 p=2，将当前结果值和当前结果之前的第二个结果中的值相比较，以此类推。
默认值: 1

用法

delta 命令按照搜索返回事件的顺序处理事件。默认情况下，历史搜索的事件按颠倒的时间顺序排列，即从新事件到旧事件，因此按时间升序排列的值将显示负增量。

对于实时搜索，事件将按接收到的次序进行比较。

delta 可以应用于任何命令序列之后，所以不保证输入顺序。例如，若您运用独立字段对结果进行排序，然后再使用 delta 命令，生成的值将是按特定顺序排列的增量值。

基本示例

1. 计算活动中的差异

通过有线电视提供商的日志 sourcetype=tv，您可以分析节目收视率、客户偏好等。订户最常观看哪个频道，activity=view，以及订户在那些频道上停留多长时间？

```
sourcetype=tv activity="View" | sort - _time | delta _time AS timeDeltaS | eval timeDeltaS=abs(timeDeltaS) | stats sum(timeDeltaS) by ChannelName
```

2. 计算当前值和前面第 3 个值之间的差异

计算计数当前值和该计数前面第三个值之间的差异，将结果存储在默认字段 delta（字段名称）中，在此示例中就是 delta(count)。

```
... | delta count p=3
```

3. 计算当前值和前一个值的差异并重命名结果字段

对于具有 'count' 的每个事件，计算计数与前一个值之差，并将结果保存在字段 countdiff 中。

```
... | delta count AS countdiff
```

延伸示例

1. 计算前 10 个买家之间购买数量的差异

此示例使用搜索教程中的示例数据，但应该可以与任意格式的 Apache Web 访问日志配合使用。要在您自己的 Splunk 实例中尝试此示例，您必须下载示例数据并遵循操作以将教程数据导入 Splunk。当运行搜索时，请使用时间范围昨天。

查找昨天买得最多的前十位顾客，统计他们购买了多少商品并计算每位顾客购买数量的差异。

```
sourcetype=access_* status=200 action=purchase | top clientip | delta count p=1
```

- 购买事件 action=purchase 被传递至 top 命令，以根据 clientip 查找购买量最大的前十大用户。
- 然后，这些结果（其中每个 count 对应一个 clientip）将通过管道符传递给 delta 命令，以使用 p=1 参数计算某一事件的 count 值与上一个事件的 count 值之间的值差。
- 默认情况下，此差异保存在名为 delta(count) 的新字段中。
- 第一个事件没有 delta(count) 值。

结果形式如下所示：

clientip	count	百分比	delta(count)
87.194.216.51	134	2.565084	
128.241.220.82	95	1.818530	-39
211.166.11.101	91	1.741960	-4
107.3.146.207	72	1.378254	-19
194.215.205.19	60	1.148545	-12
109.169.32.135	60	1.148545	0
188.138.40.166	56	1.071975	-4
74.53.23.135	49	0.937979	-7
187.231.45.62	48	0.918836	-1
91.208.184.24	46	0.880551	-2

2. 计算最近事件的时间差异

此示例使用从 USGS 地震网站下载的近期地震数据。该数据是一个逗号分隔的 ASCII 文本文件，其中包含每次记录的地震的震级（mag）、坐标（经度、纬度）、区域（地点）等。

您可以从 [USGS 地震源](#) 下载当前 CSV 文件并作为输入添加。

计算阿拉斯加最近每次地震的时间差。使用时间范围所有时间运行搜索。

```
source=all_month.csv place=*alaska* | delta _time p=1 | rename delta(_time) AS timeDeltaS | eval timeDeltaS=abs(timeDeltaS) | eval "Time Between Quakes"=tostring(timeDeltaS,"duration") | table place, _time, "Time Between Quakes"
```

- 此示例搜索阿拉斯加的地震。

`delta` 命令用于计算每次地震和其前一次地震的时间戳 `_time` 之差。默认情况下，差异放在名为 `delta(_time)` 的新字段。时间以秒为单位。

- `rename` 命令用于将默认字段名称更改为 `timeDeltaS`。
- `eval` 命令与 `abs` 函数一起使用以将时间转换为时间的绝对值。此转换是必要的，因为一次地震和其前一次地震的时间差为负值。
- 另一个 `eval` 命令与 `tostring` 函数一起使用以将时间（以秒为单位）转换为字符串值。`duration` 参数是指定将值转换成可读时间格式 HH:MM:SS 的 `tostring` 函数的一部分。

结果形式如下所示：

place	_time	地震发生的时间差
阿拉斯加 Anchor Point 北 32km	2018-04-04 19:51:19.147	
阿拉斯加 Healy 东北 6km	2018-04-04 16:26:14.741	03:25:04.406
阿拉斯加瓦尔迪兹东北 34km	2018-04-04 16:21:57.040	0:04:17.701
阿拉斯加费尔班克斯东北 23km	2018-04-04 16:10:05.595	0:11:51.445
阿拉斯加 Cantwell 东偏南 53km	2018-04-04 16:07:04.498	0:03:01.097
阿拉斯加科迪亚克东南 254km	2018-04-04 13:57:06.180	2:09:58.318
阿拉斯加北极村东偏北 114km	2018-04-04 12:08:00.384	1:49:05.796
阿拉斯加拉森湾东偏北 13km	2018-04-04 11:49:21.816	0:18:38.568
阿拉斯加 Cantwell 西 109km	2018-04-04 11:25:36.307	0:23:45.509
阿拉斯加塔尔基特纳西北 107km	2018-04-04 10:26:21.610	0:59:14.697

3. 计算连续交易之间的时间差

此示例使用搜索教程中的示例数据，但应该可以与任意格式的 Apache Web 访问日志配合使用。要在您自己的 Splunk 实例中尝试此示例，您必须下载示例数据并遵循操作以将教程数据导入 Splunk。当运行搜索时，请使用时间范围昨天。

计算连续交易之间的时间差。

```
sourcetype=access_* | transaction JSESSIONID clientip startswith="view" endswith="purchase" | delta _time AS timeDelta p=1 | eval timeDelta=abs(timeDelta) | eval timeDelta=tostring(timeDelta,"duration")
```

- 此示例将具有相同 `JSESSIONID` 和 `clientip` 值的事件归组到交易中，
- 交易的开始由包含字符串 `view`（查看）的事件定义。交易的结束由包含字符串 `purchase`（购买）的事件定义。关键字 `view` 和 `purchase` 对应 `action` 字段的值。您还可能会注意到 `action` 字段的其他值，如 `addtocart`（添加到购物车）和 `remove`（移除）。
- 然后通过管道符把各交易传递给 `delta` 命令，该命令使用 `_time` 字段计算某一笔交易与上一笔交易之间的时间。具体来说是交易中的最后事件时间戳和之前交易的最后事件时间戳之差。
- 搜索将时间更改重命名为 `timeDelta`。
- `eval` 命令与 `abs` 函数一起使用以将时间转换为时间的绝对值。此转换是必要的，因为一次交易和之前交易之差为负值。
- 另一个 `eval` 命令与 `tostring` 函数一起使用以将时间（以秒为单位）转换为字符串值。`duration` 参数是指定将值转换成可读时间格式 HH:MM:SS 的 `tostring` 函数的一部分。

i	时间	事件
>	18/06/04 18:18:58.000	<pre>198.35.1.75 - - [04/Jun/2018:18:18:58] "GET /cart.do?action=view&itemId=EST-12&JSESSIONID=SD10SL2FF4ADFF53099 HTTP 1.1" 406 3907 "http://www.buttercupgames.com/product.screen?productId=SF-BVS-G01" "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/536.5 (KHTML, like Gecko) Chrome/19.0.1084.46 Safari/536.5" 959 198.35.1.75 - - [04/Jun/2018:18:18:59] "POST /cart/success.do?JSESSIONID=SD10SL2FF4ADFF53099 HTTP 1.1" 200 2568 "http://www.buttercupgames.com/cart.do?action=purchase&itemId=EST-16" "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/536.5 (KHTML, like Gecko) Chrome/19.0.1084.46 Safari/536.5" 386 duration = 1 host = www1 source = tutorialdata.zip:/www1/access.log sourcetype = access_combined_wcookie timeDelta = 00:00:02</pre>
>	18/06/04 18:18:55.000	<pre>198.35.1.75 - - [04/Jun/2018:18:18:55] "GET /product.screen?productId=SF-BVS-G01&JSESSIONID=SD10SL2FF4ADFF53099 HTTP 1.1" 500 2809 "http://www.buttercupgames.com/cart.do?action=view&itemId=EST-14" "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/536.5 (KHTML, like Gecko) Chrome/19.0.1084.46 Safari/536.5" 370 198.35.1.75 - - [04/Jun/2018:18:18:56] "POST /cart.do?action=addtocart&itemId=EST-27&productId=MB-AG-T01&JSESSIONID=SD10SL2FF4ADFF53099 HTTP 1.1" 200 2615 "http://www.buttercupgames.com/product.screen?productId=MB-AG-T01" "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/536.5 (KHTML, like Gecko) Chrome/19.0.1084.46 Safari/536.5" 838 198.35.1.75 - - [04/Jun/2018:18:18:56] "GET /product.screen?productId=SC-MG-G10&JSESSIONID=SD10SL2FF4ADFF53099 HTTP 1.1" 200 3675 "http://www.buttercupgames.com/category.screen?categoryId=SIMULATION" "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/536.5 (KHTML, like Gecko) Chrome/19.0.1084.46 Safari/536.5" 328 198.35.1.75 - - [04/Jun/2018:18:18:56] "GET /cart.do?action=addtocart&itemId=EST-27&productId=EE&JSESSIONID=SD10SL2FF4ADFF53099 HTTP 1.1" 200 1159 "http://www.buttercupgames.com/product.screen?productId=EE" "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/536.5 (KHTML, like Gecko) Chrome/19.0.1084.46 Safari/536.5" 827 duration = 2 host = www1 source = tutorialdata.zip:/www1/access.log sourcetype = access_combined_wcookie timeDelta = 00:00:03</pre>

另请参阅

命令

accum
autoregress
streamstats
trendline

diff

描述

比较两个搜索结果并返回逐行差异或二者的比较结果。欲比较的两组搜索结果由位置值 position1 和 position2 指定。这两个值默认为 1 和 2，即比较前两个结果。

根据默认设置，将比较两组搜索结果的文本（_raw 字段）。若需比较其他字段，使用 attribute 选择另一个字段即可。

语法

diff [position1=int] [position2=int] [attribute=string] [diffheader=bool] [context=bool] [maxlen=int]

可选参数

position1

数据类型：<int>

描述：在输入搜索结果表中，选择与 position2 相比的特定搜索结果。

默认值： position1=1 表示引用第一个搜索结果。

position2

数据类型：<int>

描述：在输入搜索结果表中，选择与 position1 相比的特定搜索结果。此值必须大于 position1。

默认值： position2=2 表示引用第二个搜索结果。

attribute

数据类型：<field>

描述：要在两个搜索结果之间比较的字段名。

默认值： attribute=_raw，是指事件或结果的文本。

diffheader

数据类型：<bool>

描述：如果为 true，显示较早的 diff 标题，为比较的“文件”命名。如程序员命令行 patch 命令所预期的一样，diff 标题输出了一个有效的 diff 值。

默认值： diffheader=false。

context

数据类型: <bool>
描述: 如果为 true, 与默认的统一 diff 输出相反, 选择上下文模式的 diff 输出。
默认值: context=false, 或统一。

maxlen

数据类型: <int>
描述: 控制两个事件差异内容的最大值（字节数）。若 maxlen=0, 则无限制。
默认值: maxlen=100000, 为 100KB。

示例

示例 1:

比较第一个和第三个搜索结果的 "ip" 值。

```
... | diff pos1=1 pos2=3 attribute=ip
```

示例 2:

将第 9 个搜索结果与第 10 个搜索结果进行比较。

```
... | diff position1=9 position2=10
```

另请参阅

设置

entitymerge

entitymerge命令与 Splunk Enterprise Security 结合使用。

有关此命令的信息, 请参阅《管理 Splunk Enterprise Security》中的“在 Splunk Enterprise Security 中使用 entitymerge 覆盖资产或身份数据”。

erex

描述

不知道要使用哪个正则表达式时, 请使用 erex 命令从字段中提取数据。此命令自动提取与指定示例值相似的字段值。

若您指定了 field 参数, 从 fromfield 参数中提取的值将保存到 field 中。否则, 此搜索返回一个正则表达式, 然后, 您可使用该表达式与 rex 命令提取字段。

语法

要求的语法以粗体表示。

```
erex
[<field>]
examples=<string>
[counterexamples=<string>]
[fromfield=<field>]
[maxtrainers=<integer>]
```

必要参数

examples

语法: examples=<string>, <string>...

描述: 表示针对要提取并保存到新字段中的信息, 以逗号分隔的示例值列表。若列表中包含空格, 用引号将该列表引起来。例如: "port 3351, port 3768"。

可选参数

counterexamples

语法: counterexamples=<string>, <string>...

描述：代表不提取信息的以逗号分隔的示例值列表。

field

语法：<string>

描述：新字段的名称，该字段接受从 `fromfield` 参数中提取到的值。如果未指定 `field`，则不会提取值，而是生成一个结果正则表达式并以消息的形式将其放置在 Splunk Web 中的“任务”菜单下。之后，可以将该正则表达式与 `rex` 命令结合使用，改善提取效率。

fromfield

语法：`fromfield=<field>`

描述：要从中提取信息并保存到新字段中的现有字段的名称。

默认值： `_raw`

maxtrainers

语法：`maxtrainers=<int>`

描述：要从中了解的最大数字值。该值必须介于 1 到 1000 之间。

默认值： 100

用法

在 `examples` 和 `counterexample` 参数中指定的值必须存在于通过管道符传递给 `rex` 命令的事件中。如果值不存在，则命令将失败。

若要确保 `rex` 命令对事件有效，先在不使用 `rex` 命令的情况下运行返回所需事件的搜索。然后复制要提取的字段值，并将其用于 `example` 值。单击“任务”菜单并查看根据示例生成的正则表达式。

查看正则表达式

单击 Splunk Web 中的 Job 菜单即可查看基于 `rex` 命令生成的正则表达式。请参阅示例 3。

`rex` 命令的输出记录在 `search.log` 文件中。您可以通过搜索“成功学习的正则表达式”来查看输出。`search.log` 文件位于 `$SPLUNK_HOME/var/run/splunk/dispatch/` 目录中。默认情况下，系统不会对搜索日志建立索引。请阅读《[搜索手册](#)》中的“Dispatch 目录和搜索项目”。

示例

1. 基于示例提取字段

以下搜索提取月份和日期值，如 `7/01`，并将这些值放入 `monthday` 属性中。

```
... | rex monthday examples="7/01"
```

2. 基于示例和计数器示例提取值

以下搜索提取月份和日期值，如 `7/01` 和 `7/02`，但并不提取模式，如 `99/2`。将提取值放入 `monthday` 属性中。

```
... | rex monthday examples="7/01, 07/02" counterexamples="99/2"
```

3. 基于示例提取值并返回最常见的值

此示例使用搜索教程中的示例数据。要在您自己的 Splunk 实例中尝试此示例，您必须下载示例数据并遵循操作以将教程数据导入 Splunk。当运行搜索时，请使用时间范围所有时间。

确定哪些是潜在攻击者最常用的端口。

1. 运行搜索找出端口值示例，其中包含失败的登录尝试。

```
sourcetype=secure* port "failed password"
```

<隐藏字段		所有字段	i	时间	事件
选定字段			>	18/11/18 0:15:06.000	Thu Nov 18 2018 00:15:06 mailsv1 sshd[5276]: Failed password for invalid user appserver from 194.8.74.23 port 3351 ssh2 host = mailsv source = tutorialdata.zip:/mailsv/secure.log sourcetype = secure
a host 4			>	18/11/18 0:15:06.000	Thu Nov 18 2018 00:15:06 mailsv1 sshd[1039]: Failed password for root from 194.8.74.23 port 3768 ssh2 host = mailsv source = tutorialdata.zip:/mailsv/secure.log sourcetype = secure
a source 4			>	18/11/18 0:15:06.000	Thu Nov 18 2018 00:15:06 mailsv1 sshd[5258]: Failed password for invalid user testuser from 194.8.74.23 port 3626 ssh2 host = mailsv source = tutorialdata.zip:/mailsv/secure.log sourcetype = secure
a sourcetype 1			>	18/11/18 0:15:06.000	Thu Nov 18 2018 00:15:06 mailsv1 sshd[1165]: Failed password for apache from 194.8.74.23 port 4604 ssh2 host = mailsv source = tutorialdata.zip:/mailsv/secure.log sourcetype = secure
感兴趣的字段					
# date_hour 1					
# date_mday 8					
# date_minute 1					
a date_month 1					
# date_second 4					
a date_wday 7					
# date_year 1					
a date_zone 1					

2. 然后再使用 `rex` 命令提取端口字段。您必须使用 `rex` 命令指定若干示例。使用 `top` 命令返回最常见的端口值。默认情况下，`top` 命令返回前 10 个值。

```
sourcetype=secure* port "failed password" | rex port examples="port 3351, port 3768" | top port
```

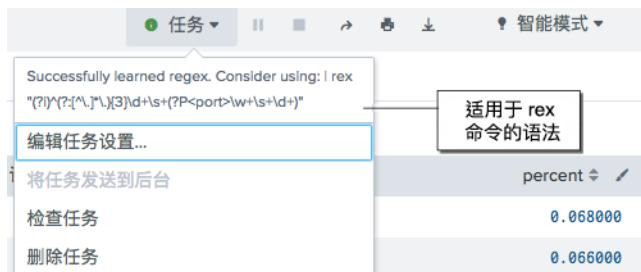
此搜索返回表格，表格中包含与搜索匹配的 `top` 端口的计数。

统计选项卡中显示的结果如下所示：

port	count	百分比
port 2444	20	0. 060145
port 3281	19	0. 057138
port 2842	19	0. 057138
port 2760	19	0. 057138
port 1174	19	0. 057138
port 4955	18	0. 054130
port 1613	18	0. 054130
port 1059	18	0. 054130
port 4542	17	0. 051123
port 4519	17	0. 051123

3. 单击任务菜单并查看根据示例生成的正则表达式。您可以使用包含正则表达式的 `rex` 命令，而不是用 `erex` 命令。此搜索示例的正则表达式为

```
| rex (?i)^(?:[^\.]*\.)\{3\}\d+\s+(?P<port>\w+\s+\d+)
```



您可以用 `rex` 命令和搜索中生成的正则表达式替换 `erex` 命令。例如：

```
sourcetype=secure* port "failed password" | rex (?i)^(?:[^\.]*\.)\{3\}\d+\s+(?P<port>\w+\s+\d+) | top port
```

结合使用 `rex` 命令和正则表达式比使用 `erex` 命令更经济有效。

另请参阅

命令

```
extract  
kvform  
multikv  
regex  
rex  
xmlkv
```

Eval

描述

eval 命令计算表达式并将结果值放入搜索结果字段中。

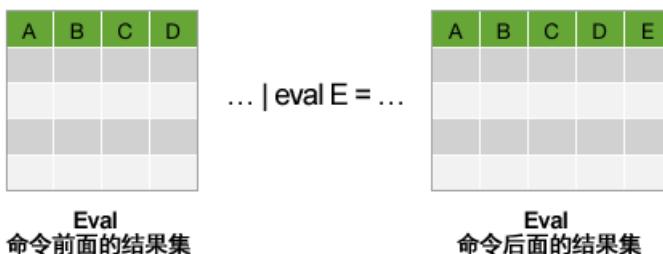
- 如果您指定的字段名称和输出中的字段不匹配，将新字段添加到搜索结果中。
- 如果您指定的字段名称和搜索结果中已经存在的字段名称匹配，eval 表达式中的结果会覆盖该字段中的值。

eval 命令评估数学表达式、字符串表达式及布尔表达式。

您可在一个搜索中使用逗号将多个 Eval 表达式链接起来，以分隔后续表达式。该搜索从左至右处理多个 Eval 表达式，并允许您在后续表达式中引用之前已评估过的字段。

Eval 和 stats 命令的区别

stats 命令根据事件中的字段计算统计信息。eval 命令使用现有字段和任意表达式在您的事件中新建字段。



语法

```
eval <field>=<expression>[,"<field>=<expression>"]...
```

必要参数

field

语法: <string>

描述: 生成的计算值的目标字段名称。如果字段名称已经存在于您的事件，Eval 将覆盖该值。

表达式

语法: <string>

描述: 用来确定您目标字段值位置的值、变量、运算符以及函数的组合。

运行搜索前会先检查 eval 表达式的语法；若表达式无效，会引发异常。

- * eval 表达式的结果不能是布尔结果。
- * 如果在搜索时间里，表达式无法对给定事件求值，则 eval 将清除结果字段。
- * 若表达式引用了包含非字母数字字符而不是下划线（_）字符的字段名称，那么需要用单引号将字段名称括起。例如，若字段名称为 server-1，您可以将字段名称指定为 new=count+'server-1'。
- * 若表达式引用了文本字符串，需要用双引号将文本字符串引起来。例如，若您要使用的字符串为 server-，您可以将其指定为 new="server-".host。

用法

eval 命令属于可分配的流命令。请参阅“命令类型”。

常规

您必须为从 eval 命令表达式返回的结果指定字段名称。您可以为新字段或现有字段指定名称。

如果您指定的字段名与现有的字段名匹配，则现有字段中的值将被 eval 表达式的结果替换。

能给字段分配数字和字符串，但不能分配布尔值。但是，您可以使用 `toString()` 将布尔值和空值转换为能分配给字段的字符串。

若您把某搜索用作 eval 命令和函数的参数，则无法使用已保存的搜索名称；您必须传递一个文本搜索字符串或一个包含文本搜索字符串（如提取自 `index=_audit` 事件的 'search' 字段）的字段。

数值计算

在计算期间，数字被视为双精度浮点数，受浮点数的所有通常行为制约。如果计算得出浮点特殊值 NaN，则在结果中会表示为 "nan"。正无穷大和负无穷大的特殊值在结果中会分别表示为 "inf" 和 "-inf"。除以零会导致空字段。

在某些情况下，计算结果包含的位数可能超过浮点数所代表的位数。此时，最低有效数字的精度可能会丢失。有关如何修正此错误的示例，请参阅 `sigfig(X)` 函数的基本示例中的“示例 2”。

支持的函数

您可以将 eval 命令与一系列函数结合使用。有关使用函数的一般信息，请参阅“评估函数”。

- 若需按类别排列的函数列表，请参阅“按类别排列的函数列表”。
- 若需按字母顺序排列的函数列表，请参阅“按字母顺序排列的函数列表”。

运算符

以下表格中列出了您可以通过 eval 命令执行的基本运算。为使这些计算正常运行，值对于运算类型而言必须有效。例如，除了加法之外，如果值不是数字，则算术运算将不会生成有效的结果。若对值进行连接，无论这些值为何，Splunk 软件都会读成字符串。

类型	运算符
算术	+ - * / %
连接	.
布尔值	AND OR NOT XOR <> <= >= != == LIKE

生成数字的运算符

- 运算符（+）将接受两个数字做加法运算，或接受两个字符串来进行连接。
- 减法（-）、乘法（*）、除法（/）和取模（%）运算符接受两个数字为参数。

生成字符串的运算符

- 句点（.）运算符用于连接字符串和数字。数字以字符串表示形式进行连接。

生成布尔值的运算符

- 运算符 AND、OR 和 XOR 接受两个布尔值。
- 运算符 <、>、<=、>=、!=、= 和 == 接受两个数字或两个字符串。
- 在表达式中，单个等号（=）等同于两个等号（==）。
- 运算符 LIKE 接受两个字符串为参数。这是类似于 SQL 中使用的模式匹配。例如，`string LIKE pattern`。模式运算符支持文字文本，一个用作通配符的百分比字符（%）和一个用作单个字符匹配的下划线字符（_）。例如，字段 `LIKE "a%b"` 将匹配任何满足以下条件的字符串：以 a 开头，之后依次跟有任意内容、b 和一个字符。

字段名称

要用多个单词指定字段名称，您可以将单词连接在一起或使用单引号指定名称。例如，要指定字段名帐户 ID，您可以指定 `AccountID` 或 'Account ID'。

要用特殊字符指定字段名，如一段时间，请使用单引号。例如，要指定字段名 `Last.Name`，请使用 'Last.Name'。

您还可以使用大括号 {} 把另一个字段的值用作目标字段的名称。例如，若事件中包含 `aName=counter` 和 `aValue=1234`。使用 `| eval {aName}=aValue` 返回 `counter=1234`。

已计算字段

您可以使用 eval 语句定义计算的字段，做法为：定义 eval 语句，该语句位于 props.conf 中。如果使用 Splunk Cloud，则可以使用 Splunk Web 通过选择设置 > 字段 > 已计算字段来定义已计算字段。当您运行某个搜索时，Splunk 软件会采用与搜索时间字段提取的计算相似的方式对语句进行评估并新建字段。设置已计算字段意味着不需要再在搜索字符串中定义 Eval 语句。反之，可以直接对生成的已计算字段进行搜索。

您可以使用计算的字段把常用的 Eval 语句移出搜索字符串，并移入 props.conf 文件。后台会在搜索时间在该文件中处理这些语句。如果使用计算的字段，那么可以将以下的搜索更改为：

```
sourcetype="cisco_esa" mailfrom=* | eval accountname=split(mailfrom,"@"), from_user=mvindex(accountname,0),  
from_domain=mvindex(accountname,-1) | table mailfrom, from_user, from_domain
```

到此搜索：

```
sourcetype="cisco_esa" mailfrom=* | table mailfrom, from_user, from_domain
```

在此示例中，定义了 accountname、from_user 和 from_domain 字段的搜索中包含了三个 Eval 语句，这三个语句在后台计算的同时，所有包含提取的字段 mailfrom 的事件都在运行前述搜索。您也可以选择另一种做法，即在 props.conf 中把这些字段设置为计算的字段后，再单独在这些字段上执行搜索。例如，您可以在 from_domain=email.com 上执行搜索。

更多有关计算的字段的信息，请参阅知识管理器手册中的“有关计算的字段”。

搜索事件标记

若您在搜索事件标记中使用 eval 命令，部分评估函数可能无法使用或产生不同的行为。请参阅《仪表板和可视化》中的“搜索标记自定义逻辑”了解哪些评估函数可用于搜索事件标记。

基本示例

1. 新建包含计算结果的新字段

在每个事件中新建一个名为 velocity 的字段。将距离字段的值除以时间字段的值得出速率。

```
... | eval velocity=distance/time
```

2. 使用 if 函数分析字段值

在每个事件中新建一个名为 error 的字段。使用 if 函数，如果 status 值为 200，将 error 字段中的值设置为 OK。否则，将 error 字段的值设置为 Problem。

```
... | eval error = if(status == 200, "OK", "Problem")
```

3. 把值转换为小写

在每个名为 low-user 的事件中新建字段。使用 lower 函数，用 username 字段中的值的小写版本填充字段。因为字段名称包含破折号（-），所以必须用单引号将名称括起来。

```
... | eval 'low-user' = lower(username)
```

4. 使用一个字段的值作为新字段的名称

在本示例中，使用字段 counter 的值作为新字段名。将 value 字段的值分配给新字段。请参阅“用法”一节中的“字段名称”。

```
index=perfmon sourcetype=Perfmon* counter=* Value=* | eval {counter} = Value
```

5. 将 sum_of_areas 设置为两个圆面积的总和。

```
... | eval sum_of_areas = pi() * pow(radius_a, 2) + pi() * pow(radius_b, 2)
```

6. 将 status 设置为一些简单的 http 错误代码

```
... | eval error_msg = case(error == 404, "Not found", error == 500, "Internal Server Error", error == 200, "OK")
```

7. 连接两个字段中的值

使用句号（.）字符将 `first_name` 字段中的值与 `last_name` 字段中的值连接。引号用于在两个名称之间插入空格字符。连接时，不管实际值如何，值作为字符串读取。

```
... | eval full_name = first_name." ".last_name
```

8. 用逗号分隔多个 `Eval` 运算

您可以使用逗号分隔运算来指定多个 `eval` 运算。在以下搜索中，`full_name` 评估使用句号（.）字符连接 `first_name` 字段中的值与 `last_name` 字段中的值。`low_name` 评估使用 `lower` 函数，将 `full_name` 评估转换为小写。

```
... | eval full_name = first_name." ".last_name, low_name = lower(full_name)
```

9. 将数字字段值转换为含有逗号和 2 位小数的字符串

如果原始值 `x` 为 1000000，则以 1,000,000 形式返回 `x`。

```
... | eval x=toString(x,"commas")
```

要在字符串的开始处添加货币符号：

```
... | eval x="$".toString(x,"commas")
```

`x` 将返回为 \$1,000,000。

延伸示例

1. 合并来自两个不同来源类型的字段并新建事件的交易

此示例显示如何合并来自两个不同来源类型的字段并使用该字段新建事件的交易。`sourcetype=A` 包含一个名为 `number` 的字段，`sourcetype=B` 则在一个名为 `subscriberNumber` 的字段中包含相同的信息。

```
sourcetype=A OR sourcetype=B | eval phone=coalesce(number,subscriberNumber) | transaction phone maxspan=2m
```

`eval` 命令用于把一个名为 `phone` 的通用字段添加到各事件中，无论这些事件是来自 `sourcetype=A` 或 `sourcetype=B`。`phone` 的值被定义为（使用 `coalesce()` 函数定义）`number` 和 `subscriberNumber` 的值。`coalesce()` 函数获取第一个非空字段（亦即，该字段存在于事件中）的值。

无论事件来自来源类型 A 或 B，只要具有相同的 `phone` 值，您现在都可以对这些事件进行分组。

2. 将事件分隔成类别，计算和显示最小和最大值

此示例使用从 USGS 地震网站下载的近期地震数据。该数据是一个逗号分隔的 ASCII 文本文件，其中包含记录的每次地震的震级（mag）、坐标（经度、纬度）、区域（地点）等。

您可以从 [USGS 地震源](#) 下载当前 CSV 文件，然后将文件上传至您的 Splunk 实例（如果您想要遵循此示例的做法）。

震源深度小于 70 公里的地震属于浅源地震，而那些震源深度在 70 到 300 公里之间的地震通常称为中源地震。在俯冲带，深源地震的震源深度会更深（可深达 300 到 700 公里）。

要根据深度对最近的地震进行分类，您可以使用以下搜索。

```
source=all_month.csv | eval Description=case(depth<=70, "Shallow", depth>70 AND depth<=300, "Mid", depth>300, "Deep") | stats count min(mag) max(mag) by Description
```

`eval` 命令用于新建一个名为 `Description` 的字段，该字段根据地震的 Depth 获取 "Shallow"、"Mid" 或 "Deep" 的值。`case()` 函数用于指定适合每种描述的深度等级。例如，若深度不足 70 公里，该地震将被描述为浅源地震，生成的描述 `Description` 因此为 `Shallow`。

该搜索还会通过管道符把 `eval` 命令的结果传递给 `stats` 命令，来统计地震的数量并显示每个描述的最大和最小震级。

统计选项卡中显示的结果如下所示：

描述	count	min(Mag)	max(Mag)
Deep	35	4.1	6.7
Mid	635	0.8	6.3

Shallow	6236	-0.60	7.70
---------	------	-------	------

3. 查找 IP 地址并使用 Eval 函数 cidrmatch 和 if 按网络分类

此示例使用搜索教程中的示例数据，但应该可以与任意格式的 Apache Web 访问日志配合使用。要在您自己的 Splunk 实例中尝试此示例，您必须下载示例数据并遵循操作以将教程数据导入 Splunk。当运行搜索时，请使用时间范围昨天。

在此搜索中，您将查找 IP 地址并对它们所属的网络进行分类。

```
sourcetype=access_* | eval network=if(cidrmatch("182.236.164.11/16", clientip), "local", "other")
```

本示例使用 `cidrmatch()` 函数将 `clientip` 字段中的 IP 地址和子网范围进行比较。搜索还使用 `if()` 函数，表明如果 `clientip` 的值在子网范围，那么 `network` 字段值是 `local`。否则，分配到的将是 `network=other`。

`eval` 命令没有对您的结果进行任何特殊格式设置。命令根据您指定的 `eval` 表达式新建字段。

在字段边栏中，单击 `network` 字段。在弹出窗口中，在已选旁单击是，然后关闭弹出窗口。现在，您可以在搜索结果中看到哪些 IP 地址是您的 `local` 网络的一部分，而哪些则不是。您的事件列表看起来应如下所示：

时间	事件
18/06/04 18:22:16.000	91.205.189.15 - - [04/Jun/2018:18:22:16] "GET /oldlink?itemId=EST-14&JSESSIONID=SD6SL7FF7ADFF53113 HTTP 1.1" 200 1665 "http://www.buttercupgames.com/oldlink?itemId=EST-14" "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/536.5 (KHTML, like Gecko) Chrome/19.0.1084.46 Safari/536.5" 159 host = www2 network = other source = tutorialdata.zip://www2/access.log sourcetype = access_combined_wcookie
18/06/04 18:22:15.000	91.205.189.15 - - [04/Jun/2018:18:22:15] "GET /category.screen?categoryId=SHOOTER&JSESSIONID=SD6SL7FF7ADFF53113 HTTP 1.1" 200 1369 "http://www.google.com" "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/536.5 (KHTML, like Gecko) Chrome/19.0.1084.46 Safari/536.5" 779 host = www2 network = other source = tutorialdata.zip://www2/access.log sourcetype = access_combined_wcookie
18/06/04 18:20:56.000	182.236.164.11 - - [04/Jun/2018:18:20:56] "GET /cart.do?action=addtocart&itemId=EST-15&productId=BS-AG-G09&JSESSIONID=SD6SL8FF10ADFF53101 HTTP 1.1" 200 2252 "http://www.buttercupgames.com/oldlink?itemId=EST-15" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_7_4) AppleWebKit/536.5 (KHTML, like Gecko) Chrome/19.0.1084.46 Safari/536.5" 506 host = www1 network = local source = tutorialdata.zip://www1/access.log sourcetype = access_combined_wcookie
18/06/04 18:20:55.000	182.236.164.11 - - [04/Jun/2018:18:20:55] "POST /oldlink?itemId=EST-18&JSESSIONID=SD6SL8FF10ADFF53101 HTTP 1.1" 408 893 "http://www.buttercupgames.com/product.screen?productId=SF-BVS-G01" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_7_4) AppleWebKit/536.5 (KHTML, like Gecko) Chrome/19.0.1084.46 Safari/536.5" 134 host = www1 network = local source = tutorialdata.zip://www1/access.log sourcetype = access_combined_wcookie

为结果设置格式的另一种做法，是通过管道符把 `eval` 的结果传递给 `table` 命令；如此一来，该命令仅显示您感兴趣的字段。

注意：本示例仅说明如何使用 `cidrmatch` 函数。如果您要对事件进行分类并快速查找这些事件，更好的方法是使用事件类型。请阅读《知识管理器手册》中的关于事件类型。

4. 从事件中提取信息并加入另一个字段，新建一个多值字段

此示例使用示例电子邮件数据。您可以将 `sourcetype=cisco:esa` 替换为 `sourcetype` 值并将 `mailfrom` 字段替换为您数据的电子邮件地址字段名，从而实现对任何电子邮件数据运行此搜索。例如：电子邮件可能为 To、From 或 Cc）。

使用电子邮件地址字段提取名称和域。本搜索中的 `eval` 命令包含多个以逗号隔开的表达式。

```
sourcetype="cisco:esa" mailfrom=* | eval accountname=split(mailfrom,"@"), from_user=mvindex(accountname,0), from_domain=mvindex(accountname,-1) | table mailfrom, from_user, from_domain
```

- `split()` 函数用于将 `mailfrom` 字段分成为 `accountname` 的多值字段。`accountname` 的第一个值是 "@" 符号前的所有内容，第二个值则为该符号后的所有内容。
- `mvindex()` 函数用于将 `from_user` 设置为 `accountname` 中的第一个值，将 `from_domain` 设置为 `accountname` 中的第二个值。
- `eval` 表达式的结果随后再通过管道符传递给 `table` 命令。

您可以在结果表格中看到原始的 `mailfrom` 值及新的 `from_user` 和 `from_domain` 值。统计选项卡中显示的结果如下所示：

mailfrom	from_user	from_domain
na.lui@example.net	na.lui	sample.net
MAILER-DAEMON@hcp2mailsec.sample.net	MAILER-DAEMON	hcp2mailsec.sample.net

M&MService@example.com	M&MService	example.com
AlexMartin@oursample.de	AlexMartin	oursample.de
Exit_Desk@sample.net	Exit_Desk	sample.net
buttercup-forum+SEMA8PUC4RETTUB@groups.com	buttercup-forum+SEMA8PUC4RETTUB	groups.com
eduardo.rodriguez@sample.net	eduardo.rodriguez	sample.net
VC00110489@techexamples.com	VC00110489	techexamples.com

注意：此示例用于演示如何使用 eval 函数识别多值字段的各个值。由于这组特定的电子邮件数据不包含任何多值字段，本示例新建了一个多值字段 accountname，新建来源为单值字段 mailfrom。

5. 使用 *match* 函数分类事件

此示例使用示例电子邮件数据。您可以将 sourcetype=cisco:esa 替换为 sourcetype 值并将 mailfrom 字段替换为您数据的电子邮件地址字段名，从而实现对任何电子邮件数据运行此搜索。例如：电子邮件可能为 To、From 或 Cc）。

此示例将根据电子邮件地址的域对电子邮件的来源进行分类。将 .com、.net 和 .org 地址视为 **local**，而将任何其他地址视为 **abroad**。域名有很多。当然，.com、.net 或 .org 以外的域不一定属于 **abroad**。这只是一个示例。

本搜索中的 eval 命令包含多个以逗号隔开的表达式。

```
sourcetype="cisco:esa" mailfrom=* | eval accountname=split(mailfrom,"@"), from_domain=mvindex(accountname,-1), location;if(match(from_domain, "[^\\n\\r\\s]+\\.\\(com|net|org\\)"), "local", "abroad") | stats count BY location
```

此搜索的前半部分与之前的示例相似。split() 函数用于拆分 mailfrom 字段中的电子邮件地址。mvindex 函数将 from_domain 定义为 mailfrom 字段 @ 符号之后的部分。

然后，使用 if() 和 match() 函数。

- 如果 from_domain 值以 .com、.net.， or .org 结尾，则为 location 字段分配值 local。
- 如果 from_domain 不匹配，则为 location 分配值 abroad。

然后，eval 结果传递到 stats 命令以对每个 location 值的结果数进行计数。

统计选项卡中显示的结果如下所示：

位置	count
abroad	3543
local	14136

注意：此示例只是说明如何使用 match() 函数。如果您要对事件进行分类并快速查找这些事件，更好的方法是使用事件类型。请阅读《知识管理器手册》中的关于事件类型。

6. 将交易的持续时间转换为更易于理解的字符串格式

此示例使用搜索教程中的示例数据，但应该可以与任意格式的 Apache Web 访问日志配合使用。要在您自己的 Splunk 实例中尝试此示例，您必须下载示例数据并遵循操作以将教程数据导入 Splunk。当运行搜索时，请使用时间范围昨天。

当您使用 transaction 命令时，正如以下搜索中显示的那样，它计算交易的时间长度。自动将名为 duration 的新字段添加到结果中。duration 是交易中首个和最后一个事件之间的时间。

```
sourcetype=access_* | transaction clientip maxspan=10m
```

在感兴趣字段列表中，单击 duration 字段以查看持续时间的前 10 个值。以秒为单位显示值。单击是将字段添加到已选字段列表中。

您可以使用 eval 命令将数字字段重新调成更可读的字符串格式。以下搜索使用有 "duration" 选项的 tostring() 函数将 duration 字段中的值转换成格式为 HH:MM:SS 的字符串。

```
sourcetype=access_* | transaction clientip maxspan=10m | eval durationstr=tostring(duration,"duration")
```

该搜索定义了一个新的字段 `durationstr`, 用于其格式重新设置过的 `duration` 值。在感兴趣字段列表中, 单击 `durationstr` 字段并选择确定以将该字段添加到已选字段列表中。字段的值现在出现在每个交易下方的字段集中。以下图像显示您的搜索结果的样式:

隐藏字段	所有字段	i 时间	事件
选定字段			
# duration 29			
<i>a durationstr 29</i>			
<i>a host 3</i>			
<i>a source 3</i>			
<i>a sourcetype 1</i>			
感兴趣的字段			
<i>a action 5</i>			
<i>a bytes 100+</i>			
<i>a categoryId 8</i>			
<i>a clientip 100+</i>			
<i># closed_bxn 2</i>			
<i># date_hour 19</i>			
<i># date_mday 1</i>			
<i># date_minute 60</i>			
<i>a date_month 1</i>			
<i>a date_second 60</i>			
<i>a date_wday 1</i>			
<i># date_year 1</i>			
<i>a date_zone 1</i>			
显示所有 11 行			
<code>duration = 6 : durationstr = 00:00:06 host = www1 source = tutorialdata.zip://www1/access.log sourcetype = access_combined_wcookie</code>			

另请参阅

函数

评估函数

命令

where

eventcount

描述

在特定索引中返回事件数。

语法

要求的语法以粗体表示。

```
| eventcount
[index=<string>]...
[summarize=<bool>]
[report_size=<bool>]
[!list_vix=<bool>]
```

必要参数

无。

可选参数

index

语法: `index=<string>`

描述: 报告的索引名称, 或与报表的多个索引匹配的通配符。您可以指定此参数多次, 例如: `index=*` `index=_*`。

默认值: 若未指定任何索引, 命令将返回有关默认索引的信息。

list_vix

语法: `list_vix=<bool>`

描述: 指定是否列出虚拟索引。若 `list_vix=false`, 命令不会列出虚拟索引。

默认值: `true`

report_size

语法: `report_size=<bool>`

描述: 指定是否报告索引大小。若 `report_size=true`, 命令返回以字节为单位的索引大小。

默认值: `false`

summarize

语法: summarize=<bool>

描述: 指定是否汇总所有对等节点和索引的事件。若 summarize=false, 命令将按索引和搜索节点拆分事件统计结果。

默认值: true

用法

eventcount 命令属于报表生成命令。请参阅“命令类型”。

生成命令使用前导管道符且应是搜索中的第一个命令。

指定时间范围对由 eventcount 命令返回的结果没有影响。您指定的索引上的所有事件都计数。

指定索引

您无法指定索引以将其排除在结果之外。例如: index!=foo 是无效的语法。

您可以指定 index 参数多次。例如:

```
|eventcount summarize=false index=_audit index=main
```

示例

示例 1:

显示默认索引中所有搜索节点的事件计数。返回单个计数。

```
| eventcount
```

示例 2:

仅返回内部默认索引中事件的数量。结果中包括索引大小（以字节为单位）。

```
| eventcount summarize=false index=_* report_size=true
```

结果出现在统计选项卡中，应与在以下表格中显示的结果类似。

count	index	server	size_bytes
52550	_audit	buttercup-mbpr15.sv.splunk.com	7217152
1423010	_internal	buttercup-mbpr15.sv.splunk.com	122138624
22626	_introspection	buttercup-mbpr15.sv.splunk.com	98619392
10	_telemetry	buttercup-mbpr15.sv.splunk.com	135168
0	_thefishbucket	buttercup-mbpr15.sv.splunk.com	0

当您指定 summarize=false 时，命令返回三个字段：count、index 和 server。当您指定 report_size=true 时，命令返回 size_bytes 字段。size_bytes 字段中的值与磁盘上索引大小不同。

示例 3:

返回每个索引和服务器对的事件计数。只返回外部索引。

```
| eventcount summarize=false index=*
```

此图像显示四行，每个索引和服务器组合显示一行。有三列：计数、索引和服务器。

要返回包括内部索引的所有索引计数，您必须指定独立于外部索引的内部索引：

```
| eventcount summarize=false index=_* index=_*
```

另请参阅

元数据、字段摘要

eventstats

描述

为事件中的字段生成摘要统计信息并将这些信息保存在新字段中。

生成摘要统计信息的过程只会使用包含与聚合相关的字段的那些事件。生成的摘要统计信息可用于搜索中后续命令中的计算。请参阅“用法”。

语法

要求的语法以粗体表示。

```
eventstats
[allnum=<bool>]
<stats-agg-term> ...
[<by-clause>]
```

必要参数

<stats-agg-term>

语法： <stats-func> (<evalued-field> | <wc-field>) [AS <wc-field>]

描述： 统计聚合函数。请参阅“Stats 函数”选项。该函数可应用于 Eval 表达式、字段或字段组。使用 AS 子句将结果放入您已指定其名称的新字段内。可以在字段名称中使用通配符字符。

可选参数

allnum

语法： allnum=<bool>

描述： 如果设置为 true，则计算每个字段的数字统计信息（当且仅当该字段的所有值均为数字时）。如果有 BY 子句，则 allnum 参数会分别应用于每个组。

默认值： false

<by-clause>

语法： BY <field-list>

描述： 分组所依据的一个或多个字段的名称。

Stats 函数选项

stats-func

语法： 语法取决于您使用的函数。请参考下方表格。

描述： 可与 eventstats 命令一起使用的统计和图表函数。每次调用 eventstats 命令时都可以使用一个或多个函数。但是，只能使用一个 BY 子句。请参阅“用法”。

以下表格按照函数类型列出支持的函数。使用表格中的链接查看每个函数的介绍和示例。有关使用带有命令的函数的概述，请参见统计和图表函数。

函数类型	支持的函数和语法			
聚合函数	avg() count() distinct_count() estdc() estdc_error()	exactperc<int>() max() median() min() mode()	perc<int>() range() stdev() stdevp()	sum() sumsq() upperperc<int>() var() varp()
事件顺序函数	earliest()	first()	last()	latest()

用法

`eventstats` 搜索处理器使用名为 `max_mem_usage_mb` 的 `limits.conf` 设置来限制 `eventstats` 命令可用于跟踪信息的内存量。当达到限制时，`eventstats` 命令处理器会停止将请求的字段添加到搜索结果中。

不要设置 `max_mem_usage_mb=0`，因为这会取消对 `eventstats` 命令处理器可以使用的内存量的限制。这可能会导致搜索失败。

前提条件

- 只有具有文件系统访问权限的用户，如系统管理员，才能使用配置文件增加 `maxresultrows` 和 `max_mem_usage_mb` 的值。
- 请参阅 Splunk Enterprise《管理员手册》中的“如何编辑配置文件”了解具体步骤。
- 您可以有几个具有相同名称的配置文件，分散在默认目录、本地目录和应用目录中。请参阅 Splunk Enterprise《管理员手册》中“在何处放置（或查找）已修改的配置文件”。

不要更改或复制默认目录中的配置文件。默认目录中的文件必须保持原样并位于其原始位置。更改本地目录中的文件。

如果您有 Splunk Cloud 并想更改这些限制，请提交支持工单。

`eventstats` 和 `stats` 的差异

`eventstats` 命令与 `stats` 命令相似。这两个命令都可用于生成聚合，例如平均值、总和和最大值。

下表描述了这两个命令之间的差异：

stats 命令	eventstats 命令
事件会经过转换并放入汇总搜索结果表	聚合被放置在一个新字段中，该字段会添加到输出中的每个事件
只能在搜索的后续命令中使用汇总结果中的字段	可以在搜索的后续命令中使用事件中的字段，因为事件尚未转换

`eventstats` 如何生成聚合

`eventstats` 命令会查找特定事件，这些事件包含要用于生成聚合的字段。该命令在每个事件中创建一个新字段，并将聚合放置在该字段中。每个事件中都会添加聚合，甚至包括未用于生成聚合的事件。

例如，您有 5 个事件，其中 3 个事件有您要聚合的字段。`eventstats` 命令基于这 3 个事件中的数据生成聚合。每个事件中都会添加一个新字段，而且生成的聚合会放入每个事件的这个新字段中。

不适用于特定字段的统计函数

除 `count` 函数外，当您将 `eventstats` 命令与未应用于特定字段的函数或解析为字段的 `eval` 表达式配对使用时，搜索头对它的处理方式类似于为它应用了一个针对所有字段的通配符。换句话说，当搜索中有 `| eventstats avg`，则会返回针对 `| eventstats avg(*)` 的搜索结果。

但是，这种“隐式通配符”语法已正式弃用。使通配符采用显式形式。若想让一个函数应用于所有可能的字段时，请使用 `| eventstats <function>(*)`。

函数和内存用法

就内存而言，某些函数本身就比其他函数占用更多内存。例如，`distinct_count` 函数需要的内存比 `count` 函数大得多。`values` 和 `list` 函数也会消耗大量的内存。

如果您使用的是未结合 `split-by` 字段或结合按字段的低基数 `split-by` 的 `distinct_count` 函数，考虑用 `estdc` 函数替换 `distinct_count` 函数（估计非重复计数）。`estdc` 函数可降低内存使用和减少运行时间。

事件顺序函数

基于时间搜索时使用 `first` 和 `last` 函数不会产生精确结果。

- 要根据时间顺序查找第一个值，请使用 `earliest` 函数，而非 `first` 函数。
- 要根据时间顺序查找最后一个值，请使用 `latest` 函数，而非 `last` 函数。

例如，看一下以下搜索。

```
index=test sourcetype=testDb | eventstats first>LastPass as LastPass, last(_time) as mostRecentTestTime BY testCaseId | where
```

```
startTime==LastPass OR _time==mostRecentTestTime | stats first(startTime) AS startTime, first(status) AS status, first(histID) AS currentHistId, last(histID) AS lastPassHistId BY testCaseId
```

使用 stats 和 eventstats，根据时间为事件进行排序时，使用 earliest 和 latest 函数。

以下搜索与之前的搜索相同，除了 first 和 last 函数替换成 earliest 和 latest 函数。

```
index=test sourcetype=testDb | eventstats latest(LastPass) AS LastPass, earliest(_time) AS mostRecentTestTime BY testCaseId | where startTime==LastPass OR _time==mostRecentTestTime | stats latest(startTime) AS startTime, latest(status) AS status, latest(histID) AS currentHistId, earliest(histID) AS lastPassHistId BY testCaseId
```

基本示例

1. 计算总体平均持续时间

此示例使用搜索教程中的示例数据，但应该可以与任意格式的 Apache Web 访问日志配合使用。要在您自己的 Splunk 实例中尝试此示例，您必须下载示例数据并遵循操作以将教程数据导入 Splunk。当运行搜索时，请使用时间范围所有时间。

计算一组交易的总体平均持续时间，然后将计算结果放在一个名为 avgdur 的新字段中。

```
host=www1 | transaction clientip host maxspan=30s maxpause=5s | eventstats avg(duration) AS avgdur
```

由于未指定 BY 子句，因此将生成单个聚合，而且此聚合会添加到每个事件的名为 avgdur 的新字段中。

当您查看“感兴趣的字段”列表时，您将看到 avgdur 只有一个值。

选定字段
a host 1
a source 2
a sourcetype 2

感兴趣的字段
a action 5
avgdur 1
a bytes 100+
a categoryid 8
a clientip 100+
closed_txn 2
date_hour 24
date_mday 8
date_minute 60

avgdur
1 个值, 100% 的事件
选定的 是 否
报表
时段平均值 时段最大值 时段最小值
上限值 时段上限值 罕见值
具有此字段的事件
平均: 4.391786903440622 最小值: 4.391786903440622 最大值: 4.391786903440622
标准偏差: 0.000009741794773875846
值 计数 %
4.391786903440622 1,802 100%

2. 计算按特定字段分组的平均持续时间

除了会为 date_minute 字段的每个非重复值计算平均值之外，此示例与之前示例并无不同。每个事件中会添加一个新字段 avgdur，其值为基于 date_minute 特定值计算的平均值。

```
host=www1 | transaction clientip host maxspan=30s maxpause=5s | eventstats avg(duration) AS avgdur BY date_minute
```

当您查看“感兴趣的字段”列表时，您将看到 avgdur 有 79 个值，基于时间戳、持续时间和 date_minute 值。



3. 搜索错误量的峰值

搜索错误量的峰值。比如说，如果错误数量超过平均值，您可以使用此搜索来触发一个告警。

```
eventtype="error" | eventstats avg(bytes) AS avg | where bytes>avg
```

延伸示例

以下示例可让您更好地了解 `eventstats` 命令的工作方式。该示例实际上是一组渐进式小示例，其中每个小示例建立在前一个小示例之上或对其进行扩展。

用一组简单的事件向您举例说明，可以让您更轻松地了解 `eventstats` 命令的功能。

这些示例使用 `makergesults` 命令创建一组事件。它们还使用 `streamstats` 和 `eval` 命令在事件中创建其他字段。

创建一组事件

让我们从创建一组四个事件开始。其中一个事件的 `age` 字段为空值。

```
| makergesults count=4 | streamstats count | eval age = case(count=1, 25, count=2, 39, count=3, 31, count=4, null()) | eval city = case(count=1 OR count=3, "San Francisco", count=2 OR count=4, "Seattle")
```

- `streamstats` 命令用于创建 `count` 字段。`streamstats` 命令会在处理每个事件时计算该事件的累计数量。
- `eval` 命令用于创建两个新字段：`age` 和 `city`。`eval` 命令使用 `count` 字段中的值。
- `case` 函数采用成对的参数，例如 `count=1, 25`。第一个参数是布尔表达式。当该表达式为 `TRUE` 时，将返回相应的第二个参数。

搜索结果如下所示：

_time	age	city	计数
2020/2/5 18:32:07	25	旧金山	1
2020/2/5 18:32:07	39	西雅图	2
2020/2/5 18:32:07	31	旧金山	3
2020/2/5 18:32:07		西雅图	4

使用带 BY 子句的 `eventstats`

`eventstats` 命令中的 `BY` 子句是可选的，但很常用。`BY` 子句按字段中的值将生成的统计信息进行分组。您可以将任何统计函数

搭配 eventstats 命令一起使用，以便生成统计信息。请参阅“统计和图表函数”。

在此示例中，eventstats 命令会生成每个城市的平均年龄。生成的平均值放置在一个名为 avg(age) 的新字段中。

以下搜索与前一个搜索相同，只是在最后添加了 eventstats 命令：

```
| makeresults count=4 | streamstats count | eval age = case(count=1, 25, count=2, 39, count=3, 31, count=4, null()) | eval city = case(count=1 OR count=3, "San Francisco", count=2 OR count=4, "Seattle") | eventstats avg(age) BY city
```

- 对于 San Francisco，平均年龄为 $28 = (25 + 31) / 2$ 。
- 对于 Seattle，只有一个事件有值。平均值是 $39 = 39 / 1$ 。eventstats 命令将该平均值放入西雅图的每个事件中（包括不包含 age 值的事件）。

搜索结果如下所示：

_time	age	avg(age)	city	计数
2020/2/5 18:32:07	25	28	旧金山	1
2020/2/5 18:32:07	39	39	西雅图	2
2020/2/5 18:32:07	31	28	旧金山	3
2020/2/5 18:32:07		39	西雅图	4

重命名新字段

默认情况下，所生成的新字段名称是统计计算的名称。在这些示例中，该名称为 avg(age)。您可以使用 AS 关键字重新命名新字段。

在以下搜索中，eventstats 命令经过了调整，会将新字段重命名为 average age by city。

```
| makeresults count=4 | streamstats count | eval age = case(count=1, 25, count=2, 39, count=3, 31, count=4, null()) | eval city = case(count=1 OR count=3, "San Francisco", count=2 OR count=4, "Seattle") | eventstats avg(age) AS "average age by city" BY city
```

搜索结果如下所示：

_time	age	average age by city	city	计数
2020/2/5 18:32:07	25	28	旧金山	1
2020/2/5 18:32:07	39	39	西雅图	2
2020/2/5 18:32:07	31	28	旧金山	3
2020/2/5 18:32:07		39	西雅图	4

带文本值的事件

前面的示例说明了如何处理不包含 age 字段值的事件。现在让我们看看，对于要用于生成统计信息的字段，如果事件包含该字段的字母字符值，则系统会如何处理这个事件。

以下搜索包括单词 test（作为 age 字段的值）。

```
| makeresults count=4 | streamstats count | eval age = case(count=1, 25, count=2, 39, count=3, 31, count=4, "test") | eval city = case(count=1 OR count=3, "San Francisco", count=2 OR count=4, "Seattle")
```

搜索结果如下所示：

_time	age	city	计数
2020/2/5 18:32:07	25	旧金山	1
2020/2/5 18:32:07	39	西雅图	2

2020/2/5 18:32:07	31	旧金山	3
2020/2/5 18:32:07	test	西雅图	4

让我们将 `eventstats` 命令添加到搜索中。

```
| makeresults count=4 | streamstats count | eval age = case(count=1, 25, count=2, 39, count=3, 31, count=4, "test") | eval city = case(count=1 OR count=3, "San Francisco", count=2 OR count=4, "Seattle") | eventstats avg(age) BY city
```

字母值被视为空值。搜索结果如下所示：

_time	age	avg(age)	city	计数
2020/2/5 18:32:07	25	28	旧金山	1
2020/2/5 18:32:07	39	39	西雅图	2
2020/2/5 18:32:07	31	28	旧金山	3
2020/2/5 18:32:07	test	39	西雅图	4

使用 `allnum` 参数

但是，假设您不希望在字段中包含字母字符或字段为空时生成统计信息？

`allnum` 参数控制 `eventstats` 命令处理字段值的方式。`allnum` 参数的默认设置为 `FALSE`。这意味着，用于生成统计信息的字段不需要包含所有数字值。带空值或字母字符值的字段会被忽略。您已经在前面的示例中看到了这一点。

只有当字段包含所有数字值时，才可以强制 `eventstats` 命令生成统计信息。要做到这一点，可以将 `allnum` 参数设置为 `TRUE`。

```
| makeresults count=4 | streamstats count | eval age = case(count=1, 25, count=2, 39, count=3, 31, count=4, "test") | eval city = case(count=1 OR count=3, "San Francisco", count=2 OR count=4, "Seattle") | eventstats allnum=true avg(age) BY city
```

搜索结果如下所示：

_time	age	avg(age)	city	计数
2020/2/5 18:32:07	25	28	旧金山	1
2020/2/5 18:32:07	39		西雅图	2
2020/2/5 18:32:07	31	28	旧金山	3
2020/2/5 18:32:07	test		西雅图	4

由于 `age` 字段包含的 `Seattle`（西雅图）的值并非全部为数字，因此 `Seattle`（西雅图）的整个值集都会被忽略。没有计算平均值。

`allnum=true` 参数适用于空值以及字母字符值。

另请参阅

命令

`stats`
`streamstats`

博客

搜索命令优于 `stats`、`eventstats` 和 `streamstats`

extract

描述

从搜索结果提取字段-值对。`extract` 命令仅适用于 `raw` 字段。如果您想要从其他字段中提取，在运行 `extract` 命令之前，您必

须执行一些字段重命名。

语法

```
extract [<extract-options>...] [<extractor-name>...]
```

必要参数

无。

可选参数

<extract-options>

语法: clean_keys=<bool> | kvdelim=<string> | limit=<int> | maxchars=<int> | mv_add=<bool> | pairdelim=<string> | reload=<bool> | segment=<bool>

描述: 用于定义提取信息的选项。请参阅本主题中的 "Extract_options" 部分。

<extractor-name>

语法: <string>

描述: transforms.conf 文件中的一个段落。props.conf 文件未能显式地引发对此数据来源、来源类型或主机的提取时使用该段落。

提取选项

clean_keys

语法: clean_keys=<bool>

描述: 指定是否进行键清理。覆盖 transforms.conf 文件中的 CLEAN_KEYS。

默认值: 在 transforms.conf 文件内的 CLEAN_KEYS 中指定的值。

kvdelim

语法: kvdelim=<string>

描述: 用于分隔键和值的字符分隔符的列表。

limit

语法: limit=<int>

描述: 指定要提取多少个自动键一值对。

默认值: 50

maxchars

语法: maxchars=<int>

描述: 指定要在事件中查看的字符数。

默认值: 10240

mv_add

语法: mv_add=<bool>

描述: 指定是否新建多值字段。覆盖 transforms.conf 文件中 MV_ADD 参数的值。

默认值: false

pairdelim

语法: pairdelim=<string>

描述: 用于分隔键值对的字符分隔符的列表。

reload

语法: reload=<bool>

描述: 指定是否强制 props.conf 和 transforms.conf 文件的重新加载。

默认值: false

segment

语法: segment=<bool>

描述: 指定是否在结果中记录键一值对的位置。

默认值: false

用法

extract 命令属于可分配的流命令。请参阅 "命令类型"。

别名

`extract` 命令的别名为 `kv`。

示例

1. 指定用于字段和值提取的分隔符

提取由管道符或分号 (|;) 分隔的字段-值对。提取由等号或冒号 (=:) 分隔的字段值。分隔符为单个字符。本例使用的是 “=” 或 “:” 字符来分隔键值。同样， “|” 或 “;” 用来分隔字段-值对本身。

```
... | extract pairdelim="|;", kvdelim=":=
```

2. 提取字段-值对，然后重新加载字段提取设置

提取字段-值对，并从磁盘中重新加载字段提取设置。

```
... | extract reload=true
```

3. 将想要从该字段中提取的字段重命名为 `_raw`

将 `_raw` 字段重命名为一个临时名称。将您想要从中提取的字段重命名为 `_raw`。在本例中，字段名称为 `uri_query`。

```
... | rename _raw AS temp uri_query AS _raw | extract pairdelim="?&" kvdelim="=" | rename _raw AS uri_query temp AS _raw
```

4. 从 `transforms.conf` 文件中的段落中提取字段-值对

提取定义于 `transforms.conf` 文件内 ‘access-extractions’ 段落中的字段-值对。

```
... | extract access-extractions
```

另请参阅

`kvform`、`multikv`、`rex`、`spath`、`xmlkv`、`xpath`

fieldformat

描述

`fieldformat` 命令允许您使用 `<eval-expression>` 在呈现结果时更改字段值的格式。此命令可以更改结果外观，而无须改变字段的基本值。

因为搜索管道后面的命令无法修改已经设置格式的结果，请尽量晚点在搜索管道中使用 `fieldformat` 命令。

`fieldformat` 命令不会应用于导出数据的命令，例如 `outputcsv` 和 `outputlookup` 命令。导出的数据会保留原始数据格式，而非显示格式。若想把显示格式应用于导出的数据，请使用 `eval` 命令而非 `fieldformat` 命令。

语法

```
fieldformat <field>=<eval-expression>
```

必要参数

`<field>`

描述：`eval` 表达式输出的新字段或现有字段的名称（不含通配符）。

`<eval-expression>`

语法：`<string>`

描述：代表目标字段值的值、变量、运算符以及函数的组合。您可以用 `fieldformat` 命令只指定一个 `<eval-expression>`。要指定多个格式，您必须使用多个 `fieldformat` 命令。请参阅“示例”。

更多信息，请参阅 `eval` 命令。

有关支持的函数的信息，请参阅“使用方法”。

用法

`fieldformat` 命令属于可分配的流命令。请参阅“命令类型”。

时间格式变量通常与 `fieldformat` 命令一起使用。请参阅“日期和时间格式变量”。

函数

您可以将 `fieldformat` 命令与一系列函数结合使用。有关使用函数的一般信息，请参阅“评估函数”。

以下表格按照函数类型列出支持的函数。使用表格中的链接以了解有关每个函数的更多信息，并查看示例。

函数类型	支持的函数和语法		
对比和条件函数	<code>case(X, "Y", ...)</code> <code>cidrmatch("X", Y)</code> <code>coalesce(X, ...)</code> <code>false()</code> <code>if(X, Y, Z)</code>	<code>in(VALUE-LIST)</code> <code>like(TEXT, PATTERN)</code> <code>match(SUBJECT, "REGEX")</code> <code>null()</code>	<code>nullif(X, Y)</code> <code>searchmatch(X)</code> <code>true()</code> <code>validate(X, Y, ...)</code>
转换函数	<code>printf("format", arguments)</code>	<code>tonumber(NUMSTR, BASE)</code>	<code>tostring(X, Y)</code>
加密函数	<code>md5(X)</code> <code>sha1(X)</code>	<code>sha256(X)</code>	<code>sha512(X)</code>
日期和时间函数	<code>now()</code> <code>relative_time(X, Y)</code>	<code>strftIME(X, Y)</code> <code>strptime(X, Y)</code>	<code>time()</code>
信息函数	<code>isbool(X)</code> <code>isint(X)</code> <code>isnotnull(X)</code>	<code>isnull(X)</code> <code>isnum(X)</code>	<code>isstr(X)</code> <code>typeof(X)</code>
数学函数	<code>abs(X)</code> <code>ceiling(X)</code> <code>exact(X)</code> <code>exp(X)</code>	<code>floor(X)</code> <code>ln(X)</code> <code>log(X, Y)</code> <code>pi()</code>	<code>pow(X, Y)</code> <code>round(X, Y)</code> <code>sigfig(X)</code> <code>sqrt(X)</code>
多值 eval 函数	<code>commands(X)</code> <code>mvappend(X, ...)</code> <code>mvcount(MVFIELD)</code> <code>mvdedup(X)</code>	<code>mvfilter(X)</code> <code>mvfind(MVFIELD, "REGEX")</code> <code>mvindex(MVFIELD, STARTINDEX, ENDINDEX)</code> <code>mvjoin(MVFIELD, STR)</code>	<code>mvrange(X, Y, Z)</code> <code>mvsort(X)</code> <code>mvzip(X, Y, "Z")</code>
统计 eval 函数	<code>max(X, ...)</code>	<code>min(X, ...)</code>	<code>random()</code>
文本函数	<code>len(X)</code> <code>lower(X)</code> <code>ltrim(X, Y)</code> <code>replace(X, Y, Z)</code>	<code>rtrim(X, Y)</code> <code>spath(X, Y)</code> <code>split(X, "Y")</code> <code>substr(X, Y, Z)</code>	<code>trim(X, Y)</code> <code>upper(X)</code> <code>urldecode(X)</code>
三角函数和双曲函数	<code>acos(X)</code> <code>acosh(X)</code> <code>asin(X)</code> <code>asinh(X)</code> <code>atan(X)</code>	<code>atan2(X, Y)</code> <code>atanh(X)</code> <code>cos(X)</code> <code>cosh(X)</code> <code>hypot(X, Y)</code>	<code>sin(X)</code> <code>sinh(X)</code> <code>tan(X)</code> <code>tanh(X)</code>

基本示例

1. 请将数字值格式设为显示逗号

此示例使用元数据命令返回主索引中的来源类型结果。

```
| metadata type=sourcetypes | table sourcetype totalCount
```

`metadata` 命令将返回很多字段。`table` 命令用于只返回 `sourcetype` 和 `totalCount` 字段。

统计选项卡中显示的结果如下所示：

sourcetype	totalCount
access_combined_wcookie	39532
cisco:esa	112421
csv	9510
secure	40088
vendor_sales	30244

使用 `fieldformat` 命令重新设置字段值的外观。将 `totalCount` 字段中的值的格式设为显示带逗号的值。

```
| metadata type=sourcetypes | table sourcetype totalCount | fieldformat totalCount=toString(totalCount, "commas")
```

统计选项卡中显示的结果如下所示：

sourcetype	totalCount
access_combined_wcookie	39,532
cisco:esa	112,421
csv	9,510
secure	40,088
vendor_sales	30,244

2. 以可读格式显示 UNIX 时间

假设 `start_time` 字段包含 UNIX 时间。设置 `start_time` 字段格式以仅显示对应 UNIX 时间的小时、分钟和秒。

```
... | fieldformat start_time = strftime(start_time, "%H:%M:%S")
```

3. 将货币符号添加到数字值

要格式化带货币符号的字段数字值，必须指定货币符号为文字，并用引号引起。将句点字符用作二进制连接运算符，后接 `toString` 函数，该函数允许您在货币值中显示逗号。

```
... | fieldformat totalSales="$".toString(totalSales,"commas")
```

延伸示例

1. 设置多值字段格式

此示例显示如何更改搜索结果外观以将数字值和日期中的逗号显示为可读格式。

首先，使用元数据命令返回主索引中的来源类型结果。

```
|metadata type=sourcetypes | table sourcetype totalCount | fieldformat totalCount=toString(totalCount, "commas")
```

```
| metadata type=sourcetypes | rename totalCount as Count firstTime as "First Event" lastTime as "Last Event" recentTime as "Last Update" | table sourcetype Count "First Event" "Last Event" "Last Update"
```

- `metadata` 命令返回字段 `firstTime`、`lastTime`、`recentTime`、`totalCount` 和 `type`。
- 除此之外，因为搜索指定 `types=sourcetypes`，还返回名为 `sourcetype` 的字段。
- `totalCount`、`firstTime`、`lastTime` 和 `recentTime` 字段重命名为 `Count`、`First Event`、`Last Event` 和 `Last Update`。
- `First Event`、`Last Event` 和 `Last Update` 字段以 UNIX 时间显示值。

统计选项卡中显示的结果如下所示：

sourcetype	Count	第一个事件	最后一个事件	上次更新
	220			

access_combined_wcookie	39532	1520904136	1524014536	1524067875
cisco:esa	112421	1521501480	1521515900	1523471156
csv	9510	1520307602	1523296313	1523392090
secure	40088	1520838901	1523949306	1524067876
vendor_sales	30244	1520904187	1524014642	1524067875

使用 `fieldformat` 命令重新设置这些字段输出的外观。Count 字段的格式设置为显示带逗号的值。First Event、Last Event 和 Last Update 字段设置为以可读时间戳显示值的格式。

```
| metadata type=sourcetypes | rename totalCount as Count firstTime as "First Event" lastTime as "Last Event" recentTime as "Last Update" | table sourcetype Count "First Event" "Last Event" "Last Update" | fieldformat Count=tostring(Count, "commas") | fieldformat "First Event"=strftime('First Event', "%c") | fieldformat "Last Event"=strftime('Last Event', "%c") | fieldformat "Last Update"=strftime('Last Update', "%c")
```

统计选项卡中显示的结果如下所示：

sourcetype	Count	第一个事件	最后一个事件	上次更新
access_combined_wcookie	39,532	3月12日星期一 18:22:16 2018	4月17日星期二 18:22:16 2018	4月18日星期三 09:11:15 2018
cisco:esa	112,421	3月19日星期一 16:18:00 2018	3月19日星期一 20:18:20 2018	4月11日星期三 11:25:56 2018
csv	9,510	3月5日星期一 19:40:02 2018	4月9日星期一 10:51:53 2018	4月10日星期二 13:28:10 2018
secure	40,088	3月12日星期一 0:15:01 2018	4月17日星期二 0:15:06 2018	4月18日星期三 9:11:16 2018
vendor_sales	30,244	3月12日星期一 18:23:07 2018	4月17日星期二 18:24:02 2018	4月18日星期三 09:11:15 2018

另请参阅

[eval, where](#)

[日期和时间格式变量](#)

字段

描述

根据字段列表的条件在搜索结果中保留或移除字段。

默认情况下，内部字段 `_raw` 和 `_time` 包含在 Splunk Web 的输出中。使用 `outputcsv` 命令将其他内部字段包含在输出中。请参阅“用法”。

语法

`fields [+|-] <wc-field-list>`

必要参数

`<wc-field-list>`

语法：`<field>, <field>, ...`

描述：要保留或移除的字段列表，各字段以逗号分隔。您可以使用星号（*）作为通配符来指定具有相似名称的字段列表。例如，如果要指定所有以“value”开头的字段，则可以使用通配符，例如 `value*`。

可选参数

+ | -
语法: + | -
描述: 如果已指定加 (+) 号, 只有 `wc-field-list` 中的字段保留在结果中。如果已指定负 (-) 号, `wc-field-list` 中的字段从结果中移除。
默认值: +

用法

`fields` 命令属于可分配的流命令。请参阅“命令类型”。

内部字段和 *Splunk Web*

内部字段名称保留前导下划线, 如 `_raw` 和 `_time`。默认情况下, 内部字段 `_raw` 和 `_time` 包含在 *Splunk Web* 中的搜索结果中。`fields` 命令无法移除这些内部字段, 除非您明确指定字段不应出现在 *Splunk Web* 的输出中。

例如, 要删除所有内部字段, 您可以指定:

```
... | fields - _*
```

要排除某个字段, 如 `_raw`, 则应指定:

```
... | fields - _raw
```

移除 `_time` 字段时请务必小心。诸如 `timechart` 和 `chart` 等统计命令无法在没有 `_time` 字段的情况下显示日期或时间信息。

在 *Splunk Web* 中显示内部字段

除了 `_raw` 和 `_time` 字段, *Splunk Web* 中不显示内部字段, 即使您在搜索中明确指定字段。例如, 以下搜索不显示结果中的 `_bkt` 字段。

```
index=_internal | head 5 | fields + _bkt | table _bkt
```

要显示结果中的内部字段, 字段必须复制或重命名到不包含前导下划线字符的字段名称。例如:

```
index=_internal | head 5 | fields + _bkt | eval bkt=_bkt | table bkt
```

内部字段和 `outputcsv` 命令

`outputcsv` 命令用于搜索时, 有自动添加到 CSV 文件的其他内部字段。添加的最常用内部字段是:

- `_raw`
- `_time`
- `_indextime`

要将内部字段从输出中排除, 请指定您想要排除的每个字段。例如:

```
... | fields - _raw _indextime _sourcetype _serial | outputcsv MyTestCsvFile
```

示例

示例 1:

将 `host` 和 `ip` 字段从结果中移出

```
... | fields - host, ip
```

示例 2:

仅保留 `host` 和 `ip` 字段。删除所有内部字段。内部字段以下划线字符开头, 例如 `_time`。

```
... | fields host, ip | fields - _*
```

示例 3:

将不需要的内部字段从输出 CSV 文件中删除。要排除的字段是 `_raw`、`_indextime`、`_sourcetype`、`_subsecond` 和 `_serial`。

```
index= internal sourcetype="splunkd" | head 5 | fields - raw, _indextime, _sourcetype, _subsecond, _serial | outputcsv
```

示例 4:

仅保留字段 source、sourcetype、host 和所有以 error 开头的字段。

```
... | fields source, sourcetype, host, error*
```

另请参阅

rename, table

fieldsummary**描述**

`fieldsummary` 命令计算所有字段或事件中所含字段子集的摘要统计数据。摘要信息以结果表显示。

语法

```
fieldsummary [maxvals=<unsigned_int>] [<wc-field-list>]
```

可选参数

`maxvals`

语法: `maxvals=<unsigned_int>`

描述: 指定为每个字段返回的非重复值的最大数量。不可以为负值。设置 `maxvals = 0` 以返回每个字段的所有可用的非重复值。

默认值: 100

`wc-field-list`

语法: `<field> ...`

描述: 单个字段名称或以空格分隔的字段名称列表。您可以使用星号 (*) 作为通配符来指定具有相似名称的字段列表。例如, 如果要指定所有以 “value” 开头的字段, 则可以使用通配符, 例如 `value*`。

用法

`fieldsummary` 命令以结果表显示摘要信息。以下信息以结果表显示:

摘要字段名称	描述
<code>field</code>	事件中的字段名称。
<code>count</code>	带该字段的事件/结果数量。
<code>distinct_count</code>	字段中唯一值的数量。
<code>is_exact</code>	无论该字段是否准确。这与字段值的非重复计数相关。若字段值的数量超过 <code>maxvals</code> , <code>fieldsummary</code> 不仅停止保留所有的值, 还会停止计算精确的非重复统计结果, 改为计算一个近似值。1 意味着准确, 0 意味着不准确。
<code>max</code>	如果该字段为数字, 其值的最大值。
<code>mean</code>	如果该字段为数字, 其值的平均值。
<code>min</code>	如果该字段为数字, 其值的最小值。
<code>numeric_count</code>	字段的数字值的计数。这不包括空值。
<code>stdev</code>	如果该字段为数字, 其值的标准偏差。
<code>values</code>	字段的非重复值和每个值的计数。先按最高计数对这些值排序, 再按不同值以升序排序。

示例**1. 返回所有字段的摘要**

此示例返回之前 15 分钟 `_internal` 索引中所有字段的摘要。

```
index=_internal earliest=-15m latest=now | fieldsummary
```

在此示例中，`max`、`min` 和 `stdev` 字段中的结果格式被设为最多显示 4 个小数点。

事件	模式	统计信息 (145)	可视化						
field	计数	distinct_count	is_exact	最大值	平均值	最小值	numeric_count	stdev	values
abandoned_channels	29	1	1	0	0	0	29	0	[{"value": "0", "count": 29}]
active	29	2	1	1	0.034482758620689655	0	29	0.18569533817705186	[{"value": "0", "count": 28}, {"value": "1", "count": 1}]
active_hist_searches	32	3	1	2	0.25	0	32	0.5679618342470648	[{"value": "0", "count": 26}, {"value": "1", "count": 4}, {"value": "2", "count": 2}]
active_realtime_searches	32	2	1	1	0.0625	0	32	0.24593468841898236	[{"value": "0", "count": 30}, {"value": "1", "count": 2}]
average_kbps	87	59	1	2.0506133101100894	1.3660900344827587	0	87	0.9715716381848098	[{"value": "0", "count": 29}, {"value": "2.047379668941307", "count": 1}, {"value": "2.0474428224644496", "count": 1}, {"value": "2.047644623565664", "count": 1}, {"value": "2.04784357813175", "count": 1}, {"value": "2.047881285497503", "count": 1}]

2. 返回特定字段的摘要

此示例为 `_internal` 索引中字段名称包含 "size" 和 "count" 的字段返回摘要。搜索只返回前 15 分钟每个字段的前 10 个值。

```
index=_internal earliest=-15m latest=now | fieldsummary maxvals=10 *size* *count*
```

事件	模式	统计信息 (18)	可视化						
field	计数	distinct_count	is_exact	最大值	平均值	最小值	numeric_count	stdev	values
count	94	6	1	1000	385.51063829787233	-1	94	486.761400663541	[{"value": "1000", "count": 36}, {"value": "-1", "count": 30}, {"value": "20", "count": 13}, {"value": "0", "count": 10}, {"value": "2", "count": 3}, {"value": "1", "count": 2}]
current_queue_size	87	1	1	0	0	0	87	0	[{"value": "0", "count": 87}]
current_size	551	2	1	229	12.052631578947368	0	551	51.18145036284161	[{"value": "0", "count": 522}, {"value": "229", "count": 29}]
current_size_kb	464	1	1	0	0	0	464	0	[{"value": "0", "count": 464}]

另请参阅

`analyzefields`, `anomalies`, `anomalousvalue`, `stats`

filldown

描述

使用一个字段或一组字段的最后一个非空值替换空值。若未给定任何字段列表，`filldown` 命令将应用于所有字段。如果字段先前没有任何值，将保留空白（空值）。

语法

```
filldown <wc-field-list>
```

必要参数

```
<wc-field-list>
```

语法: `<field> ...`

描述: 字段名称列表，以空格作分隔。您可以使用星号 (*) 作为通配符来指定具有相似名称的字段列表。例如，如果要指定所有以 "value" 开头的字段，则可以使用通配符，例如 `value*`。

示例

示例 1:

为所有字段向下填充空值。

```
... | filldown
```

示例 2:

仅为 count 字段向下填充空值。

```
... | filldown count
```

示例 3:

为 count 字段及以 'score' 开头的所有字段向下填充空值。

```
... | filldown count score*
```

另请参阅

`fillnull`

fillnull

描述

使用指定值替换空值。空值是在特定结果中丢失但却在其他结果中存在的字段值。使用 `fillnull` 命令即可用字符串替换空的字段值。您可以替换一个或多个字段中的空值。您可以指定一个字符串来填充空字段值，或使用默认的字段值，即零（0）。

语法

要求的语法以**粗体**表示。

```
fillnull
[value=<string>]
[<field-list>]
```

必要参数

无。

可选参数

`field-list`

语法： `<field>...`

描述： 以空格分隔的一个或多个字段的列表。如果您指定字段列表，则该列表中的所有字段都会填入您指定的 `value`。如果您指定以前不存在的字段，则会创建该字段。如果未指定字段列表，则该 `value` 将应用于所有字段。

`值`

语法： `value=<string>`

描述： 指定一个字符串值以替换空值。如果未指定值，则默认值会应用于 `<field-list>`。

默认值： 0

用法

指定 `field-list` 时，`fillnull` 命令属于可分配的流命令。没有指定 `field-list` 时，`fillnull` 命令属于数据集处理类型。请参阅“命令类型”。

示例

1. 使用默认值填充所有空字段的值

您的搜索产生了以下搜索结果：

_time	ACCESSORIES	ARCADE	SHOOTER	SIMULATION	SPORTS	STRATEGY	TEE
2021-03-17	5	17	6	3	5	32	
2021-03-16		63	39	30	22	127	56
2021-03-15	65	94	38	42		128	60

通过将 `fillnull` 命令添加到搜索的末尾，您可以用零（0）填充所有空字段值。

```
... | fillnull
```

搜索结果将如下所示：

_time	ACCESSORIES	ARCADE	SHOOTER	SIMULATION	SPORTS	STRATEGY	TEE
2021-03-17	5	17	6	3	5	32	0
2021-03-16	0	63	39	30	22	127	56
2021-03-15	65	94	38	42	0	128	60

2. 用字符串“NULL”填充所有空字段

对于当前搜索结果，用字符串“NULL”填充所有空字段值。

```
... | fillnull value=NULL
```

使用上一示例中的搜索结果，在搜索末尾添加 `value=NULL` 会将结果更改为：

_time	ACCESSORIES	ARCADE	SHOOTER	SIMULATION	SPORTS	STRATEGY	TEE
2021-03-17	5	17	6	3	5	32	NULL
2021-03-16	NULL	63	39	30	22	127	56
2021-03-15	65	94	38	42	NULL	128	60

3. 用字符串“unknown”填充指定的字段

您的搜索产生了以下搜索结果：

_time	host	average_kbps	instantaneous_kbps	kbps
2021/02/14 12:00	danube.sample.com		1.865	3.420
2021/02/14 11:53	mekong.buttercupgames.com	0.710	0.164	1.256
2021/02/14 11:47	danube.sample.com	1.325		2.230
2021/02/14 11:42	yangtze.buttercupgames.com	2.249	0.000	2.249
2021/02/14 11:39		2.874	3.841	1.906
2021/02/14 11:33	nile.example.net	2.023	0.915	

通过将 `fillnull` 命令添加到搜索中，您可以使用字符串“unknown”填充“host”和“kbps”字段中的所有空字段值。

```
... | fillnull value=unknown host kbps
```

_time	host	average_kbps	instantaneous_kbps	kbps
2021/02/14 12:00	danube.sample.com		1.865	3.420

2021/02/14 11:53	mekong. buttercupgames. com	0. 710	0. 164	1. 256
2021/02/14 11:47	danube. sample. com	1. 325		2. 230
2021/02/14 11:42	yangtze. buttercupgames. com	2. 249	0. 000	2. 249
2021/02/14 11:39	unknown	2. 874	3. 841	1. 906
2021/02/14 11:33	nile. example. net	2. 023	0. 915	unknown

如果您指定的字段不存在，则会创建该字段并将您指定的值添加到新字段中。

例如，如果您在字段列表中指定 bytes，则会创建 bytes 字段并填入字符串“unknown”。

```
... | fillnull value=unknown host kbps bytes
```

_time	host	average_kbps	instantaneous_kbps	kbps	bytes
2021/02/14 12:00	danube. sample. com		1. 865	3. 420	unknown
2021/02/14 11:53	mekong. buttercupgames. com	0. 710	0. 164	1. 256	unknown
2021/02/14 11:47	danube. sample. com	1. 325		2. 230	unknown
2021/02/14 11:42	yangtze. buttercupgames. com	2. 249	0. 000	2. 249	unknown
2021/02/14 11:39	unknown	2. 874	3. 841	1. 906	unknown
2021/02/14 11:33	nile. example. net	2. 023	0. 915	unknown	unknown

4. 将 fillnull 命令与 timechart 命令配合使用

按主机构建 Web 事件的时间系列图，并用字符串“NULL”填充所有空字段。

```
sourcetype="web" | timechart count by host | fillnull value=NULL
```

另请参阅

filldown
streamstats

findtypes

描述

使用搜索结果并新建一个潜在事件类型列表，以生成建议的事件类型。最多分析 5000 个事件，以发现事件类型。

语法

```
findtypes max=<int> [notcovered] [useraw]
```

必要参数

max

数据类型：<int>

描述：要返回的最大事件数。

默认值：10

可选参数

notcovered

描述：若使用了该关键字，findtypes 命令仅返回尚未涵盖的事件类型。

useraw

描述：若使用了该关键字，`findtypes` 命令将使用事件 `_raw` 文本中的短语来生成事件类型。

示例

示例 1：

发现 10 个常用事件类型。

```
... | findtypes
```

示例 2：

发现 50 个常用事件类型，并为查找文本短语添加支持。

```
... | findtypes max=50 useraw
```

另请参阅

typer

folderize

描述

新建更高级别的分组，例如，用目录替换文件名。用更一般性的值替换 `attr` 属性值，这是将此 `attr` 值与来自其他结果的其他值组合在一起的结果，通过在 `sep` 分隔符值上标记 `attr` 值实现分组。

例如，`folderize` 命令可以将搜索结果进行分组，如 Splunk Web 主页上用来列出分层数据桶（如目录或类别）的搜索结果。`folderize` 命令不是列出 200 个来源，而是用分隔符（例如 `/`）将数据来源字符串分隔开，并确定只查看这些目录是否会产生所请求的结果数。

语法

```
folderize attr=<string> [sep=<string>] [size=<string>] [minfolders=<int>] [maxfolders=<int>]
```

参数

attr

语法：attr=<string>

描述：用更一般性的值替换 `attr` 属性值，这是将其与来自其他结果的其他值组合在一起的结果，通过在 `sep` 分隔符值上标记 `attr` 值实现分组。

sep

语法：sep=<string>

描述：指定当多个数据系列与一个拆分依据字段结合使用时用于构建输出字段名称的分隔符。

默认值： ::

size

语法：size=<string>

描述：提供用于文件夹大小的名称。

默认值： totalCount

minfolders

语法：minfolders=<int>

描述：设置要归组的最小文件夹数。

默认值： 2

maxfolders

语法：maxfolders=<int>

描述：设置要归组的最大文件夹数。

默认值： 20

示例

1. 根据 `URI` 将结果归组到文件夹中

思考以下搜索。

```
index=_internal | stats count(uri) by uri
```

以下图像显示使用所有时间范围的搜索运行的结果。许多结果以 /en-US/account 开头。由于部分 URI 过长，因此图像没有在最右边显示第二列。该列是由 stats 命令新建的 count(uri) 列。

新搜索

另存为 ▾ 关闭

index=_internal | stats count(uri) by uri

所有时间 ▾

✓ 251,354 个事件 (18/11/03 9:32:32.000 之前) 无事件采样 ▾

任务 ▾ || ■ ▶ ■ ▶ ■ ▶ 智能模式 ▾

事件 模式 统计信息 (6,671) 可视化

每页 20 个 ▾ / 格式 预览 ▾

uri ▾ < 上一个 1 2 3 4 5 6 7 8 ... 下一步 >

uri	count(uri)
/	34
/en-GB	20
/en-GB/	52
/en-GB/account/login	8
/en-GB/account/login?return_to=%2Fen-GB%2F	10
/en-GB/account/logout	4
/en-GB/api/manager/control	2
/en-GB/api/messages/index	16
/en-GB/app/launcher	32
/en-GB/app/launcher/home	34
/en-GB/app/search/	8
/en-GB/app/search/dashboards	4
/en-GB/app/search/job_manager	2
/en-GB/app/search/my_dashboard/edit	2
/en-GB/app/search/report?s=%2FservicesNS%2Fadmin%2Fsearch%2Fsaved%2Fsearches%2FSavedSearch2&sid=1541228509.162	2
/en-GB/app/search/report?s=%2FservicesNS%2Fnobody%2Fsearch%2Fsaved%2Fsearches%2FErrors%2520in%2520the%2520last%252024%2520hours	2

使用 folderize 命令时，您可以将 URI 值汇总成易于管理的分组。

```
index=_internal | stats count(uri) by uri | folderize size=count(uri) attr=uri sep="/"
```

下图显示分组到结果集中的 URI。

新搜索

另存为 ▾ 关闭

index=_internal | stats count(uri) by uri | folderize size=count(uri) attr=uri sep="/"

所有时间 ▾

✓ 251,573 个事件 (18/11/03 9:33:41.000 之前) 无事件采样 ▾

任务 ▾ || ■ ▶ ■ ▶ ■ ▶ 智能模式 ▾

事件 模式 统计信息 (16) 可视化

每页 20 个 ▾ / 格式 预览 ▾

uri ▾ < 上一个 count(uri) ▾ memberCount ▾ >

uri	count(uri)	memberCount
/	34	1
/en-GB	20	1
/en-GB/	52	1
/en-GB/*	4469	3639
/favicon.ico	4	1
/ja-JP/*	1207	930
/ko-KR/*	806	600
/robots.txt	3	1
/services/*	1534	113
/servicesNS/*	671	124

在此示例中，count(uri) 列是从 stats 命令返回的唯一 URI 的计数。memberCount 列显示每组中的 URI 计数。例如：事件中发现 /en-US/ URI 22 次，如 count(uri) 列中所示。folderize 命令将 URI 分成组时，/en-US/ 组中只有 1 个成员。尽管以 /services/ 开头的 URI 在事件中出现 10088 次，但 /services/* 组中只有 1648 个唯一成员。

foreach

描述

使用此命令运行流子搜索，该搜索使用模板循环访问通配字段列表中的每个字段。

语法

要求的语法以粗体表示。

```
foreach
<wc-field-list>
[fieldstr=<string>]
[matchstr=<string>]
[matchseg1=<string>]
[matchseg2=<string>]
[matchseg3=<string>]
<subsearch>
```

必要参数

<wc-field-list>

语法： <field> ...

描述：有效字段名称的列表，以空格进行分隔。您可以使用星号（*）作为通配符来指定具有相似名称的字段列表。例如，如果要指定所有以“value”开头的字段，则可以使用通配符，例如 value*。

subsearch

语法： [subsearch]

描述：包含模板的子搜索，该模板用于替换指定字段的值。子搜索可以使用以下标记：

标记	描述
<<FIELD>>	每次运行子搜索时，字段值都会替换您在 <field-list> 中指定的每个字段的整个字段名称。
<<MATCHSTR>>	与说明符中的通配符相匹配的字段名称的一部分。
<<MATCHSEG1>>	与第一个通配符相匹配的字段名称的一部分。
<<MATCHSEG2>>	与第二个通配符相匹配的字段名称的一部分。
<<MATCHSEG3>>	与第三个通配符相匹配的字段名称的一部分。

可选参数

fieldstr

语法： fieldstr=<string>

描述：用整个字段名称替换 <<FIELD>> 标记。

matchstr

语法： matchstr=<string>

描述：用与说明符中的通配符相匹配的字段名称的一部分替换 <<MATCHSTR>>。

matchseg1

语法： matchseg1=<string>

描述：用与第一个通配符相匹配的字段名称的一部分替换 <<MATCHSEG1>>。

matchseg2

语法： matchseg2=<string>

描述：用与第二个通配符相匹配的字段名称的一部分替换 <<MATCHSEG2>>。

matchseg3

语法： matchseg3=<string>

描述：用与第三个通配符相匹配的字段名称的一部分替换 <<MATCHSEG3>>。

用法

如果字段名称包含除字母数字字符之外的字符，如短划线或句号，您需要用单引号将搜索中的 eval 命令部分中的 <<FIELD>>

标记起来。

例如：以下搜索添加来自以相似名称开头的所有字段中的值。

```
... | eval total=0 | eval test_1=1 | eval test_2=2 | eval test_3=3 | foreach test* [eval total=total + '<<FIELD>>']
```

foreach 子搜索中的 <<FIELD>> 标记只是字段名称 test* 的字符串替换。Eval 表达式不会识别含非字母数字字符的字段名称，除非字段名称用单引号括起。要使 eval 表达式正常运行，<<FIELD>> 标记需要用单引号括起。

示例

1. 添加以相似名称开始的所有字段中的值

以下搜索添加以相似名称开头的所有字段中的值。您可以在自己的 Splunk 实例中运行此搜索。

```
|makeresults 1| eval total=0 | eval test1=1 | eval test2=2 | eval test3=3 | foreach test* [eval total=total + <<FIELD>>]
```

- 此搜索使用 makeresults 命令新建 1 个结果。
- 然后搜索使用 eval 命令新建有相应值的字段 total、test1、test2 和 test3。
- foreach 命令用于为每个以 "test" 开头的字段执行子搜索。每次子搜索运行时，之前的总计将添加到测试字段的值以计算新的总计。处理完所有 "test" 字段后的最终总计为 6。

以下表格显示子搜索如何在每个 "test" 字段中重复。表格显示每次子搜索运行时 "total" 字段的开始值以及根据 "test" 字段的值计算的总计数量。

子搜索重复	测试字段	总计字段开始值	测试字段值	"总计" 字段的计算
1	test1	0	1	0+1=1
2	test2	1	2	1+2=3
3	test3	3	3	3+3=6

2. 监视许可证使用情况

使用 foreach 命令监视许可证使用情况。

首先在许可证主服务器上运行以下搜索，以返回每个来源类型的日常许可证使用情况（以字节为单位）：

```
index=_internal source=*license_usage.log type!="*Summary" earliest=-30d | timechart span=1d sum(b) AS daily_bytes by st
```

使用 foreach 命令计算每个字段的日常许可证使用情况 (**gigabytes**)：

```
index=_internal source=*license_usage.log type!="*Summary" earliest=-30d | timechart span=1d sum(b) AS daily_bytes by st | foreach * [eval <<FIELD>>='<<FIELD>>' /1024/1024/1024]
```

3. 使用 <<MATCHSTR>>

向相应的 bar* 添加匹配 foo* 的每个字段，并将结果写入 new_* 字段。例如，new_X = fooX + barX。

```
... | foreach foo* [eval new_<<MATCHSTR>> = <<FIELD>> + bar<<MATCHSTR>>]
```

4.

相当于 ... | eval foo="foo" | eval bar="bar" | eval baz="baz"

```
... | foreach foo bar baz [eval <<FIELD>> = "<<FIELD>>"]
```

5.

对于 fooXbarY 字段，这相当于：... | eval fooXbarY = "Y"

```
... | foreach foo*bar* fieldstr="#field#" matchseg2="#matchseg2#" [eval #field# = "#matchseg2#"]
```

另请参阅

eval, map

format

描述

此命令由子搜索隐式使用。此命令获取子搜索的结果，把所有结果格式化为一个结果，并将此结果放入名为 `search` 的新字段中。

语法

```
format [mvsep="<mv separator>"] [maxresults=<int>] [<row prefix> "<column prefix>" "<column separator>"<column end>" "<row separator>" "<row end>"]
```

如果您想指定一个行或列选项，则必须指定所有的行和列选项。

可选参数

mvsep

语法: `mvsep="<string>"`

描述: 用于多值字段的分隔符。

默认值: 或

maxresults

语法: `maxresults=<int>`

描述: 返回的结果数量的上限。

默认值: 0 表示返回的结果数量没有限制。

<row prefix>

语法: `"<string>"`

描述: 用于 `row prefix` 的值。

默认值: 左圆括号字符 "("

<column prefix>

语法: `"<string>"`

描述: 用于 `column prefix` 的值。

默认值: 左圆括号字符 "("

<column separator>

语法: `"<string>"`

描述: 用于 `column separator` 的值。

默认值: 且

<column end>

语法: `"<string>"`

描述: 用于 `column end` 的值。

默认值: 右圆括号字符 ")"

<row separator>

语法: `"<string>"`

描述: 用于 `row separator` 的值。

默认值: 或

<row end>

语法: `"<string>"`

描述: 用于 `row end` 的值。

默认值: 右圆括号字符 ")"

用法

默认情况下，如果您未指定任何可选的行和列参数，则 `format` 命令输出默认为: `"(" " (" "AND" ")") "OR" ")"`。

指定行和列参数

指定行和列参数有几个原因：

子搜索

子搜索末尾有一个隐式 `format`，它使用列和行参数的默认值。例如，您可以通过在子搜索末尾包含 `format` 命令来指定 OR 作为列的分隔符。

将搜索导出到不同的系统

当您需要将搜索导出到另一个需要不同格式的系统时，请指定行和列参数。

示例

1. 无可选参数的示例

假设您有如下所示的结果：

source	sourcetype	host
syslog.log	syslog	my_laptop
bob-syslog.log	syslog	bobs_laptop
laura-syslog.log	syslog	lauras_laptop

以下搜索返回前 2 个结果，并根据主机、数据来源和来源类型字段新建搜索。使用默认格式设置。

```
... | head 2 | fields source, sourcetype, host | format
```

此搜索返回基于前 2 个结果中的字段值的搜索语法。语法放在一个名为 `搜索` 的新字段中。

source	sourcetype	host	搜索
			((host="my_laptop" AND source="syslog.log" AND sourcetype="syslog") OR (host="bobs_laptop" AND source="bob-syslog.log" AND sourcetype="syslog"))

2. 使用可选参数的示例

您想生成一个输出，此输出已格式化以便用于一个外部系统。

```
... | format "[" "[" "&&" "]" "||" "]"
```

采用示例 1 中的数据，则结果为：

source	sourcetype	host	搜索
			[[host="my_laptop" && source="syslog.log" && sourcetype="syslog"] [host="bobs_laptop" && source="bob-syslog.log" && sourcetype="syslog"]]

3. 多值分隔符示例

以下搜索使用 `eval` 命令创建一个名为 “foo”的字段，该字段包含一个值 `eventtype, log_level`。`makemv` 命令用于使 `foo` 字段为多值字段，并指定逗号作为各个值之间的分隔符。搜索然后只输出 `foo` 字段并格式化该字段。

```
index=_internal |head 1 |eval foo="eventtype,log_level" | makemv delim="," foo | fields foo | format mvsep="mvseparator" "{"
"[ " "AND" "]" "AND" "}"
```

该示例将产生以下输出：

foo	搜索
	[(foo="eventtype" mvseparator foo="log_level")] }

另请参阅

搜索

`from`

描述

`from` 命令从数据集（例如数据模型数据集、CSV 查找、KV 存储查找、保存的搜索或表数据集）检索数据。

设计使用 `from` 命令引用数据集的搜索。（可选）向搜索添加其他 SPL（如查找、`eval` 表达式和转换命令）。将结果另存为报表、告警或仪表板面板。如果使用 Splunk Cloud 或使用 Splunk Enterprise，并安装了 Splunk Datasets 加载项，还可以将搜索另存为表数据集。

请参阅用法一节。

语法

要求的语法以粗体表示。

```
| from  
<dataset_type>:<dataset_name> | <dataset_type> <dataset_name>
```

您可以指定一个冒号（`:`）或 `<dataset_type>` 和 `<dataset_name>` 之间的空格。

必要参数

`<dataset_type>`

语法：`<dataset_type>`

描述：数据集类型。有效值包括：`datamodel`、`lookup` 和 `savedsearch`。

`datamodel` 数据集类型可以是数据模型数据集或表数据集。使用“数据模型编辑器”新建数据模型集。如果使用 Splunk Cloud 或使用 Splunk Enterprise，并安装了 Splunk Datasets 加载项，则可以使用“表编辑器”新建表数据集。

`lookup` 数据集类型可以是 CSV 查找或 KV 存储查找。

`savedsearch` 数据集类型为保存的搜索。您可使用 `from` 将任何已保存的搜索引用为数据集。

请参阅《知识管理器手册》中的“关于数据集”。

`<dataset_name>`

语法：`<dataset_name>`

描述：要从中检索数据的数据集的名称。如果 `dataset_type` 为数据模型，则语法为 `<datamodel_name>.<dataset_name>`。如果数据集的名称包含空格，请将数据集名称包含在引号中。

示例：如果数据模型名称为 `internal_server`，且数据集名称为 `splunkdaccess`，则指定 `dataset_name` 的 `internal_server.splunkdaccess`。

在旧版本的 Splunk 软件中，使用术语“数据模型对象”。该术语已替换为“数据模型数据集”。

可选参数

无。

用法

`from` 命令属于生成命令。根据命令引用的搜索或知识对象，可以是生成报表或者生成事件的类型。请参阅“命令类型”。

生成命令使用前导管道符且应是搜索中的第一个命令。但是，可以使用 `from` 命令（位于 `append` 命令内）。

当使用 `from` 命令时，必须引用现有数据集。您可以引用“数据集”列表页列出的任意数据集（例如数据模型数据集、CSV 查找文件、CSV 查找定义和表数据集）。您还可以引用保存的搜索和 KV 存储查找定义。请参阅《知识管理器手册》中的“查看和管理数据集”。

知识对象依赖关系

当您新建知识对象（如报表、告警、仪表板面板或表数据集）时，该知识对象依赖于引用的数据集。这称为数据集扩展。对原始数据集进行更改（例如删除或添加字段）时，更改将向下传输到已从原始数据集扩展的报表、告警、仪表板面板和表。请参阅《知识管理器手册》中的“数据集扩展”。

当为数据模型禁用字段筛选时

当您使用 `from` 命令搜索数据模型的内容时，默认搜索会返回严格筛选的字段集。它只返回默认字段和在定义数据模型的约束搜索中显式标识的字段。

如果您有本地 `datamodel.conf` 文件的编辑权限，您可以将 `strict_fields=false` 设置添加到段落，针对特定的数据模型禁用字段筛选。禁用后，具有该设置的数据模型 `| from` 搜索会返回与数据模型相关的所有字段，包括从父级数据模型继承的字段、搜索时提取的字段、计算字段以及从查找衍生的字段。

示例

1. 搜索数据模型

搜索包含 REST API 调用的内部服务器日志事件的数据模型。在此示例中，`internal_server` 为数据模型名称且 `splunkdaccess` 为 `internal_server` 数据模型内的数据集。

```
| from datamodel:internal_server.splunkdaccess
```

2. 搜索查找文件

搜索包含每个国家地理属性的查找文件，例如大陆、双字母 ISO 代码和子区域。

```
| from lookup geo_attr_countries.csv
```

3. 使用查找文件检索数据

搜索 KV 存储集合的内容，该集合 `kvstorecoll` 含有一个大于 500 的 `CustID` 值和一个以字母 P 开头的 `CustName` 值。名为 `kvstorecoll_lookup` 的查找表会引用该集合。使用 `stats` 命令提供从表中接收的事件计数。

```
| from lookup:kvstorecoll_lookup | where (CustID>500) AND (CustName="P*") | stats count
```

4. 使用已保存的搜索检索数据

从保存的名为 `mysecurityquery` 的搜索检索时间戳和客户端 IP。

```
| from savedsearch:mysecurityquery | fields _time clientip ...
```

5. 指定包含空格的数据集名称

如果数据集的名称包含空格，请将数据集名称包含在引号中。

```
| from savedsearch "Top five sourcetypes"
```

另请参阅

命令

```
datamodel  
inputlookup  
inputcsv  
lookup
```

gauge

描述

使用 `gauge` 命令将搜索结果转换为可和仪表图表结合使用的格式。仪表图表是单个聚合指标（如计数或总和）的可视化。

`gauge` 命令的输出是单个数字值，存储在被称为 `x` 的字段中。您可指定要在仪表中显示的范围，或使用默认范围 0 到 100。

有关将 `gauge` 命令和仪表图表类型结合使用的更多信息，请参阅仪表板和可视化中“仪表”一节中的“使用仪表”。

语法

```
gauge <value> [<range_val1> <range_val2> ...]
```

必要参数

值

语法：`field_name | <num>`

描述：用作仪表当前值的数字字段或文本数字。如果您指定一个数字字段，`gauge` 命令会使用该字段中的第一个值作为仪表值。

可选参数

范围值

语法: <range_val1> <range_val2> ...

描述: 用作仪表中显示的整个数字范围的两个或多个数字字段或数字的列表, 字段或数字间用空格分隔。每个范围值可以是数字字段名称或文本数字。如果您指定一个字段名称, 该字段的第一个值将用作范围值。仪表的总范围从第一个

range_val 到最后一个 range_val。请参阅[用法](#)。

默认范围: 0 至 100

用法

只要搜索结果是单个值, 您即使不用 gauge 命令也可以新建仪表图表。使用 gauge 命令的优势在于您可指定一组范围值, 而不是使用默认的范围值 0 到 100。

指定范围

如果您指定范围值, 必须指定至少两个值。仪表以您指定的第一个值开始, 最后一个值结束。

如果您指定两个以上 range_val 参数, 使用中间范围值将整个范围拆分为若干子范围。各子范围会以不同的颜色显示, 产生明显的视觉区分。

将返回一些范围值作为一系列名为 y1、y2 等的字段。

如果您没有指定范围值, 则默认范围将是较低的值 0 和较高的值 100。

如果指定了单个范围值, 则忽略。

仪表颜色

使用仪表图表, 将单个数字值映射到一组颜色。这些颜色可能具有特定商业含义或商业逻辑。当该值随时间变化时, 仪表标记在此颜色范围内的位置也随之改变。

仪表图表中的颜色范围是基于您用 gauge 命令指定的范围值。您指定范围值之后, 可定义仪表表示的整个数字范围。您可在该范围内定义色带的大小。如果您想使用色带, 可以在搜索字符串中添加四个“范围值”。用以指出范围的开始和结束以及范围内色带的相对大小。

示例

1. 新建带多个范围的仪表

统计事件的数量并在仪表上用四个范围显示计数值。这四个范围分别为 0–750、750–1000、1000–1250 和 1250–1500。

首先使用此搜索生成结果表。使用过去 15 分钟时间范围运行搜索。

```
index=_internal | stats count as myCount | gauge myCount 750 1000 1250 1500
```

统计选项卡中显示的结果如下所示:

x	y1	y2	y3	y4
3321	750	1000	1250	1500

单击可视化选项卡。您可以从下面三种仪表类型中选择: 径向仪表、塞尺和标记规。下图显示基于搜索结果新建的径向仪表。



有关将 `gauge` 命令和仪表图类型结合使用的更多信息，请参阅《仪表板和可视化》手册中的“仪表”一节。

另请参阅

命令

`Eval`
`stats`

gentimes

描述

`gentimes` 命令与 `map` 命令结合使用时效果颇佳。

生成时间戳结果，其在指定为启动时间的准确时间启动。每个结果将描述以离散值指示的相邻非重叠时间范围，直到生成足够的结果来传递结束时间值。

此命令不适用于未来日期。

语法

```
| gentimes start=<timestamp> [end=<timestamp>] [increment=<increment>]
```

必要参数

start

语法: `start=<timestamp>`
描述: 指定为开始时间。

<timestamp>

语法: `MM/DD/YYYY[:HH:MM:SS] | <int>`
描述: 指示时间范围，例如: `10/1/2017` 表示 2017 年 10 月 1 日，`4/1/2017:12:34:56` 表示 2017 年 4 月 1 日 12:34:56，或 `-5` 表示五天之前。

可选参数

end

语法: `end=<timestamp>`
描述: 指定结束时间。
默认值: 午夜，早于本地的当前时间。

increment

语法: `increment=<int>(s | m | h | d)`
描述: 指定从开始时间到结束时间的递增量。支持的增量是秒、分钟、小时和天。
默认值: `1d`

用法

`gentimes` 命令属于事件生成命令。请参阅“命令类型”。

生成命令使用前导管道符且应是搜索中的第一个命令。

`gentimes` 命令将返回四个字段。

字段	描述
<code>starttime</code>	开始时间范围用 UNIX 时间表示。
<code>starthuman</code>	用户可读时间范围格式为 DDD MMM DD HH:MM:SS YYYY。例如: Sun Apr 1 00:00:00 2018。
<code>endtime</code>	结束时间范围用 UNIX 时间表示。
<code>endhuman</code>	用户可读时间范围格式为 DDD MMM DD HH:MM:SS YYYY。例如: Fri Apr 13 23:59:59 2018。

示例

1. 通过指定日期生成以天为单位的时间范围。

生成从 2018 年 4 月 1 日起至 4 月 5 日的以天为单位的时间范围。此搜索将生成四个包含一天时间的间隔，与 2018 年 4 月 1、2、3 和 4 日的日历日对齐。

```
| gentimes start=4/1/18 end=4/5/18
```

统计选项卡中显示的结果如下所示：

starttime	starthuman	endtime	endhuman
1522566000	Sun Apr 1 00:00:00 2018	1522652399	Sun Apr 1 23:59:59 2018
1522652400	Mon Apr 2 00:00:00 2018	1522738799	Mon Apr 2 23:59:59 2018
1522738800	Tue Apr 3 00:00:00 2018	1522825199	Tue Apr 3 23:59:59 2018
1522825200	Wed Apr 4 00:00:00 2018	1522911599	Wed Apr 4 23:59:59 2018

2. 通过指定相对时间生成以天为单位的时间范围

生成从 30 天前至 27 天前的以天为单位的时间范围。

```
| gentimes start=-30 end=-27
```

3. 生成以小时为单位的时间范围

生成从 2017 年 12 月 1 日起至 12 月 5 日以小时为单位的时间范围。

```
| gentimes start=12/1/17 end=12/5/17 increment=1h
```

4. 仅指定开始日期生成时间范围

生成从 9 月 25 日起至今的以天为单位的时间范围。

```
| gentimes start=9/25/17
```

5. 生成以周为单位的时间范围

尽管不支持周增量，您仍旧可以通过指定 `increment=7d` 生成以周为单位的增量。

此示例生成以周为单位的时间范围，从 2017 年 12 月 1 日到 2018 年 4 月 30 日。

```
| gentimes start=12/1/17 end=4/30/18 increment=7d
```

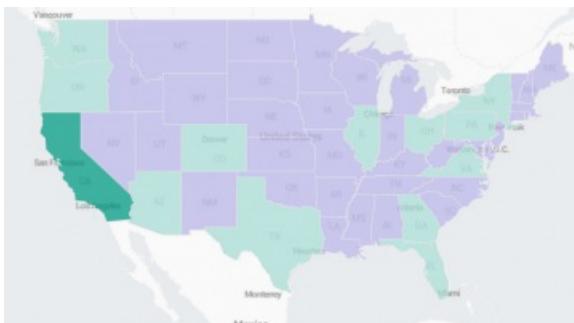
另请参阅

`makeresults`, `map`

geom

描述

geom 命令把一个名为 `geom` 的字段添加到每个事件。该字段包含 JSON 中多边形几何体的地理数据结构。这些地理数据结构用于新建地区分布图可视化。



有关地区分布图的更多信息，请参阅《仪表板和可视化》手册中的“映射数据”。

语法

```
geom [<featureCollection>] [allFeatures=<boolean>] [featureIdField=<string>] [gen=<double>] [min_x=<double>]  
[min_y=<double>] [max_x=<double>] [max_y=<double>]
```

必要参数

无。

可选参数

featureCollection

语法: <geo_lookup>

描述: 指定您要用的地理查找文件。默认情况下，Splunk 软件中包含两个地理查找文件：`geo_us_states` 和 `geo_countries`。您可以在 KMZ 或 KML 文件中安装自己的地理查找。更多信息请参阅“用法”。

allFeatures

语法: allFeatures=<bool>

描述: 指定输出包括特征集合中的每个几何特征。如果形状没有值，使用此参数时任何聚合字段如 `average` 或 `count` 会显示零。使用此参数时将为搜索结果中不存在的每个功能附加其他行。请参阅“示例”。

默认值: `false`

featureIdField

语法: featureIdField=<field>

描述: 若事件中一个命名为 "featureId" 以外名称的字段包含 `featureId`，请使用该选项指定字段名称。

gen

语法: gen=<double>

描述: 指定泛化，以数据单位为单位。例如，`gen=0.1` 通过在带 0.1 度参数的多边形上运行 Douglas-Peucker 算法来泛化几何体（或减少几何体大小）。

默认值: 0.1

min_x

语法: min_x=<double>

描述: 几何形状边框左下角的 X 轴。Y 轴的范围为 -180 到 180。更多信息请参阅“用法”。

默认值: -180

min_y

语法: min_y=<double>

描述: 几何形状边框左下角的 Y 轴。Y 轴的范围为 -90 到 90。

默认值: -90

max_x

语法: max_x=<double>

描述: 几何形状边框右上角的 X 轴。X 轴的范围为 -180 到 180。

默认值: 180

max_y

语法: max_y=<double>

描述: 几何形状边框右上角的 Y 轴。范围为 -90 到 90。

默认值: 90

用法

指定查找

要在 Splunk Enterprise 中使用自己的查找文件，您可以在 Splunk Web 中定义查找或编辑 transforms.conf 文件。Splunk Cloud 订户必须使用 Splunk Web 来定义查找。

在 Splunk Web 中定义地理空间查找

- 要在 Splunk Web 中新建地理空间查找，请在设置菜单中使用查找选项。您必须添加查找文件，新建查找定义并设置查找自动运作。请参阅知识管理器手册中的“在 Splunk Web 中定义地理空间查找”。

在 transforms.conf 中配置地理空间查找

- 如果文件已经不存在，则请编辑 %SPLUNK_HOME%\etc\system\local\transforms.conf 文件，或新建名为 transforms.conf 的文件（在 %SPLUNK_HOME%\etc\system\local 目录中）。请参阅《管理员手册》中的“如何编辑配置文件”。
- 为 featureCollection 参数在 transforms.conf 文件中指定查找段落的名称。
- 在该段落中设置 external_type=geo。请参阅《知识管理器手册》中的“配置地理空间查找”。

指定无可选参数

若未指定任何参数，geom 命令将在事件中查找一个名为 featureCollection 的字段和一个名为 featureIdField 的字段。这些字段显示于 geoindex 查找的默认输出。

剪辑几何体

min_x、min_y、max_x 和 max_y 参数用于剪辑几何体。使用这些参数定义几何形状的边框。您可以指定最小矩形角 (min_x, min_y) 和最大矩形角 (max_x, max_y)。通过指定 X 轴和 Y 轴，只会返回坐标轴内的数据。

测试查找文件

您可以使用 inputlookup 命令验证地图上的几何特征是否正确。语法是 | inputlookup <your_lookup>。

例如，要验证内置 geo_us_states 查找中的集合特征是否正确呈现在地区分布图中：

- 运行以下搜索：

```
| inputlookup geo_us_states
```

- 在可视化选项卡中，更改为地区分布图。
- 放大以查看几何特征。在此示例中，是美国的州。

测试几何特征

您可以新建任意结果以测试几何特征。

要显示输出如何出现 allFeatures 参数，以下搜索将新建一组简单的字段和值。

```
| stats count | eval featureId="California" | eval count=10000 | geom geo_us_states allFeatures=true
```

- 搜索将使用 stats 命令指定 count 字段。在 count 字段中新建一个值为零 (0) 的单一结果。
- eval 命令用于将带有加利福尼亚值的 featureId 字段添加到结果。
- 另一个 eval 命令用于为 count 字段指定值 10000。您现在有一个单一结果，其中包含两个字段 count 和 featureId。
- 添加 geom 命令之后，添加另外两个字段 featureCollection and geom。

下图显示的是统计选项卡中显示的搜索结果。

新搜索

另存为 ▾ 关闭

| stats count | eval featureId="California" | eval count=10000 | geom geo_us_states allFeatures=true 过去 24 小时 ▾

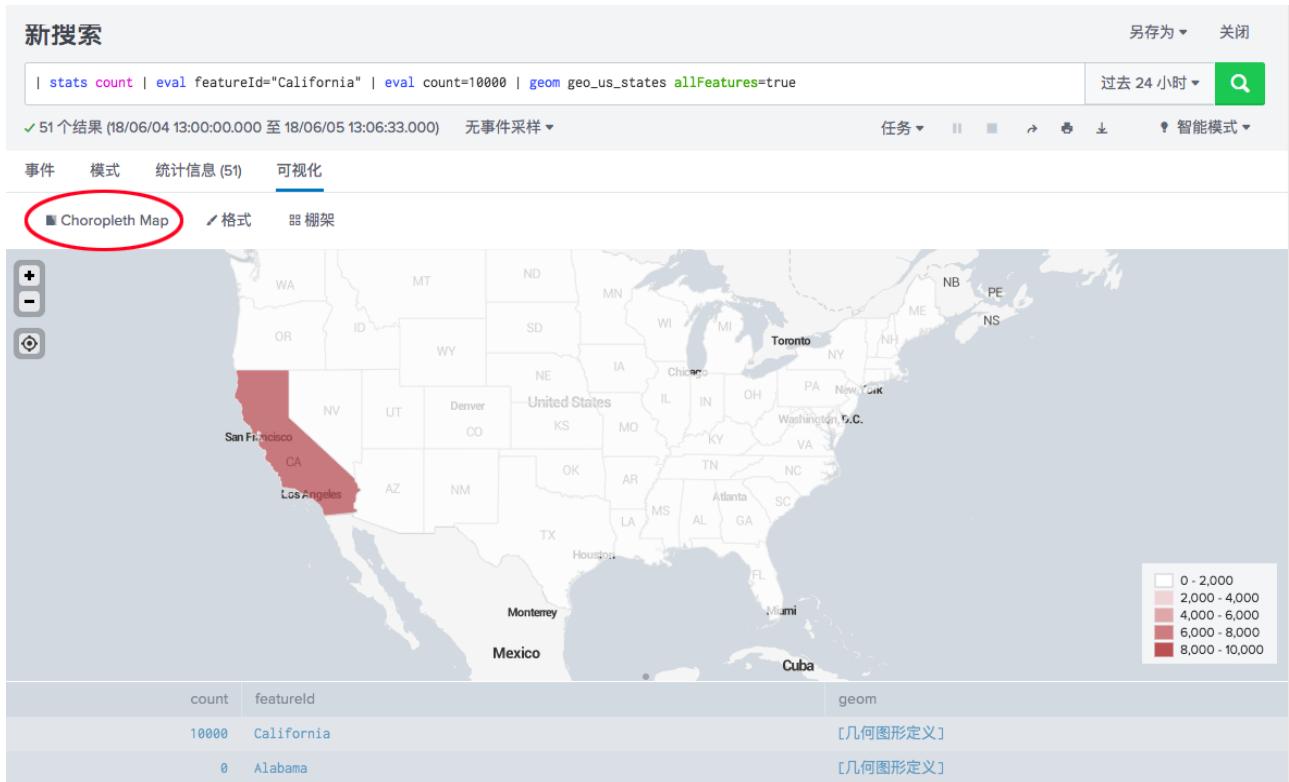
51 个结果 (18/06/04 13:00:00.000 至 18/06/05 13:06:33.000) 无事件采样 ▾ 任务 ▾ || ⌂ ⌄ ⌅ ⌆ ⌈ ⌉ ⌋ 智能模式 ▾

事件 模式 统计信息 (51) 可视化

每页 20 个 ▾ 格式 预览 ▾ < 上一个 1 2 3 下一步 >

count	featureCollection	featureId	geom
10000	geo_us_states	California	{"type": "MultiPolygon", "coordinates": "[[[-118.60337829589844, 33.47809982299805], [-118.60337829589844, 33.47809982299805]], [[[-118.60553741455078, 33.03099822998047], [-118.60553741455078, 33.03099822998047]], [[[-119.57894134521484, 33.278629302978516], [-119.57894134521484, 33.278629302978516]], [[[-119.91915130615234, 34.07727813720783], [-119.91915130615234, 34.07727813720783]], [[[-120.24739074707031, 34.00191116333008], [-120.24739074707031, 34.00191116333008]], [[[-120.46257781982422, 34.042625427246094], [-120.46257781982422, 34.042625427246094]], [[[-122.37787628173828, 37.83064651489258], [-122.37787628173828, 37.83064651489258]], [[[-122.44631958007812, 37.861045837402344], [-122.44631958007812, 37.861045837402344]], [[[-124.40859985351562, 40.443199157714844], [-124.40859985351562, 40.443199157714844]]]]]
0	geo_us_states	Alabama	{"type": "MultiPolygon", "coordinates": "[[-88.31002807617188, 30.233232498168945], [-88.31002807617188, 30.233232498168945], [[[-88.47322845458984, 31.893856048583984], [-88.47322845458984, 31.893856048583984], [[[-88.20295715332031, 35.008026123046875], [-85.60516357421875, 34.984676361083984], [-85.60516357421875, 34.984676361083984], [[[-85.00250244140625, 31.000682830810547], [-85.00250244140625, 31.000682830810547], [[[-88.02840423583984, 30.221132278442383], [-88.47322845458984, 31.893856048583984]]]]]]]

以下图像显示的是可视化选项卡中显示的搜索结果。确保地图是分级统计地图。放大此图像显示更多详细信息。



示例

1. 使用默认设置

若未提供任何参数，geom 命令将在事件中查找一个名为 **featureCollection** 的字段和一个名为 **featureId** 的字段。这些字段显示于 geospatial 查找的默认输出。

... | geom

2. 使用内置 geospatial 查找

示例使用内置 geo_us_states 查找文件，用于 featureCollection。

```
... | geom geo_us_states
```

3. 指定包含 featureId 的字段

示例使用内置 geo_us_states 查找，将 state 指定为 featureIdField。在大多数地理空间查找文件中，特征 ID 存储于名为 featureId 的文件中。当事件包含名为其他字段，而非 "featureId" 字段中的特征 ID 时，使用 featureIdField 参数。

```
... | geom geo_us_states featureIdField="state"
```

4. 显示输出中的所有几何特征

以下示例指定输出包括特征集合中的每个几何特征。如果没有值代表几何功能，零为默认值。使用 allFeatures 参数会导致地区分布图可视化提交所有形状。

```
... | geom geo_us_states allFeatures=true
```

5. 使用内置国家和地区查找

以下示例使用内置 geo_countries 查找。此搜索使用 lookup 命令指定纬度和经度字段更短的字段名称。stats 命令用于对特征 ID 进行计数并将 featureIdField 字段重命名为 country。geom 命令使用重命名的字段 country 生成分层统计地图的信息。

```
... | lookup geo_countries latitude AS lat, longitude AS long | stats count BY featureIdField AS country | geom geo_countries featureIdField="country"
```

6. 为几何形状指定边界框

此示例使用 geom 命令属性，这样您可以通过指定边界框剪辑几何体。

```
... | geom geo_us_states featureIdField="state" gen=0.1 min_x=-130.5 min_y=37.6 max_x=-130.1 max_y=37.7
```

另请参阅

《仪表板和可视化》手册中的“映射数据”。

geomfilter

描述

使用 geomfilter 命令指定边框的各点，用于剪辑地区分布图。

更多有关地区分布图的信息，请参阅《仪表板和可视化》手册中的“映射数据”。

语法

```
geomfilter [min_x=<float>] [min_y=<float>] [max_x=<float>] [max_y=<float>]
```

可选参数

min_x

语法：min_x=<float>

描述：边框左下角的 X 轴，范围 [-180, 180]。

默认值：-180

min_y

语法：min_y=<float>

描述：边框左下角的 Y 轴，范围 [-90, 90]。

默认值：-90

max_x

语法：max_x=<float>

描述：边框右上角的 X 轴，范围 [-180, 180]。

默认值：180

max_y

语法: max_y=<float>
描述: 边框右上角的 Y 轴, 范围 [-90, 90]。
默认值: max_y=90

用法

geomfilter 命令接受两个指定边框以剪辑地区分布图的点。落在边框外的点会被过滤掉。

示例

示例 1: 本示例使用默认的边框, 该边框将剪辑整张地图。

```
... | geomfilter
```

示例 2: 本示例剪辑整张地图的一半。

```
... | geomfilter min_x=-90 min_y=-90 max_x=90 max_y=90
```

另请参阅

geom

geostats

描述

使用 geostats 命令生成统计信息, 以在地图上显示地理数据并汇总这些数据。

此命令生成将在世界地图上呈现且群集化成地理数据箱的统计信息。将基于事件中的纬度和经度群集事件。然后, 将评估生成群集的统计数据。使用 BY 子句可以按字段对统计信息进行分组或拆分。

为高效呈现和缩放地图, geostats 命令在一个搜索内的不同缩放级别上生成群集统计信息, 并可在其中进行可视化选择。缩放级别的数量可由 binspanlat、binspanlong 和 maxzoomlevel 选项控制。初始粒度通过 binspanlat 和 binspanlong 进行选择。在缩放的每个级别上, 数据箱的数量在两个维度上都将翻倍, 合计是每个放大级别上数据箱数量的 4 倍。

语法

要求的语法以粗体表示。

```
geostats
[ translateToxy=<bool> ]
[ latfield=<string> ]
[ longfield=<string> ]
[ globalLimit=<int> ]
[ localLimit=<int> ]
[ outputLatfield=<string> ]
[ outputLongfield=<string> ]
[ binspanlat=<float> binspanlong=<float> ]
[ maxzoomlevel=<int> ]
<stats-agg-term>...
[ <by-clause> ]
```

必要参数

stats-agg-term

语法: <stats-func> (<evalued-field> | <wc-field>) [AS <wc-field>]

描述: 统计聚合函数。请参阅“Stats 函数”选项。该函数可应用于 Eval 表达式、字段或字段组。使用 AS 子句将结果放入您已指定其名称的新字段内。可以在字段名称中使用通配符字符。更多有关 Eval 表达式的信息, 请参阅《搜索手册》中的“Eval 表达式的类型”。

可选参数

binspanlat
语法: binspanlat=<float>
描述: 纬度中的数据桶大小处于最小缩放级别。如果将 binspanlat 设置为低于默认值，则地图上的可视化可能无法呈现。
默认值: 22.5. 如果使用 binspanlat 和 binspanlong 的默认值，则生成 8×8 的网格大小。

binspanlong
语法: binspanlong=<float>
描述: 经度中的数据桶大小处于最小缩放级别。如果将 binspanlong 设置为低于 33，则地图上的可视化可能无法呈现。
默认值: 45.0. 如果使用 binspanlat 和 binspanlong 的默认值，则生成 8×8 的网格大小。

by-clause
语法: BY <field>
描述: 分组依据字段的名称。

globallimit
语法: globallimit=<int>
描述: 控制要添加到每个饼图中的已命名分类的数量。还有另一个名为 "OTHER" 的类别，其他所有拆分依据值都在该类别下分组。设置 globallimit=0 将移除所有限制并呈现所有类别。目前，分组为 "OTHER" 仅直观用于计数和加法统计数据。
默认值: 10

locallimit
语法: locallimit=<int>
描述: 指定系列筛选的限制。若设置 locallimit=N，将会根据每个系列的总和过滤前 N 个值。若 locallimit=0，则不会执行任何筛选。
默认值: 10

latfield
语法: latfield=<field>
描述: 指定来自预搜索的字段，代表用于您分析的纬度坐标。
默认值: lat

longfield
语法: longfield=<field>
描述: 指定来自预搜索的字段，代表用于您分析的经度坐标。
默认值: lon

maxzoomlevel
语法: maxzoomlevel=<int>
描述: 在四叉树中新建的最大级别数。
默认值: 9. 指定新建 10 个缩放级别，0—9。

outputlatfield
语法: outputlatfield=<string>
描述: 为您的 geostats 输出数据的纬度字段指定名称。
默认值: latitude

outputlongfield
语法: outputlongfield=<string>
描述: 为您的 geostats 输出数据的经度字段指定名称。
默认值: longitude

translatetoxy
语法: translatetoxy=<bool>
描述: 如果 true，geostats 为每个区位分级定位生成一个结果。此模式适用于在地图上呈现。如果 false，geostats 为每个区位分级定位的每个类别（或多重拆分数据集的数组）生成一个结果。实际上，这会导致数据按类别拆分。此模式不能在地图上呈现。
默认值: true

Stats 函数选项

stats-func
语法: 语法取决于您使用的函数。请参阅“用法”。
描述: 可与 geostats 命令一起使用的统计和图表函数。每次调用 geostats 命令时都可以使用一个或多个函数。

用法

若要显示地图上的信息，您必须使用 geostats 命令运行报表搜索。

如果您在使用 `geostats` 命令之前使用 `lookup` 命令，请参阅“优化查找搜索”。

支持的函数

您可以将 `geostats` 命令与一系列函数结合使用。有关使用函数的一般信息，请参阅“统计和图表函数”。

- 若需按类别排列的统计函数列表，请参阅“按类别排列的函数列表”
- 若需按字母顺序排列的统计列表，请参阅“按字母顺序排列的函数列表”

内存和 `geostats` 搜索性能

一对 `limits.conf` 设置在 `geostats` 搜索的性能及其在搜索过程中所占用的 RAM 和磁盘内存量之间取得平衡。如果 `geostats` 一直完成得都很慢，则可以调整这些设置以提高其性能，但同时会增加搜索时的内存使用率，并可能因此导致搜索失败。

有关更多信息，请参阅《搜索手册》中的“内存和 `stats` 搜索性能”。

基本示例

1. 使用默认设置并计算计数

分别按默认纬度和经度字段 “`lat`” 和 “`lon`” 群集化事件。计算事件的数量。

```
... | geostats count
```

2. 指定 `latfield` 和 `longfield` 并计算字段平均值

按 “`eventlat`” 和 “`eventlong`” 值群集化/分组事件后，为每个性别计算平均等级。

```
... | geostats latfield=eventlat longfield=eventlong avg(rating) by gender
```

延伸示例

3. 统计供应商售出的每个产品的计数，并在图上显示信息

此示例使用搜索教程中的示例数据。要在您自己的 Splunk 实例中尝试此示例，您必须下载示例数据并遵循操作以将教程数据导入 Splunk。当运行搜索时，请使用时间范围所有时间。

除此之外，此示例使用您必须下载 (`prices.csv.zip` 和 `vendors.csv.zip`) 并解压缩文件的几个查找文件。您必须为 `prices.csv` 和 `vendors.csv` 文件完成教程中“启用字段查找部分”中的步骤。教程中的步骤针对 `prices.csv` 文件。如果是 `vendors.csv` 文件，请为查找定义使用名称 `vendors_lookup`。跳过教程中使查找自动化的步骤。

本搜索运用 `stats` 命令缩小 `lookup` 和 `geostats` 命令必须要处理的事件数。

使用以下搜索计算供应商售出的每个产品的计数，并在地图上显示信息。

```
sourcetype=vendor_sales | stats count by Code VendorID | lookup prices_lookup Code OUTPUTNEW product_name | table product_name  
VendorID | lookup vendors_lookup VendorID | geostats latfield=VendorLatitude longfield=VendorLongitude count by product_name
```

- 在此示例中，`sourcetype=vendor_sales` 与包含在搜索教程示例数据中的日志文件相关。这一日志文件包含如下所示的供应商信息：

```
[10/Apr/2018:18:24:02] VendorID=5036 Code=B AcctID=6024298300471575
```

- `vendors_lookup` 用于以 `vendors.csv` 文件格式输出匹配 `vendor_sales.log` 文件中 `VendorID` 的所有字段。`vendors.csv` 文件中的字段是：`Vendor`、`VendorCity`、`VendorID`、`VendorLatitude`、`VendorLongitude`、`VendorStateProvince` 和 `VendorCountry`。
- `prices_lookup` 用于把每个事件中的“代码”字段匹配到表格中的 `product_name`。

本搜索将 CSV 文件上载并定义查找，但操作都不是自动的。

本搜索将生成一份表格，并显示在统计选项卡中：

Splunk > enterprise 应用: Search & Reporting

Administrator 消息 设置 活动 帮助 查找

搜索 数据集 报表 告警 仪表板 > Search & Reporting

新搜索

sourcetype=vendor_sales | stats count by Code VendorID | lookup prices_lookup Code OUTPUTNEW product_name | table product_name VendorID | lookup vendors_lookup VendorID | geostats latfield=VendorLatitude longfield=VendorLongitude count by product_name

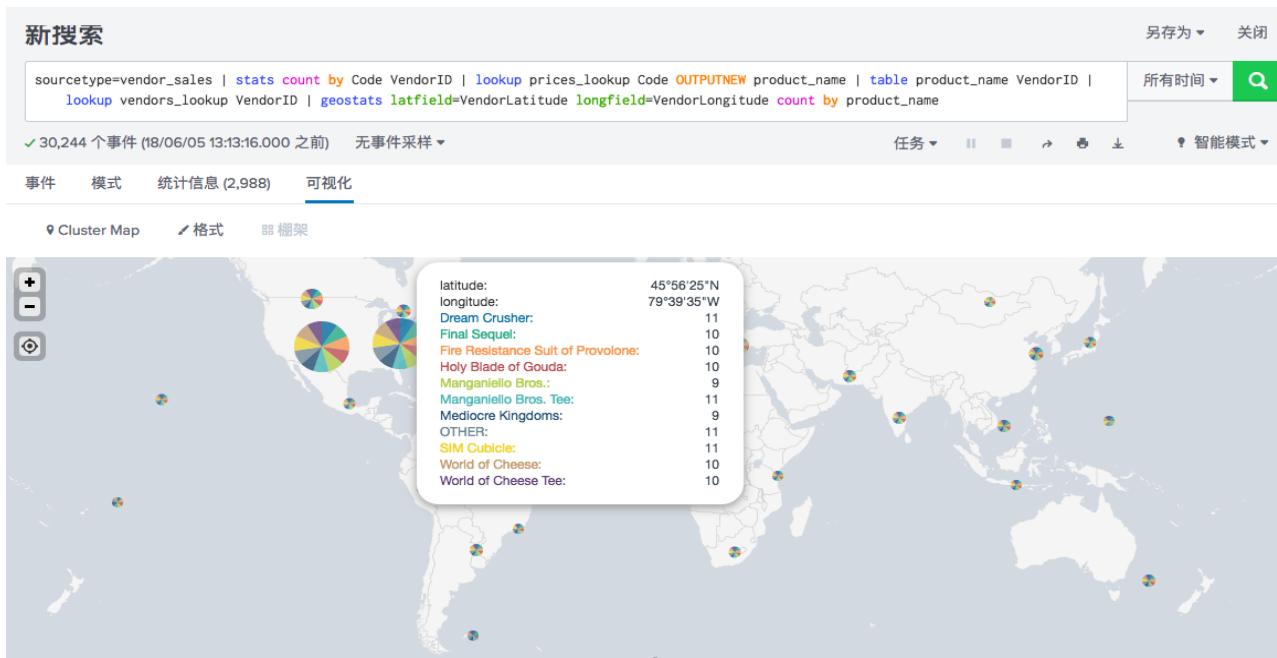
30,244 个事件 (18/06/05 13:13:16.000 之前) 无事件采样 任务 另存为 关闭

事件 模式 **统计信息 (2,988)** 可视化

每页 20 个 预览 < 上一个 1 2 3 4 5 6 7 8 ... 下一步 >

geobin	Dream Crusher	Final Sequel	Fire Resistance Suit of Provolone	Holy Blade of Gouda	Manganelli Bros.	Manganelli Bros. Tee	Mediocre Kingdoms	OTHER	SIM Cubicle	World of Cheese	World of Cheese Tee	latitude	longitude
bin_id_zl_0_y_1_x_2	1	1		1		1	1	1	1	1	1	-51.70778	-57.8
bin_id_zl_0_y_2_x_2	7	7	7	7	7	7	7	7	6	7	7	-29.44057	-56.4
bin_id_zl_0_y_2_x_3	2	2	1	2	2	2	2	2	2	1	2	-22.88780	-43.2
bin_id_zl_0_y_2_x_4	3	3	3	3	2	3	3	3	3	3	3	-30.02566	25.5
bin_id_zl_0_y_2_x_7	6	6	6	6	3	4	6	6	6	6	5	-38.13657	157.6
bin_id_zl_0_y_3_x_0	1			1	1	1	1	1	1	1		-14.33100	-170.7
bin_id_zl_0_y_3_x_2	3	3	2	3	3	3	3	3	3	3	3	-9.54565	-73.9

单击可视化选项卡。结果画在世界地图上。结果中，每个供应商都有饼图。饼状图越大，计数值越大。



在此截屏中，鼠标悬停在美国东北部地区的饼图上。弹出的信息框显示供应商纬度和经度以及供应商售出的每个产品的计数。您可以放大以在地图上查看更多详细信息。

另请参阅

命令

iplocation
stats
xseries

参考信息

《仪表板和可视化》中的“映射数据”

head

描述

按搜索顺序返回前 N 个指定结果。即历史搜索的最近 N 个事件或实时搜索的前 N 个已捕获事件。将搜索结果限制为按搜索顺序的第一个结果。

能应用的有两种类型的限制：一个以绝对数字表示的结果数量，或一个返回所有结果直至该表达式值为 false 的表达式。

语法

要求的语法以粗体表示。

```
head
[<N> | (<eval-expression>)]
[limit=<int>]
[null=<bool>]
[keeplast=<bool>]
```

必要参数

无。

若指定选项或限制，head 命令将返回前 10 个结果。

可选参数

<N>

语法：<int>
描述：要返回的结果数。
默认值：10

limit

语法：limit=<int>
描述：这是指定返回结果数的另一种方式。
默认值：10

Eval-expression

语法：<eval-compare-exp> | <eval-bool-exp>
描述：一个求值结果为布尔值的有效 <eval-expression>。此搜索会在表达式求值结果为 false 时返回结果。有关更多信息，请参阅《搜索参考》中的“评估函数”。

keeplast

语法：keeplast=<bool>
描述：您必须指定一个 eval-expression 才能使用 keeplast 参数。控制是否保留结果集中的最后一个结果。返回的最后结果是导致 eval-expression 求值为 false 或 NULL 的结果。将 keeplast 设为 true 以保留结果集汇总的最后结果。将 keeplast 设为 false 弃用最后的结果。
默认值：false

null

语法：null=<bool>
描述：您必须为 null 参数指定 <eval-expression> 才会有效果。控制如何处理求值结果为 NULL 的 <eval-expression>。例如，如果 <eval-expression> 为 (x > 10)，而且字段 x 不存在，则 <eval-expression> 将求值为 NULL 而非 true 或 false。

- 如果 null=true，则 head 命令的结果会包括输出中 <eval-expression> 求值结果为 NULL 的事件。head 命令继续处理其余事件。
- 如果 null=false，则 head 命令会将求值为 NULL 的 <eval-expression> 视为等同于求值为 false 的 <eval-expression>。head 命令停止处理事件。如果 keeplast=true，则输出中会包含 <eval-expression> 求值结果为 NULL 的事件。

默认值：false

用法

head 命令属于中央流命令。请参阅“命令类型”。

设定限制

若使用的是数字限制（如数字文本）或参数 `limit=<int>`, `head` 命令将返回前 N 个结果，其中 N 为选定的数字。同时使用数字限制和 `limit=<int>` 会导致出错。

使用 `<eval-expression>`

如果使用了 `<eval-expression>`, 那么将返回所有的初始结果直到遇到此表达式求值为 `false` 的首个结果。`<eval-expression>` 求值为 `false` 的结果将依据 `keeplast` 选项予以保留或丢弃。

若同时使用了数字限制和 `<eval-expression>`, 将应用两种限制中较小的那个。例如，以下搜索最多返回前 10 个结果，因为 `<eval-expression>` 始终为 `true`。

```
... | head limit=10 (l==1)
```

但是，此搜索不会返回任何结果，因为 `<eval-expression>` 始终为 `false`。

```
... | head limit=10 (0==1)
```

基本示例

1. 返回特定结果数

返回前 20 个结果。

```
... | head 20
```

2. 根据指定的限制返回结果

返回事件直至数据的时间跨度 ≥ 100 秒

```
... | streamstats range(_time) as timerange | head (timerange<100)
```

延伸示例

1. 使用 `keeplast` 和 `null` 参数

以下示例显示了 `<eval-expression>` 求值为 `NULL` 时的搜索结果，以及 `keeplast` 和 `null` 参数对这些结果的影响。

让我们从创建一组事件开始。`eval` 命令将 `count` 字段中的值 3 取代为 `NULL`。

```
| makeresults count=7 | streamstats count | eval count;if(count=3,null(), count)
```

结果形式如下所示：

_time	计数
2020-05-18 12:46:51	1
2020-05-18 12:46:51	2
2020-05-18 12:46:51	
2020-05-18 12:46:51	4
2020-05-18 12:46:51	5
2020-05-18 12:46:51	6
2020-05-18 12:46:51	7

当 `null` 设置为 `true` 时，`head` 命令将继续处理结果。在此示例中，只要计数小于 5，该命令就会处理结果，并忽略 `NULL` 值。由于 `keeplast=true`，停止处理的事件（计数 5）也包含在输出中。

```
| makeresults count=7 | streamstats count | eval count;if(count=3,null(), count) | head count<5 keeplast=true null=true
```

结果形式如下所示：

_time	计数
2020-05-18 12:46:51	1
2020-05-18 12:46:51	2
2020-05-18 12:46:51	
2020-05-18 12:46:51	4
2020-05-18 12:46:51	5

如果将 `null` 设置为 `false`, 则 `head` 命令在遇到 `NULL` 值时将停止处理结果。返回计数为 1 和 2 的事件。由于 `keeplast=true`, 停止处理的、包含 `NULL` 值的事件（第 3 个事件）也会包含在输出中。

```
| makeresults count=7 | streamstats count | eval count;if(count=3,null(), count) | head count<5 keeplast=true null=false
```

结果形式如下所示：

_time	计数
2020-05-18 12:46:51	1
2020-05-18 12:46:51	2
2020-05-18 12:46:51	

另请参阅

命令

```
reverse  
tail
```

highlight

描述

突出显示事件列表中的指定术语。对字符串或字符串列表进行匹配，并使它们在 **Splunk Web** 中突出显示。此匹配不区分大小写。

语法

```
highlight <string>...
```

必要参数

<string>

语法：<string> ...

描述：要在结果中突出显示的字符串列表，以空格作为分隔。您指定的列表不区分大小写。突出显示匹配字符串的任何大小写字母组合。

用法

`highlight` 命令属于可分配的流命令。请参阅“命令类型”。

您指定的字符串必须是字段值。字符串不得为字段名称。

您必须使用保留原始事件的搜索中的 `highlight` 命令并在“事件”选项卡中显示输出。您不能使用生成计算或生成的结果的 `stats` 命令使用突出显示命令。

示例

示例 1:

突出显示 "login" 和 "logout" 两个词。

```
... | highlight login,logout
```

示例 2:

突出显示词组 "Access Denied"。

```
... | highlight "access denied"
```

另请参阅

rangemap

history

描述

使用此命令查看您在当前应用的搜索历史记录。以事件集或表形式显示搜索历史。

语法

```
| history [events=<bool>]
```

必要参数

无。

可选参数

events

语法: events=<bool>

描述: 若指定 events=true, 搜索历史将以事件的形式返回。这会调用面向事件的 UI, 该 UI 允许进行方便的突出显示或字段检查。若指定 events=false, 搜索历史将以表格格式返回, 以方便聚合视图。

默认值: false

若 events=false 将返回字段。

输出字段	描述
_time	搜索开始的时间。
api_et	API 调用的最早时间, 即需要事件的最早时间。
api_lt	API 调用的最晚时间, 即需要事件的最晚时间。
event_count	如果搜索检索或生成事件, 则搜索还将返回事件计数。
exec_time	以 Unix epoch 的整秒数表示的搜索的执行时间。
is_realtime	指示搜索是为实时 (1) 还是历史 (0)。
result_count	如果搜索为转换搜索, 则为搜索结果的计数。
scan_count	在低级别上从 Splunk 索引中检索到的事件数量。
search	搜索字符串。
search_et	搜索要运行的最早时间设置。
search_lt	搜索要运行的最晚时间设置。
sid	搜索任务 ID。
splunk_server	运行搜索的计算机的主机名。
status	搜索的状态。

total_run_time	以秒表示的运行搜索所需的总时间。
----------------	------------------

用法

history 命令属于生成命令，应该是搜索中的第一个命令。生成命令使用前导管道字符。

history 命令只会从命令所运行的应用中返回搜索历史记录。

示例

以表的形式返回搜索历史

以表的形式返回搜索历史。您不必指定 events=false，因为这是默认设置。

```
| history
```

The screenshot shows the Splunk search interface with the following details:

- Search Bar:** Contains the query `| history`.
- Results Summary:** Shows 156 results from May 24, 2017, to May 25, 2017.
- Event View:** Displays a table of search events with columns: _time, api_et, api_lt, event_count, exec_time, is_realtimetime, provenance, result_count, scan_count, and search.
- Sampled Events:** The first few rows of the table are shown, illustrating the search results.

_time	api_et	api_lt	event_count	exec_time	is_realtimetime	provenance	result_count	scan_count	search
2017/05/25 14:26:07.562	0	1495718767	0	Ui:Search	156	0	0	0	history
2017/05/25 14:23:27.500	30244	1495718607	0	Ui:Search	2988	30244	search source stats count by lookup prices. OUTPUTNEW table product_ lookup vendor geostats latfield=Vend longfield=Ven count by prod	0	
2017/05/25 14:20:38.155	0	1495718438	0	Ui:Search	51	0	0	0	stats count featureId="Cal count=10000 geo_us_states
2017/05/25 14:17:58.413	39532	1495718278	0	Ui:Search	1	39532	search source chart count(ev AS GET, count(eval(me POST by host	0	

返回搜索历史作为事件

返回搜索历史作为一组事件。

```
| history events=true
```

i	时间	事件
>	17/05/25 14:27:31.315	history
>	17/05/25 14:26:27.515	history
>	17/05/25 14:26:07.562	history
>	17/05/25 14:23:27.500	search sourcetype=vendor_* stats count by Code VendorID lookup prices_lookup Code OUTPUTNEW product_name table product_name V endorID lookup vendors_lookup VendorID geostats latfield=VendorLatitude longfield=VendorLongitude count by product_name
>	17/05/25 14:20:38.155	stats count eval featureId="California" eval count=10000 geom geo_us_states allFeatures=true
>	17/05/25 14:17:58.413	search sourcetype=access_* chart count(eval(method="GET")) AS GET, count(eval(method="POST")) AS POST by host
>	17/05/25 14:16:19.640	search sourcetype=access_* timechart per_day(eval(method="GET")) AS Views_day, per_hour(eval(method="GET")) AS Views_hour, per_mi nute(eval(method="GET")) AS Views_minute, per_day(eval(action="purchase")) AS Purchases
>	17/05/25 14:14:53.378	search sourcetype=access_* chart count(eval(method="GET")) AS GET, count(eval(method="POST")) AS POST BY host
>	17/05/25 14:13:38.888	search sourcetype=access_* action=purchase stats dc(clientip) BY categoryId
>	17/05/25 14:11:36.056	search sourcetype=access_* chart count(eval(method="GET")) AS GET, count(eval(method="POST")) AS POST BY host
>	17/05/25 14:11:32.322	search sourcetype=access_* chart count(eval(method="GET")) AS GET, count(eval(method="POST")) AS POST BY host
>	17/05/25 14:10:37.875	metadata type=sourcetypes search totalCount > 0
>	17/05/25 13:50:08.440	search sourcetype=access_combined* head 5 sort _time streamstats sum(bytes) as ASimpleSumOfBytes by clientip table _time, cl ientip, bytes, ASimpleSumOfBytes

另请参阅

命令

search

iconify

描述

导致 Splunk Web 在您所指定的字段列表中为每个不同的值显示一个图标。

iconify 命令把一个名为 _icon 的字段添加到每个事件。此字段是事件的哈希值。在 Splunk Web 中，字段的每个唯一值的不同图标会显示在事件列表中。如果列出多个字段，则 UI 将为每个唯一字段值组合显示一个不同的图标。

语法

iconify <field-list>

必要参数

field-list

语法：<field>...

描述：以逗号或空格分隔的字段的列表。无法在字段列表中指定通配符字符。

用法

iconify 命令属于可分配的流命令。请参阅“命令类型”。

示例

1. 为每个 `eventtype` 显示不同图标

`... | iconify eventtype`

2. 为唯一的字段值对显示不同的图标

为唯一的 `clientip` 和 `method` 值对显示不同的图标。

`... | iconify clientip method`

从下图可以看到 Splunk Web 如何在事件列表中显示结果：

14/06/17 18:20:55.000		182.236.164.11 - - [17/Jun/2014:18:20:55] "POST /oldlink?itemId=EST-18&JSESSIONID=SD65L8FF10ADFF53101 HTTP 1.1" 408 893 "http://www.buttercupgames.com/product.screen?productId=SF-BVS-G01" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_7_4) AppleWebKit/536.5 (KHTML, like Gecko) Chrome/19.0.1084.46 Safari/536.5" 134 host = www1 itemId = EST-18 source = tutorialdata.zip.:./www1/access.log sourcetype = access_combined_wcookie
14/06/17 18:20:55.000		182.236.164.11 - - [17/Jun/2014:18:20:55] "POST /oldlink?itemId=EST-18&JSESSIONID=SD65L8FF10ADFF53101 HTTP 1.1" 408 893 "http://www.buttercupgames.com/product.screen?productId=SF-BVS-G01" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_7_4) AppleWebKit/536.5 (KHTML, like Gecko) Chrome/19.0.1084.46 Safari/536.5" 134 host = www1 itemId = EST-18 source = tutorialdata.zip.:./www1/access.log sourcetype = access_combined_wcookie
14/06/17 18:18:59.000		198.35.1.75 - - [17/Jun/2014:18:18:59] "GET /cart.do?action=addtocart&itemId=EST-13&JSESSIONID=SD10SL2FF4ADF53099 HTTP 1.1" 500 2324 "http://www.buttercupgames.com/category.screen?categoryId=NULL" "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/536.5 (KHTML, like Gecko) Chrome/19.0.1084.46 Safari/536.5" 645 host = localhost.localdomain itemId = EST-13 source = tutorialdata.zip.:./www1/access.log sourcetype = access_combined_wcookie

另请参阅

`highlight`

inputcsv

描述

对于 Splunk Enterprise 部署，从指定 `.csv` 文件加载搜索结果，此文件不会修改。文件名必须引用 `$SPLUNK_HOME/var/run/splunk/csv` 中的相对路径。如果 `dispatch=true`，则路径必须为 `$SPLUNK_HOME/var/run/splunk/dispatch/<job id>`。

如果指定文件不存在并且文件名不带有扩展名，则 Splunk 软件假设此文件有一个带 `.csv` 扩展名的文件名。

若使用 `inputcsv` 命令时出错，请确保您的 `.csv` 文件以空行结尾。

语法

要求的语法以粗体表示。

```
| inputcsv
[dispatch=<bool>]
[append=<bool>]
[strict=<bool>]
[start=<int>]
[max=<int>]
[events=<bool>]
<filename>
[WHERE <search-query>]
```

必要参数

`filename`

语法: <filename>

描述: 指定位于 `$SPLUNK_HOME/var/run/splunk/csv` 中的 `.csv` 文件的名称。

可选参数

`dispatch`

语法: `dispatch=<bool>`

描述: 当设置为 `true` 时，此参数表示文件名为 `dispatch` 目录中的 `.csv` 文件。相对路径为 `$SPLUNK_HOME/var/run/splunk/dispatch/<job id>/`。

默认值: `false`

append
语法: `append=<bool>`
描述: 指定是将来自 .csv 文件的数据附加到当前结果集中 (true) 还是替换当前结果集 (false)。
默认值: false

strict
语法: `strict=<bool>`
描述: 设置为 true 时, 如果 `inputcsv` 引发错误, 则此参数将强制使搜索彻底失败。即使错误适用于子搜索, 也会发生这种情况。设置为 false 时, 许多 `inputcsv` 错误条件会返回警告消息, 但不会导致搜索失败。某些错误条件会导致搜索失败, 即使 `strict=false` 时也会失败。
默认值: false

events
语法: `events=<bool>`
描述: 指定将 CSV 文件中的数据视为事件还是搜索结果表。默认情况下, `events=false` 会以表格形式返回数据, 且以字段名称作为列标题。该表显示于“统计信息”选项卡中。如果设置 `events=true`, 则导入的 CSV 数据必须有 `_time` 和 `_raw` 字段。数据被视为事件, 显示于“事件”选项卡中。
默认值: false

max
语法: `max=<int>`
描述: 控制将从文件中读取的最大事件数。若未指定 `max`, 则可读取的事件数量没有限制。
默认值: 1000000000 (10 亿)

start
语法: `start=<int>`
描述: 控制要读取的第一个事件以 0 为基础的偏移值。
默认值: 0

WHERE
语法: `WHERE <search-criteria>`
描述: 使用此子句通过预先筛选 CSV 文件返回的数据来改进搜索性能。支持有限的搜索查询运算符集: =、!=、<、>、<=、>=、AND、OR、NOT。可将以上运算符组合使用。同时还支持通配符字符串搜索。

用法

`inputcsv` 命令属于事件生成命令。请参阅“命令类型”。

生成命令使用前导管道符且应是搜索中的第一个命令。

添加或替换结果

若把 `append` 参数设置为 `true`, 则可以使用 `inputcsv` 命令将来自 CSV 文件的数据附加到当前的搜索结果集中。如果 `append=true`, 则可以在搜索返回一组结果之后在搜索中使用 `inputcsv` 命令。请参阅“示例”。

`append` 参数默认设置为 `false`。如果未指定 `append` 参数或将其设置为 `false`, 则 `inputcsv` 命令必须是搜索中的第一个命令。数据从指定 CSV 文件加载到搜索中。

处理大型 CSV 文件

使用 `WHERE` 子句可以缩小 `inputcsv` 文件的搜索范围。它将把 `inputcsv` 限制在较少行中, 这样做可以在您使用庞大的 CSV 文件时提高搜索效率。

分布式部署

`inputcsv` 命令和搜索头合并以及搜索头群集化不兼容。

命令将 *.csv 文件保存在 `$SPLUNK_HOME/var/run/splunk/` 目录中的本地搜索头上。*.csv 文件不会在其他搜索头中复制。

strict 的错误处理

每当 `inputcsv` 搜索遇到错误情况时, 使用 `strict` 参数使搜索失败。您可以为所有 `inputcsv` 和 `inputlookup` 搜索在系统级别上进行此设置, 具体做法为: 更改 `input_errors_fatal`, 位于 `limits.conf`

如果您使用的是 Cloud, 请向 Cloud 支持提出请求, 请示更改 `input_errors_fatal` 设置。

使用 `strict` 参数覆盖 `inputcsv` 搜索的 `input_errors_fatal` 设置。

示例

1. 加载包含特定字符串的结果

此示例加载 \$SPLUNK_HOME/var/run/splunk/csv/all.csv 文件中的搜索结果。将这些包含字符串 error 的结果保存到 \$SPLUNK_HOME/var/run/splunk/csv/error.csv 文件中。

```
| inputcsv all.csv | search error | outputcsv errors.csv
```

2. 加载特定范围内的结果

此示例从 bar 文件（若存在）或 bar.csv 文件将结果从 101 加载到 600。

```
| inputcsv start=100 max=500 bar
```

3. 指定使用运算符和表达式加载哪些结果

您可使用比较运算符和布尔表达式指定要加载哪些结果。本示例加载 CSV 文件 \$SPLUNK_HOME/var/run/splunk/csv/students.csv 中的所有事件，然后筛选出与 WHERE 子句不匹配的事件，其中 age 字段中的值大于 13，小于 19，但不是 16。搜索会返回剩余搜索结果的计数。

```
| inputcsv students.csv WHERE (age>=13 age<=19) AND NOT age=16 | stats count
```

4. 将 CSV 文件中的数据附加到搜索结果中

您可以使用 append 参数将 CSV 文件中的数据附加到一组搜索结果中。在以下示例中，合并后的数据随后会输出回同一 CSV 文件中。

```
error earliest=-d@d | inputcsv append=true all_errors.csv | outputcsv all_errors.csv
```

5. 添加多个 CSV 文件

您也可使用 append 命令和子搜索将一个 CSV 文件中的搜索结果添加到另一个 CSV 文件。以下示例使用 eval 命令向每个数据集添加一个字段，用于指示数据源自哪个 CSV 文件。

```
| inputcsv file1.csv | eval source="file1" | append [inputcsv file2.csv | eval source="file2"]
```

另请参阅

outputcsv

inputintelligence

inputintelligence命令与 Splunk Enterprise Security 结合使用。

如需此命令的相关信息，请参阅管理 *Splunk Enterprise Security* 中的“在具有输入情报的搜索中使用通用情报”。

inputlookup

描述

使用 inputlookup 命令搜索查找表的内容。查找表可以是 CSV 查找或 KV 存储查找。

语法

要求的语法以粗体表示。

```
| inputlookup
[append=<bool>]
[strict=<bool>]
[start=<int>]
[max=<int>]
[<filename> | <tablename>]
[WHERE <search-query>]
```

必要参数

您必须指定 <filename> 或 <tablename>。

<filename>

语法: <string>

描述: 查找文件的名称必须以 .csv 或 .csv.gz 结尾。如果查找不存在, 将显示警告消息 (但不会生成语法错误)。

<tablename>

语法: <string>

描述: transforms.conf 文件中的段落名称所指定的查找表名称。查找表能配置为任意查找类型 (CSV、外部或 KV 存储)。

可选参数

append

语法: append=<bool>

描述: 若设置为 true, 从查找文件返回的数据将附加到当前的结果集中, 而不是进行替换。默认为 false。

strict

语法: strict=<bool>

描述: 设置为 true 时, 如果 inputlookup 引发错误, 则此参数将强制使搜索彻底失败。即使错误适用于子搜索, 也会发生这种情况。设置为 false 时, 许多 inputlookup 错误条件会返回警告消息, 但不会导致搜索失败。某些错误条件会导致搜索失败, 即使 strict=false 时也会失败。

默认值: false

max

语法 max=<int>

描述: 指定将从文件中读取的最大事件数。默认为 1000000000。

start

语法: start=<int>

描述: 指定要读取的第一个事件以 0 为基础的偏移值。若 start=0, 则从第一个事件开始。若 start=4, 则从第五个事件开始。默认为 0。

WHERE 子句

语法: WHERE <search-query>

描述: 使用此子句通过预先筛选查找表返回的数据来改进搜索性能。支持有限的搜索查询运算符集: =、!=、<、>、<=、>=、AND、OR、NOT。可将以上运算符组合使用。同时还支持通配符字符串搜索。

用法

inputlookup 命令属于事件生成命令。请参阅“命令类型”。

生成命令使用前导管道符且应是搜索中的第一个命令。inputlookup 命令可以是搜索或子搜索中的第一个命令。

查找既可以是以 .csv 或 .csv.gz 结尾的文件名, 也可以是设置 > 查找 > 查找定义 中的查找表定义。

添加或替换结果

在子搜索中使用 inputlookup 命令时, 如果 append=true, 则来自查找文件或 KV 存储集合的数据将附加到主搜索的搜索结果中。当 append=false 时, 主搜索结果将替换为查找搜索的结果。

使用大的 CSV 查找表

可以使用 WHERE 子句缩小 inputlookup 针对查找表所执行的查询的范围。它将 inputlookup 限制在较少量的查找表格行中, 这样做可以在您使用庞大的 CSV 查找表时提高搜索效率。

测试几何查找文件

您可以使用 inputlookup 命令验证地图上的几何特征是否正确。语法是 | inputlookup <your_lookup>。

1. 例如, 要验证分级统计地图上的内置 geo_us_state 查找里的几何特征显示是否正确, 可以运行以下搜索。

```
| inputlookup geo_us_states
```

2. 在可视化选项卡中, 放大以查看集合特征。在此示例中, 是美国的州。

strict 的错误处理

每当 `inputlookup` 搜索遇到错误情况时，使用 `strict` 参数使搜索失败。您可以为所有 `inputcsv` 和 `inputlookup` 搜索在系统级别上进行此设置，具体做法为：更改 `limits.conf` 中的 `input_errors_fatal`。

如果您使用的是 Cloud，请向 Cloud 支持提出请求，请示更改 `input_errors_fatal` 设置。

使用 `strict` 参数覆盖 `inputlookup` 搜索的 `input_errors_fatal` 设置。

其他信息

有关新建查找的更多信息，请参阅知识管理器手册中的“关于查找”。

更多有关“应用键值存储”的信息，请参阅《管理员手册》中的“关于 KV 存储”。

示例

1. 读取查找表

读取 `usertogroup` 查找表，该查找表在 `transforms.conf` 文件中定义。

```
| inputlookup usertogroup
```

2. 将查找表字段附加到当前搜索结果中

使用子搜索，读取 `usertogroup` 查找表，该查找表由 `transforms.conf` 中的段落定义。将字段附加到主搜索的结果中。

```
... [| inputlookup append=t usertogroup]
```

3. 读取 CSV 文件中的查找表

搜索 `users.csv` 查找文件，该文件位于 `$SPLUNK_HOME/etc/system/lookups` 或 `$SPLUNK_HOME/etc/apps/<app_name>/lookups` 目录。

```
| inputlookup users.csv
```

4. 从 KV 存储集合中读取查找表

搜索 KV 存储集合的内容，该集合 `kvstorecoll` 含有一个大于 500 的 `CustID` 值和一个以字母 P 开头的 `CustName` 值。名为 `kvstorecoll_lookup` 的查找表会引用该集合。提供从表中接收的事件计数。

```
| inputlookup kvstorecoll_lookup where (CustID>500) AND (CustName="P*") | stats count
```

在此示例中，查找定义将 `CustID` 字段显式定义为“数字”类型。如果未显式定义字段类型，则 `where` 子句会不起作用。定义字段类型为可选操作。

5. 查看 KV 存储集合的内部关键 ID 值

示例 5：查看 KV 存储集合 `kvstorecoll` 的内部关键 ID 值，执行此操作需使用查找表 `kvstorecoll_lookup`。内部关键 ID 是集合中每个记录的唯一标识符。以下示例使用 `eval` 和 `table` 命令。

```
| inputlookup kvstorecoll_lookup | eval CustKey = _key | table CustKey, CustName, CustStreet, CustCity, CustState, CustZip
```

6. 为单个 KV 存储集合记录更新字段值

为单个 KV 存储集合记录更新字段值。本示例使用 `inputlookup`、`outputlookup` 和 `eval` 命令。内部关键 ID (`_key` 字段) 指示记录，本搜索将用新的客户姓名和客户所在城市更新记录。记录属于 KV 存储集合 `kvstorecoll`，通过查找表 `kvstorecoll_lookup` 可以访问该集合。

```
| inputlookup kvstorecoll_lookup | search _key=544948df3ec32d7a4c1d9755 | eval CustName="Claudia Garcia" | eval CustCity="San Francisco" | outputlookup kvstorecoll_lookup append=true key_field=_key
```

7. 将 CSV 文件的内容写入 KV 存储集合

把 CSV 文件的内容写入 KV 存储集合 `kvstorecoll` 中，执行此操作需使用查找表 `kvstorecoll_lookup`。CSV 文件位于 `$SPLUNK_HOME/etc/system/lookups` 或 `$SPLUNK_HOME/etc/apps/<app_name>/lookups` 目录。

```
| inputlookup customers.csv | outputlookup kvstorecoll_lookup
```

另请参阅

命令

```
inputcsv  
join  
lookup  
outputlookup
```

iplocation

描述

使用第三方数据库 IP 地址提取位置信息。此命令支持 IPv4 和 IPv6。

将在数据库中查找您在 `ip-address-fieldname` 参数中指定的 IP 地址。包含位置信息的该数据库的字段添加到每个事件。用于 `allfields` 参数的设置决定了哪些字段添加到事件。

因为对于每个 IP 地址不是所有的信息都可得，因此事件会有空字段值。

对于没有位置信息的 IP 地址，例如没有内部地址，则不会添加字段。

语法

```
iplocation [prefix=<string>] [allfields=<bool>] [lang=<string>] <ip-address-fieldname>
```

必要参数

ip-address-fieldname

语法: <field>

描述: 指定一个 IP 地址字段，如 `clientip`。

可选参数

allfields

语法: allfields=<bool>

描述: 指定是否要添加数据库中的所有字段到事件。如果设置为 `true`，添加字段 `City`、`Continent`、`Country`、`lat` (`latitude`)、`lon` (`longitude`)、`MetroCode`、`Region` 和 `Timezone`。

默认值: `false`。只有 `City`、`Country`、`lat`、`lon` 和 `Region` 字段添加到事件。

lang

语法: lang=<string>

描述: 以不同的语言呈现结果字符串。例如，使用 "`lang=es`" 代表西班牙语。语言的设置取决于使用的地理 IP 数据库。要指定多种语言，使用逗号将它们分隔。这也将按降序顺序指示优先级。指定 "`lang=code`" 以两个字母的 ISO 缩写的形式返回字段。

prefix

语法: prefix=<string>

描述: 指定添加到字段名称前缀的字符串。使用此参数，您可以添加前缀到已添加字段名称以避免与现有字段的名称冲突。例如：如果您指定 `prefix=iploc_`，添加到事件中的字段名称变成 `iploc_City`、`iploc_County`、`iploc_lat` 等。

默认值: `NULL`/空字符串

用法

`iplocation` 命令属于可分配的流命令。请参阅“命令类型”。

Splunk 软件随附 `GeoLite2-City.mmdb` 数据库文件的副本。此文件位于 `$SPLUNK_HOME/share/` 目录中。

更新 MMDB 文件

您可以将随附于 Splunk 软件的 `.mmdb` 文件替换成文件付费版本的副本或替换成文件免费版本的每月更新。

1. 从 <http://dev.maxmind.com/geoip/geoip2/geolite2/> 中下载 `GeoLite2 City` 数据库文件的二进制 `gzip` 版本。
2. 复制文件到 Splunk Enterprise 实例上的搜索头。
3. 解压缩 `GZ` 文件。
4. 停止正在运行的任何实时搜索。

5. 复制 GeoLite2-City.mmdb 文件到 \$SPLUNK_HOME/share/ 目录以覆盖其中的文件。
6. 重新开始实时搜索。

更新 Splunk 软件的影响

升级 Splunk 平台时, share 目录下的 GeoLite2-City.mmdb 文件将被替换成 Splunk 软件随附的文件版本。可以选择以不同的路径存储 MMDB 文件。

以不同的路径存储 MMDB 文件

如果您想要自己更新 GeoLite2-City.mmdb 文件, 例如使用文件付费版本的情况下, 可以将 MMDB 文件存储在不同的路径下。必须更新 Splunk 软件用于访问文件的路径。

前提条件

- 只有有着文件系统访问权限的用户 (如系统管理员) 才可以在 limits.conf 文件中指定 MMDB 文件的不同路径。
- 请参阅《管理员手册》中的“如何编辑配置文件”了解具体步骤。
- 您可以有几个具有相同名称的配置文件, 分散在默认目录、本地目录和应用目录中。请参阅《管理员手册》中“在何处放置 (或查找) 已修改的配置文件”。

不要更改或复制默认目录中的配置文件。默认目录中的文件必须保持原样并位于其原始位置。在本地目录进行更改。

如果您正在使用 Splunk Cloud, 仅通过 Splunk 版本升级来更新 MMDB 文件。如果您希望讨论或请求更新, 请提交支持工单。

步骤

1. 打开搜索应用的本地 limits.conf 文件。例如, \$SPLUNK_HOME/etc/system/local。
2. 添加 [iplocation] 段落。
3. 添加 db_path 设置, 指定 GeoLite2-City.mmdb 文件的绝对路径。db_path 设置不支持标准 Splunk 环境变量, 如 \$SPLUNK_HOME。
例如: db_path = /Applications/Splunk/mmdb/GeoLite2-City.mmdb 指定名为 mmdb 的新目录。
4. 确保 MMDB 文件的副本存储于 ../Applications/Splunk/mmdb/ 目录中。
5. 因为您正在编辑 MMDB 文件的路径, 您应重新启动 Splunk 服务器。

以不同的名字存储 MMDB 文件

另外, 您可以使用不同的名称添加已更新的 MMDB 到 share 目录, 然后在 db_path 设置中指定该名称。例如: db_path = /Applications/Splunk/share/GeoLite2-City_paid.mmdb。

MMDB 文件和分布式部署

iplocation 命令是可分配的流命令, 这表示可以在索引器上进行处理。share 目录不是知识软件包的一部分。如果您在 share 目录中更新 MMDB 文件, 更新文件不会自动发送到分布式部署中的索引器。要添加 MMDB 文件到索引器, 使用您常使用的工具将文件推送到索引器。

示例

1. 添加位置信息到 Web 访问事件

此示例使用搜索教程中的示例数据, 但应该可以与任意格式的 Apache Web 访问日志配合使用。要在您自己的 Splunk 实例中尝试此示例, 您必须下载示例数据并遵循操作以将教程数据导入 Splunk。当运行搜索时, 请使用时间范围所有时间。

添加位置信息到 Web 访问事件。默认情况下, iplocation 命令向结果添加 City、Country、lat、lon 和 Region 字段。

```
sourcetype=access_* | iplocation clientip
```

2. 搜索客户端错误并返回前 20 个结果

此示例使用搜索教程中的示例数据, 但应该可以与任意格式的 Apache Web 访问日志配合使用。要在您自己的 Splunk 实例中尝试此示例, 您必须下载示例数据并遵循操作以将教程数据导入 Splunk。当运行搜索时, 请使用时间范围所有时间。

搜索 Web 访问事件中的客户端错误, 仅返回前 20 个结果。添加位置信息并为每个客户端错误返回有 IP 地址、城市和国家地区的表格。

```
sourcetype=access_* status>=400 | head 20 | iplocation clientip | table clientip, status, City, Country
```

统计选项卡中显示的结果如下所示：

clientip	status	城市	国家
182.236.164.11	408	郑州	中国
198.35.1.75	500	普林斯顿	美国
198.35.1.75	404	普林斯顿	美国
198.35.1.75	406	普林斯顿	美国
198.35.1.75	500	普林斯顿	美国
221.204.246.72	503	太原	中国
1.192.86.205	503	埃姆斯伯里	美国
91.205.189.15	406		
216.221.226.11	505	雷德伍德城	美国
216.221.226.11	404	雷德伍德城	美国
195.2.240.99	400		俄国

3. 向 iplocation 命令添加的字段添加前缀

通过 iploc_ 为 iplocation 命令添加的字段添加前缀。向结果添加 GeoLite2-City.mmdb 数据库文件中的所有字段。

```
sourcetype = access_* | iplocation prefix=iploc_ allfields=true clientip | fields iploc_*
```

新搜索

另存为 ▾ 关闭

sourcetype = access_* | iplocation prefix=iploc_ allfields=true clientip | fields iploc_*

所有时间 ▾

49,532 个事件 (18/06/05 13:17:37.000 之前) 无事件采样 ▾

任务 ▾ || ■ ▾ ⌂ ⌃ ⌄ ⌅ ⌆ 智能模式 ▾

事件 (49,532) 模式 统计信息 可视化

每列 1 小时

设定时间线的格式 ▾ - 缩小 + 缩放到所选区域 × 取消选择

列表 ▾ / 格式 每页 20 个 ▾ < 上一个 1 2 3 4 5 6 7 8 ... 下一步 >

< 隐藏字段

感兴趣的字段

a_iploc_City 95
a_iploc_Continent 6
a_iploc_Country 42
iploc_lat 100+
iploc_lon 100+
a_iploc_MetroCode 21
a_iploc_Region 76
a_iploc_Timezone 46

+ 提取新字段

i 时间 事件

> 18/06/04 18:22:16.000 91.205.189.15 - [04/Jun/2018:18:22:16] "GET /oldlink?itemId=EST-14&JSESSIONID=SD6SL7FF7ADFF53113 HTTP 1.1" 200 1665 "http://www.buttercupgames.com/oldlink?itemId=EST-14" "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/536.5 (KHTML, like Gecko) Chrome/19.0.1084.46 Safari/536.5" 159

> 18/06/04 18:22:15.000 91.205.189.15 - [04/Jun/2018:18:22:15] "GET /category.screen?categoryId=SHOOTER&JSESSIONID=SD6SL7FF7ADFF53113 HTTP 1.1" 200 1369 "http://www.google.com" "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/536.5 (KHTML, like Gecko) Chrome/19.0.1084.46 Safari/536.5" 779

> 18/06/04 18:20:56.000 182.236.164.11 - [04/Jun/2018:18:20:56] "GET /cart.do?action=addtocart&itemId=EST-15&productId=BS-AG-G09&JSESSIONID=SD6SL8FF10ADFF53101 HTTP 1.1" 200 2252 "http://www.buttercupgames.com/oldlink?itemId=EST-15" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_7_4) AppleWebKit/536.5 (KHTML, like Gecko) Chrome/19.0.1084.46 Safari/536.5" 506

> 18/06/04 18:20:55.000 182.236.164.11 - [04/Jun/2018:18:20:55] "POST /oldlink?itemId=EST-18&JSESSIONID=SD6SL8FF10ADFF53101 HTTP 1.1" 408 893 "http://www.buttercupgames.com/product.screen?productId=SF-BVS-G01" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_7_4) AppleWebKit/536.5 (KHTML, like Gecko) Chrome/19.0.1084.46 Safari/536.5" 134

4. 使用 IP 地址生成分级统计地图

使用 iplocation 命令生成数据分级统计地图，如下所示。请参阅《仪表板和可视化》中的“使用 IP 地址生成分级统计地图”。



join

描述

使用 `join` 命令合并子搜索和主搜索的结果。每个结果集必须要有一个或多个共同的字段。您还可以使用 `selfjoin` 命令将搜索结果集与其自身合并。

如果您对 SQL 很熟悉，但对 SPL 较为陌生，请参阅“面向 SQL 用户的 Splunk SPL”。

替代命令

针对灵活性和性能，如果不需要连接语义，请考虑使用以下命令之一。这些命令使用时间和地理位置、交易、子搜索、字段查找和连接提供事件分组和相关性。

命令	使用
append	将子搜索的结果附加到当前搜索的结果中。两个结果集中的事件都会保留。 <ul style="list-style-type: none"> 仅用于历史数据。<code>append</code> 命令若用于实时搜索则不会生成正确的结果。 若您使用 <code>append</code> 合并事件，请使用 <code>stats</code> 命令以有意义的方式对事件进行分组。使用 <code>transaction</code> 命令之前不能使用 <code>append</code> 命令。
appendcols	将子搜索结果的字段附加到输入搜索结果字段中。第一个子搜索结果与第一个主结果合并，第二个子搜索结果与第二个主结果合并，依此类推。
lookup	仅当其中一个结果集或源文件为静态或极少变化时使用。例如，来自外部系统的文件，像 CSV 文件。 查找不能为子搜索。
search	在最简单的方案中，您可能仅需要使用 OR 运算符搜索来源，然后再使用 <code>stats</code> 或 <code>transaction</code> 命令对事件进行分组。
stats	按字段将事件分组并对事件执行统计函数。例如，按主机名确定事件的平均持续时间。 <ul style="list-style-type: none"> 若要使用 <code>stats</code>，该字段必须具有唯一标识符。 若要查看原始事件数据，请改用 <code>transaction</code> 命令。
transaction	下列情况请使用 <code>transaction</code> 。 <ul style="list-style-type: none"> 使用 <code>eval</code> 命令（含有诸如 <code>if</code>、<code>case</code> 或 <code>match</code> 等条件表达式）对事件进行分组。 使用回收的字段值，如 ID 或 IP 地址，来分组事件。 使用一种模式，如事件的开始或结束时间，来分组事件。 将持续时间大于某值的组分成多个小组。例如，当一个交易未以一条消息显式结束，且您想指定交易开始后的最大时间跨度时。 显示已分组事件的原始事件数据。

有关何时使用 `join` 的相关信息，请参阅《搜索手册》中的“有关事件分组和相关性”的流程图。

语法

`join [join-options...] [field-list] subsearch`

必要参数

`subsearch`

语法: `"[subsearch]"`

描述: 一种辅助搜索, 可用于指定要连接的事件源。子搜索必须用方括号括起来。子搜索结果不能超过可用内存。

`join` 命令的子搜索限制定义于 `limits.conf.spec` 文件中。该限制包含可连接的最大子搜索, 子搜索的最大搜索时间, 以及等待子搜索彻底完成的最大时间。请参阅《搜索手册》中的“子搜索”。

可选参数

`field-list`

语法: `<field>, <field>, ...`

描述: 指定要用于连接的字段。如果未指定字段, 则使用两个结果集的所有公共字段。

字段名称必须一样, 包括大小写也必须一样。无法连接 `product_id` 和 `product_ID`。要实现这种连接, 必须先修改子搜索字段的大小写, 使其名称和大小写都与主搜索中的字段一模一样。

`join-options`

语法: `type=(inner | outer | left) | usetime=<bool> | earlier=<bool> | overwrite=<bool> | max=<int>`

描述: `join` 命令的选项。使用 `outer` 或 `left` 指定左外部连接。

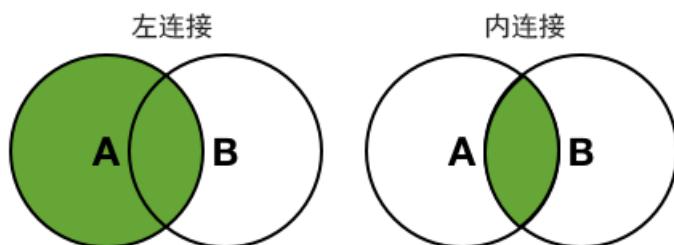
`join-options` 参数的描述

类型

语法: `type=inner | outer | left`

描述: 指示要执行的连接类型。`inner` 连接与 `left` (或 `outer`) 连接的区别在于: 二者在未匹配到任何子搜索事件的主搜索中如何对待事件。对于内部连接和左连接这两种类型, 都是对匹配的事件进行连接。`inner` 连接的结果不包含来自主搜索 (在子搜索中没有匹配项) 的事件。`left` (或 `outer`) 连接的结果包括主搜索中的所有事件; 只有子搜索内的值才具有匹配的字段值。

默认值: `inner`



连接中所包含数据的示例。

`usetime`

语法: `usetime=<bool>`

描述: 布尔值, 表示是否使用时间来限制子搜索结果中的匹配。与 `earlier` 选项一起使用, 用于将子搜索结果限制在早于或晚于主搜索结果的匹配项上。

默认值: `false`

`earlier`

语法: `earlier=<bool>`

描述: 若 `usetime=true` 且 `earlier=true`, 则主搜索结果仅针对子搜索结果中早于主搜索结果的那一部分执行匹配操作。若 `earlier=false`, 主搜索结果仅针对子搜索结果中晚于主搜索结果的那一部分执行匹配操作。同时发生 (同一秒) 的结果不能通过任一个值来消除。

默认值: `true`

`overwrite`

语法: `overwrite=<bool>`

描述: 指示来自子结果的字段是否覆盖来自主结果的字段 (如果两者字段名相同)。

默认值: `true`

`max`

语法: `max=<int>`

描述: 指定每个主搜索结果可以连接的最大子搜索结果数。若设置为 `max=0`, 则无限制。

默认值: `1`

用法

当有要加入的已定义字段集时，`join` 命令是集中流命令。请参阅“命令类型”。

若子搜索的结果相对较小，如 50,000 行或以下，请使用 `join` 命令。为了最小化此命令对性能和资源消耗的影响，Splunk 软件对子搜索设定一些默认限制。请参阅语法中的子搜索小节了解此类限制的详细信息。

一对多和多对多关系

要返回一对多、多对一或多对多关系的匹配，在您的 `join` 语法中将 `max` 参数包含在内，并将值设置为 0。默认情况下 `max=1`，意味着子搜索只返回子搜索中的首个搜索。将值设置为更高的数或设置为 0（即不受限制），将返回子搜索的多值结果。

示例

示例 1

合并主搜索和子搜索 `search vendors` 的结果。两个结果集在 `product_id` 字段处连接，两个来源共用此字段。

```
... | join product_id [search vendors]
```

示例 2

如果两种源中的字段名不匹配，您可以重命名子搜索结果集中的相应字段。主搜索中的字段为 `product_id`。子搜索中的字段为 `pid`。

注意：两个字段不但名称要一样，大小写也要一样。无法连接 `product_id` 和 `product_ID`。

```
... | join product_id [search vendors | rename pid AS product_id]
```

示例 3

默认情况下，仅返回子搜索中第一个与主搜索中的某行相匹配的行。若要返回子搜索中的所有匹配行，请使用 `max=<int>` 参数并将值设置为 0。此参数将子搜索中每个匹配行与主搜索中相应的行连接。

```
... | join product_id max=0 [search vendors]
```

示例 4

分布式管理控制台中的仪表板和告警会显示有关您的 Splunk 部署的性能信息。资源使用情况：实例仪表板包含一个表格，显示计算机、内核数量、物理内存容量、操作系统和 CPU 架构。

若要显示表格中的信息，请使用以下搜索。该搜索包含一个 `join` 命令。此搜索使用 `dmc_assets` 表格中的信息查找实例名和机器名。然后再使用 `serverName` 字段连接该信息和来自 `/services/server/info` REST 端点的信息。`/services/server/info` 前往 Splunk REST API 端点的 URI 路径，该端点提供计算机的硬件和操作系统信息。搜索的 `$splunk_server$` 部分是一个标记变量。

```
| inputlookup dmc_assets  
| search serverName = $splunk_server$  
| stats first(serverName) AS serverName, first(host) AS host, first(machine) AS machine  
| join type=left serverName  
  [ | rest splunk_server=$splunk_server$ /services/server/info  
    | fields serverName, numberOfCores, physicalMemoryMB, os_name, cpu_arch]  
| fields machine numberOfCores physicalMemoryMB os_name cpu_arch  
| rename machine AS Machine, numberOfCores AS "Number of Cores",  
  physicalMemoryMB AS "Physical Memory Capacity (MB)", os_name AS "Operating System",  
  cpu_arch AS "CPU Architecture"
```

另请参阅

`selfjoin`, `append`, `set`, `appendcols`

kmeans

描述

将事件分区为 `k` 群集，每个群集由其平均值定义。每个事件归属于平均值最接近的群集。在您指定的字段列表上执行 `k` 均值群集化。若未指定任何字段，在所有数字字段上执行群集化。同一群集内的事件将移动到一起。您可以选择显示每个事件的群集数量。

语法

`kmeans [kmeans-options...] [field-list]`

必要参数

无。

可选参数

`field-list`

语法: <field> ...

描述: 指定要用于连接的具体字段的列表，以空格作为分隔。

默认值: 若未指定任何字段，则会使用两个结果集共用的所有数字字段。并会忽略带非数字字段的事件。

`kmeans-options`

语法: <reps> | <iters> | <t> | <k> | <cnumfield> | <distype> | <showcentroid>

描述: `kmeans` 命令的选项。

`kmeans` 选项

`reps`

语法: `reps=<int>`

描述: 指定使用随机启动群集重复 `kmeans` 的次数。

默认值: 10

`iters`

语法: `maxiters=<int>`

描述: 指定收敛失败前允许的最大迭代数。

默认值: 10000

`t`

语法: `t=<num>`

描述: 指定算法收敛容差。

默认值: 0

`k`

语法: `k=<int> | <int>-<int>`

描述: 指定为标量整数值或整数范围。当作为单个数字提供的时候，选择要使用的群集数。这将产生由群集标签加注的事件。若以范围的形式表示，范围内的每个群集计数都将完成群集化，并生成结果摘要。这些结果表示群集大小，'distortion' 字段代表数据与所选群集符合的程度。值必须大于 1 而小于 `maxkvalue`（请参阅“限制”部分）。

默认值: `k=2`

`cnumfield`

语法: `cfield=<field>`

描述: 对字段进行命名，以对带有每个事件群集数的结果进行加注。

默认值: `CLUSTERNUM`

`distype`

语法: `dt= (l1 | l1norm | cityblock | cb) | (l2 | l2norm | sq | sqeuclidean) | (cos | cosine)`

描述: 指定要使用的距离指标。`l1`, `l1norm` 和 `cb` 距离指标等同于 `cityblock`。`l2`, `l2norm` 和 `sq` 等同于 `sqeclidean` 或 `sqEuclidean`。`cos` 距离指标等同于 `cosine`。

默认值: `sqeclidean`

`showcentroid`

语法: `showcentroid= true | false`

描述: 指定是否要显露搜索结果中的中心点 (`showcentroid=true`)。

默认值: `true`

用法

限制

将值收集到其中的群集的数量 — k — 不得超过 maxkvalue。在 limits.conf 文件、[kmeans] 段落中指定了 maxkvalue。maxkvalue 的默认值为 1,000。

若给定了 k 选项的范围，开始与结束群集计数之间的总距离不得超过 maxkrange。在 limits.conf 文件、[kmeans] 段落中指定了 maxkrange。maxkrange 的默认值为 100。

设计以上限制是为了避免计算工作成本急剧增多。

通过算法群集化的值的总数（通常为输入结果的数量）由 maxdatapoints 参数限制，该参数位于 [kmeans] 段落（属于 limits.conf）中。如果在运行时超过此限制，将在 Splunk Web 中显示一条警告消息。默认为 100,000,000 或一亿。该 maxdatapoints 限制用于避免耗尽内存。

示例

示例 1：根据 "date_hour" 和 "date_minute" 字段的值将搜索结果分组为 4 个群集。

```
... | kmeans k=4 date_hour date_minute
```

示例 2：根据所有数字字段的值将结果分为 2 个群集。

```
... | kmeans
```

另请参阅

anomalies, anomalousvalue, cluster, outlier,

kvform

描述

基于一个描述如何提取值的表单模板从事件中提取键/值对。

语法

```
kvform [form=<string>] [field=<field>]
```

可选参数

form

语法：form=<string>

描述：指定位于 \$SPLUNK_HOME/etc/apps/*/forms/ 目录中的 .form 文件。

field

语法：field=<field_name>

描述：使用字段名称查找对应该字段名称的字段值的 .form 文件。例如：Splunk 部署使用 splunkd 和 mongod 的来源类型。如果您指定 field=sourcetype，kvform 命令查找在 \$SPLUNK_HOME/etc/apps/*/forms/ 目录下的 splunkd.form 和 mongod.form。

默认值：sourcetype

用法

在您使用 kvform 命令之前，您必须：

- 在合适的应用程序目录下新建 forms 目录。例如，\$SPLUNK_HOME/etc/apps/<app_name>/forms。
- 新建 .form 文件并添加文件到 forms 目录。

如果您有 Splunk Cloud 并要从文件进行安装，请提交支持工单。

.form 文件的格式

.form 文件实质上是表单所有静态部分的文本文件。该文件可以包含对正则表达式（其类型可在 transforms.conf 中找到）的指定引用。

以下是 .form 文件的一个示例：

```
Students Name: [[string:student_name]]
```

```
Age: [[int:age]] Zip: [[int:zip]]
```

指定表单

如果已指定 `form` 参数，`kvform` 命令使用在 Splunk 配置 `forms` 目录中找到的 `<form_name>.form` 文件。例如：如果 `form=sales_order`，`kvform` 命令查找所有应用的 `$SPLUNK_HOME/etc/apps/<app_name>/forms` 目录下的 `sales_order.form` 文件。所有的处理事件匹配表单，尝试提取值。

指定字段

如果您指定 `field` 参数，`kvform` 命令查找 `forms` 目录下的对应该字段值的表单。例如：如果您指定 `field=error_code`，事件有字段值 `error_code=404`，命令查找 `$SPLUNK_HOME/etc/apps/<app_name>/forms` 目录下名为 `404.form` 的表单。

默认值

如果没有指定 `form` 或 `field` 参数，`kvform` 命令使用 `field` 参数的默认值，也就是 `sourcetype`。`kvform` 命令查找 `<sourcetype_value>.form` 文件以提取值。

示例

1. 使用特定表单提取值

使用特定表单提取值。

```
... | kvform form=sales_order
```

2. 使用字段名称提取值

指定 `field=sourcetype` 从像 `splunkd.form` 和 `mongod.form` 这样的表单中提取值。如果有来源类型的表单，从该表单中提取值。如果来源类型是 `access_combined`，但没有 `access_combined.form` 文件，该来源类型被忽略。

```
... | kvform field=sourcetype
```

3. 使用事件类型字段提取值

```
... | kvform field=eventtype
```

另请参阅

`extract`, `multikv`, `rex`, `xmalkv`

loadjob

描述

加载先前完成的搜索任务的事件或结果。要加载的项目按搜索任务 `id <sid>` 或已计划的搜索名称以及当前搜索时间范围来识别。如果提供保存的搜索的名称并且在该范围内找到多个项目，将加载最新的项目。

无法在即席或实时搜索中运行 `loadjob` 命令。

语法

要求的语法以粗体表示。

```
| loadjob  
(<sid> | <savedsearch>)  
[<result-event>]  
[<delegate>]  
[<artifact_offset>]  
[<ignore_running>]
```

必要参数

您必须指定 `sid` 或 `savedsearch`。

sid
语法: <string>
描述: 需要加载项目的任务的搜索 ID, 例如: 1233886270.2。您可以通过任务查看器或 addinfo 命令找到 sid。

savedsearch
语法: savedsearch="<user-string>:<app-string>:<search-name-string>"
描述: 需要加载项目的保存的搜索的唯一标识符。由三元组 {用户、应用程序、保存的搜索名称} 唯一标识的保存的搜索, 例如: savedsearch="admin:search:my Saved Search" 没有可以指定通配符或全部匹配行为的方法。必须提供三元组的所有部分。

可选参数

result-event
语法: events=<bool>
描述: events=true 加载事件, events=false 则加载结果。
默认值: false

delegate
语法: job_delegate=<string>
描述: 指定保存的搜索时, 此选项将选择由给定用户启动的任务。计划任务将通过委派“计划程序”来运行。嵌入仪表板的搜索将依据保存的搜索的 dispatchAs 参数 (通常是搜索所有者) 来运行。
默认值: scheduler

artifact_offset
语法: artifact_offset=<int>
描述: 选择最近匹配的搜索项目以外的一个搜索项目。例如, 若 artifact_offset=1, 将使用第二个最近的项目。若 artifact_offset=2, 则使用第三个最近的项目。若 artifact_offset=0, 则选择最近的项目。如果使用一个值选择过去所有可得项目, 则会出错。
默认值: 0

ignore_running
语法: ignore_running=<bool>
描述: 跳过搜索仍在运行的项目。
默认值: true

用法

loadjob 命令属于事件生成命令。请参阅“命令类型”。

生成命令使用前导管道符且应是搜索中的第一个命令。

loadjob 命令可用于多种目的, 但其中最有用的是运行一个相当昂贵的搜索来计算统计信息。您可以使用 loadjob 搜索显示这些统计信息, 以进一步就图表和显示执行聚合、分类、字段选择及其他操作。

完成一项搜索任务并缓存好结果后, 您可以使用该命令访问或加载结果。

搜索头群集

搜索头群集仅在保存的计划搜索上才能运行 loadjob 命令。搜索头群集在其复制的结果或项目上运行搜索。

有关项目复制的更多信息, 请参阅《分布式搜索》手册中的“搜索头群集化架构”。

示例

1. 加载保存的搜索的结果

针对由用户 admin 所有的 'search' 应用程序中的保存的搜索 MySavedSearch, 加载此搜索最近一次按计划运行后生成的结果。

```
| loadjob savedsearch="admin:search:MySavedSearch"
```

2. 加载特定搜索任务的结果

加载由 id=1233886270.2 的搜索任务生成的事件。

```
| loadjob 1233886270.2 events=true
```

另请参阅

命令

```
addinfo  
inputcsv  
savedsearch
```

相关信息

管理搜索任务

localize

描述

localize 命令生成代表时间上连续的事件区域的列表的结果。事件区域是最多按 maxpause 时间值分隔开的连续事件的期限。可使用 timeafter 和 timebefore 参数扩大找到的区域。

localize 命令发现的地区将提供给 map 命令。map 命令针对每次迭代都使用不同的区域。

语法

```
localize [<maxpause>] [<timeafter>] [<timebefore>]
```

可选参数

maxpause

语法: maxpause=<int>(s|m|h|d)

描述: 指定连续时间区域中的两个连续事件之间的最长时间（含端值）。

默认值: 1m

timeafter

语法: timeafter=<int>(s|m|h|d)

描述: 指定与输出结束时间字段相加的时间量（即向前延长时间区域）。

默认值: 30s

timebefore

语法: timebefore=<int>(s|m|h|d)

描述: 指定从输出起始时间字段减去的时间量（即向后延长时间区域）。

默认值: 30s

用法

展开事件范围

您可以在最后一个事件之后或在该区域中的第一个事件之前展开事件范围。这些扩展可任意进行，如果其值大于 maxpause，则可能导致区域重叠。

事件区域顺序

区域按搜索顺序返回，或历史搜索中按时间降序返回以及实时搜索中按数据到达顺序返回。每个区域的时间是初始预扩展的启动时间。

本地化命令返回的其他信息

localize 命令也会报告：

- 范围中的事件数
- 范围持续时间（以秒为单位）
- 区域密度定义为 number of events in range 除以 <范围持续时间 - events per second>

示例

1. 在每个先前结果的时间范围内搜索术语 "failure"

```
... | localize maxpause=5m | map search="search failure starttime=$starttime$ endtime=$endtime$"
```

2: 查找发生“错误”的区域

搜索 "error" 并调用 `localize` 命令可以查找发生错误的区域，并将每个找到的区域传递给 `map` 命令中的搜索。每次迭代都使用特定的时间范围来查找潜在的交易。

```
error | localize | map search="search starttimeu::$starttime$ endtimeu::$endtime$ | transaction uid,qid maxspan=1h"
```

另请参阅

`map`, `transaction`

localop

描述

阻止在远程对等方执行后续命令。反之，指示搜索在本地运行后续命令。

`localop` 命令强制后续命令成为 `mapreduce` 流程简化步骤的一部分。

语法

```
localop
```

示例

示例 1:

在这种情况下，`iplocation` 命令永远不会在远程对等节点上运行。远程对等节点发来的、所有来自对术语 FOO 和 BAR 的初始搜索的事件都将被转发给搜索头。搜索头即为运行 `iplocation` 命令的地方。

```
FOO BAR | localop | iplocation clientip
```

lookup

描述

使用 `lookup` 命令调用字段值查找。

有关您可以定义的查找类型的信息，请查看[知识管理器手册](#)中的“关于查找”。

语法

要求的语法以**粗体**表示。

```
lookup
[local=<bool>]
[update=<bool>]
<lookup-table-name>
( <lookup-field> [AS <event-field>] )...
[ OUTPUT | OUTPUTNEW (<lookup-destfield> [AS <event-destfield>] )... ]
```

注意：查找命令可以接受多个查找与事件 `fields` 以及 `destfields`。例如：

```
... | lookup <lookup-table-name><lookup-field1> AS <event-field1>, <lookup-field2> AS <event-field2> OUTPUTNEW <lookup-destfield1> AS <event-destfield1>, <lookup-destfield2> AS <event-destfield2>
```

必要参数

<**lookup-table-name**>

语法：<string>

描述：可以是要用作查找的 CSV 文件的名称，也可以是 `transforms.conf` 文件中指定查找表文件位置的段落的名称。

可选参数

`local`

语法：`local=<bool>`

描述：若 `local=true`，则强制在搜索头而非远程对等节点上运行查找。

默认值: false

update

语法: update=<bool>

描述: 如果在搜索运行期间修改了磁盘上的查找表，则实时搜索将不会自动反映更新。若要自动反映更新，请指定 update=true。此操作不适用于非实时搜索。这意味着 local=true。

默认值: false

<lookup-field>

语法: <string>

描述: 是指要与事件进行匹配的查找表中的字段。您可以指定多个 <lookup-field> 值。

<event-field>

语法: <string>

描述: 是指从事件中获取值与查找表匹配的字段。您可以指定多个 <event-field> 值。

默认值: <lookup-field> 的值。

<lookup-destfield>

语法: <string>

描述: 是指要复制到事件的查找表中的字段。您可以指定多个 <lookup-destfield> 值。

<event-destfield>

语法: <string>

描述: 所有事件中的一个字段。您可以指定多个 <event-destfield> 值。

默认值: <lookup-destfield> 参数的值。

用法

当默认设置 local=false 时，lookup 命令属于可分配的流命令。请参阅“命令类型”。

使用 lookup 命令时，若未指定 OUTPUT 或 OUTPUTNEW 子句，查找表中的所有字段（非匹配字段）都将用作输出字段。若已指定 OUTPUT 子句，输出查找字段将覆盖现有字段。如果已指定 OUTPUTNEW 子句，则不会为已包含输出字段的事件执行查找。

优化查找搜索

若您将同一管道中的 lookup 命令用作转换命令，且可在转换命令后保留您要查找的字段，则请在转换命令后执行查找。例如，您可以运行以下命令：

```
sourcetype=access_* | stats count by status | lookup status_desc status OUTPUT description
```

而不是：

```
sourcetype=access_* | lookup status_desc status OUTPUT description | stats count by description
```

第一个搜索中的查找会更快，因为它只需与 stats 命令的结果进行匹配，而不必与所有 Web 访问事件进行匹配。

基本示例

1. 查找用户并返回用户所属的相应组

假设您在名为 usertogroup 的段落（位于 transforms.conf 文件内）中指定了一个查找表。该查找表包含（至少）两个字段：user 和 group。您的事件包含名为 local_user 的字段。对于各事件，以下搜索检查查看 local_user 字段中的值在查找表中的 user 字段中是否有对应的值。对于匹配的任何条目，将查找表中 group 字段的值写入事件的字段 user_group 中。

```
... | lookup usertogroup user as local_user OUTPUT group as user_group
```

延伸示例

2. 查找价格和供应商信息并返回供应商售出的每个产品的计数

此示例使用来自搜索教程的 tutorialdata.zip 文件。您可以下载此文件并按照说明上载教程数据到您的 Splunk 部署。另外，此示例使用 prices.csv 和 vendors.csv 文件。若要在 Splunk 部署中使用本示例，下载这些 CSV 文件并为 prices.csv 和 vendors.csv 文件完成搜索教程中“使用字段查找”一节内列出的步骤。当您为 vendors.csv 文件创建查找定义时，将查找命名为 vendors_lookup。您可以跳过教程内有关查找自动化的步骤。

本示例计算每个供应商出售的每个产品的计数。

prices.csv 文件包含产品名称、价格和代码。例如：

productID	product_name	price	sale_price	代码
DB-SG-G01	Mediocre Kingdoms	24.99	19.99	A
DC-SG-G02	Dream Crusher	39.99	24.99	B
FS-SG-G03	Final Sequel	24.99	16.99	C
WC-SH-G04	World of Cheese	24.99	19.99	D

vendors.csv 文件包含供应商信息，如供应商名称、所在城市和 ID。例如：

Vendor	VendorCity	VendorID	VendorLatitude	VendorLongitude	VendorStateProvince	VendorCountry	权重
Anchorage Gaming	安克雷奇	1001	61.17440033	-149.9960022	阿拉斯加州	美国	3
Games of Salt Lake	Salt Lake City	1002	40.78839874	-111.9779968	犹他州	美国	3
New Jack Games	纽约	1003	40.63980103	-73.77890015	纽约	美国	4
Seals Gaming	旧金山	1004	37.61899948	-122.375	加利福尼亚州	美国	5

搜索将查询 vendor_sales.log 文件，即 tutorialdata.zip 文件的一部分。vendor_sales.log 文件包含 VendorID、代码和 AcctID 字段。例如：

Vendor_sales.log 文件中的条目
[13/Mar/2018:18:24:02] VendorID=5036 Code=B AcctID=6024298300471575
[13/Mar/2018:18:23:46] VendorID=7026 Code=C AcctID=8702194102896748
[13/Mar/2018:18:23:31] VendorID=1043 Code=B AcctID=2063718909897951
[13/Mar/2018:18:22:59] VendorID=1243 Code=F AcctID=8768831614147676

以下搜索计算每个供应商售出的每种产品的计数并使用时间范围 All time。

```
sourcetype=vendor_* | stats count by Code VendorID | lookup prices_lookup Code OUTPUTNEW product_name
```

- stats 命令按 Code 和 VendorID 计算 count。
- lookup 命令使用 prices_lookup 匹配每个事件中的 Code 字段并返回产品名称。

搜索结果在统计选项卡上显示。

新搜索

sourcetype=vendor_* | stats count by Code VendorID | lookup prices_lookup Code OUTPUTNEW product_name

30,244 个事件 (18/06/05 13:21:58.000 之前) 无事件采样

另存为 关闭

任务 智能模式

事件 模式 统计信息 (6,001) 可视化

每页 20 个 格式 预览

Code	VendorID	count	product_name
A	1001	4	Mediocre Kingdoms
A	1002	1	Mediocre Kingdoms
A	1003	5	Mediocre Kingdoms
A	1004	8	Mediocre Kingdoms
A	1005	4	Mediocre Kingdoms
A	1006	5	Mediocre Kingdoms
A	1007	2	Mediocre Kingdoms
A	1008	5	Mediocre Kingdoms
A	1009	4	Mediocre Kingdoms
A	1010	5	Mediocre Kingdoms

您可以通过使用 `vendors_lookup` 扩展搜索以显示有关供应商的更多信息。

使用 `table` 命令仅返回您需要的字段。在此示例中，您想要 `product_name`、`VendorID` 和 `count`。使用 `vendors_lookup` 文件输出 `vendors.csv` 文件中和每个事件中 `VendorID` 匹配的所有字段。

```
sourcetype=vendor_* | stats count by Code VendorID | lookup prices_lookup Code OUTPUTNEW product_name | table product_name
VendorID count | lookup vendors_lookup VendorID
```

修改的搜索结果显示在统计选项卡上。

新搜索

sourcetype=vendor_* | stats count by Code VendorID | lookup prices_lookup Code OUTPUTNEW product_name | table product_name VendorID count | lookup vendors_lookup VendorID

30,244 个事件 (18/06/05 13:23:07.000 之前) 无事件采样

另存为 关闭

任务 智能模式

事件 模式 统计信息 (6,001) 可视化

每页 20 个 格式 预览

product_name	VendorID	count	Vendor	VendorCity	VendorCountry	VendorLatitude	VendorLongitude	VendorStateProvince	Weight
Mediocre Kingdoms	1001	4	Anchorage Gaming	Anchorage	United States	61.17440033	-149.9960022	Alaska	3
Mediocre Kingdoms	1002	1	Games of Salt Lake	Salt Lake City	United States	40.78839874	-111.9779968	Utah	3
Mediocre Kingdoms	1003	5	New Jack Games	New York	United States	40.63980103	-73.77890015	New York	4
Mediocre Kingdoms	1004	8	Seals Gaming	San Francisco	United States	37.61899948	-122.375	California	5
Mediocre Kingdoms	1005	4	Lost Angels Games	Los Angeles	United States	33.94250107	-118.4079971	California	5
Mediocre Kingdoms	1006	5	Flyin Hawaiian Hobbyist	Honolulu	United States	21.31870079	-157.9219971	Hawaii	3

要扩大搜索显示地图上的结果，请参阅 `geostats` 命令。

另请参阅

命令

- appendcols
- inputlookup
- outputlookup

相关信息

《知识管理器手册》中的“关于查找”

makecontinuous

描述

通过在没有数据的时间段添加空数据桶以及量化存在数据的时间段来使得 X 轴上的字段在数字上连续。可用 chart 和 timechart 命令调用 X 轴字段。

语法

```
makecontinuous [<field>] <bins-options>...
```

必要参数

<bins-options>

数据类型: bins | span | start-end

描述: 离散化选项。详情请参阅“Bins 选项”。

可选参数

<field>

数据类型: <field>

描述: 指定一个字段名称。

Bins 选项

bins

语法: bins=<int>

描述: 设置离散为数据箱的最大数量。

span

语法: <log-span> | <span-length>

描述: 使用基于时间的跨度长度或基于对数的跨度设置每个数据箱的大小。

<start-end>

语法: end=<num> | start=<num>

描述: 设置数字型数据箱的最小和最大范围。在 [start, end] 范围之外的数据将遭丢弃。

跨度选项

<log-span>

语法: [<num>] log [<num>]

描述: 设置为基于对数的跨度。第一个数字为系数, 第二个数字为底。若提供第一个数字, 该数字必须是大于等于 1.0 且小于底的实数。如果提供了底, 则底必须是大于 1.0 的实数, 即必须严格大于 1。

span-length

语法: [<timescale>]

描述: 基于时间的跨度长度。

语法: <int>

描述: 每个数据箱的跨度。如果使用 timescale, 则将其用作时间范围。否则, 这是一个绝对的数据箱“长度”。

<timescale>

语法: <sec> | <min> | <hr> | <day> | <month> | <subseconds>

描述: 时间刻度单位。

时间刻度	语法	描述
<sec>	s sec secs second seconds	时间刻度(秒)。
<min>	m min mins minute minutes	时间刻度(分钟)。
<hr>	h hr hrs hour hours	时间刻度(小时)。
<day>	d day days	时间刻度(天)。
<month>	mon month months	时间刻度(月)。

<subseconds> | us | ms | cs | ds

单位为微秒 (us)、毫秒 (ms)、百分之一秒 (cs) 或十分之一秒 (ds) 的时间刻度。

用法

`makecontinuous` 命令是一个转换命令。请参阅“命令类型”。

示例

示例 1:

使 `_time` 连续，时间跨度为 10 分钟。

```
... | makecontinuous _time span=10m
```

另请参阅

`chart`, `timechart`

makemv

描述

通过使用简单的字符串分隔符拆分单值字段，将此字段转换为多值字段。分隔符可以是多字符分隔符。或者，也可以用 `regex` 拆分字段。

`makemv` 命令不适用于内部字段。

请参阅《知识管理器手册》中的“使用默认字段”。

语法

```
makemv [delim=<string> | tokenizer=<string>] [allowempty=<bool>] [setsv=<bool>] <field>
```

必要参数

`field`

语法: `<field>`

描述: 指定字段的名称。

可选参数

`delim`

语法: `delim=<string>`

描述: 字符串值用作分隔符。每次出现此字符串时拆分 `field` 中的值。

默认值: 一个空格 (" ")。

`tokenizer`

语法: `tokenizer=<string>`

描述: 包含捕获组的 `regex`，针对字段文本重复匹配。对于每个匹配，第一个捕获组均用作新建的多值字段的值。

`allowempty`

语法: `allowempty=<bool>`

描述: 指定是否允许多值字段中的空字符串值。使用 `delim=true` 时，分隔符字符串的重复在多值字段中产生空的字符串值。例如：如果 `delim=","` 和 `field="a,,b"`，默认情况下，不能产生空的字符串的值。使用 `tokenizer` 参数时，零长度匹配产生空的字符串值。但是默认不会生成任何值。

默认值: `false`

`setsv`

语法: `setsv=<bool>`

描述: 若为 `true`, `makemv` 命令会将字段中已确定的值合并为在同一个字段上设置的单个值。（同样字段中多值与单值同时存在是该标记有问题的一个方面。）

默认值: `false`

用法

`makemv` 命令属于可分配的流命令。请参阅“命令类型”。

您可以对多值字段使用评估函数和统计函数或返回多值字段。

示例

1. 使用逗号分隔字段值

对于 `sendmail` 搜索结果，将 “`senders`” 的值分隔为多个值。显示上限值。

```
eventtype="sendmail" | makemv delim="," senders | top senders
```

2. 使用冒号分隔符并允许空值

将 “`foo`” 的值分隔成多个值。

```
... | makemv delim=":" allowempty=true foo
```

3. 使用正则表达式分隔值

以下搜索可新建结果并将三个值添加到 `my_multival` 字段。`makemv` 命令用于将值安排到单独的行中。

```
| makeresults | eval my_multival="one,two,three" | makemv tokenizer="([^\,]+),?" my_multival
```

另请参阅

命令：

`mvcombine`
`mvexpand`
`nomv`

函数：

多值 `eval` 函数
多值统计和 `chart` 函数
拆分

makeresults

描述

在临时内存中生成指定数量的搜索结果。

如果未指定任何可选参数，则此命令在本地计算机上运行，并生成一个仅带有 `_time` 字段的结果。

语法

要求的语法以粗体表示。

```
| makeresults
[<count>]
[<annotate>]
[<splunk-server>]
[<splunk-server-group>...]
```

必要参数

无。

可选参数

<count>

语法： `count=<num>`

描述： 要生成的结果数。如果您未指定 `annotate` 参数，则结果仅有 `_time` 字段。

默认值： 1

<annotate>

语法: `annotate=<bool>`
描述: 如果 `annotate=true`, 则仅使用下表中显示的字段生成结果。
若 `annotate=false`, 则仅使用 `_time` 字段生成结果。
默认值: `false`

使用 `annotate = true` 生成的字段

字段	值
<code>_raw</code>	无。
<code>_time</code>	运行 <code>makergesults</code> 命令的日期和时间。
<code>host</code>	无。
<code>source</code>	无。
<code>sourcetype</code>	无。
<code>splunk_server</code>	<code>makergesults</code> 命令运行的服务器名称。
<code>splunk_server_group</code>	无。

可以使用这些字段计算聚合统计信息。

`<splunk-server>`
语法: `splunk_server=<string>`
描述: 用于在一个特定的服务器上生成结果。使用 "local" 代表搜索头。
默认值: `local`。请参阅用法一节。

`<splunk-server-group>`
语法: `(splunk_server_group=<string>)...`
描述: 用于在一个特定的服务器组或多个组上生成结果。您可以指定多个 `<splunk_server_group>`。
默认值: `none`。请参阅用法一节。

用法

`makergesults` 命令属于报表生成命令。请参阅“命令类型”。

生成命令使用前导管道符且应是搜索中的第一个命令。

`makergesults` 命令创建的搜索结果会创建于临时内存中，并且不会保存到磁盘，也不会建立索引。

将此命令与 `eval` 命令结合使用，即可生成一个空结果供 `eval` 命令使用。请参阅示例一节。

若内部 `_time` 字段不存在，顺序敏感型处理器可能会失效。

指定服务器和服务器组

如果使用 `Splunk Cloud`, 请忽略任何服务器或服务器组参数。

如果您使用的是 `Splunk Enterprise`, 则默认情况下，仅在生成搜索头上生成结果，这相当于指定 `splunk_server=local`。如果提供特定的 `splunk_server` 或 `splunk_server_group`, 则使用 `count` 参数指定的结果数在您指定的所有服务器或服务器组上生成。

若您指定了服务器，无论与其关联的服务器组为何，都将生成该服务器的计数结果。

若您将计数指定为 5 并锁定 3 个服务器，您一共将生成 15 个结果。若 `annotate=true`, 每个服务器的名字都将出现在 `splunk_server` 列中。该列将显示每个服务器生成了 5 个结果。

基本示例

1. 在 `eval` 命令中将结果新建为输入

有时，您想把 `eval` 命令用作搜索中的第一个命令。但是，`eval` 命令会把事件当做输入。您可以在搜索之初用 `makergesults` 命令新建一个虚拟事件。然后就可以在搜索中使用 `eval` 命令了。

```
| makergesults | eval newfield="some value"
```

结果形式如下所示：

_time	newfield
2020/1/9 14:35:58	一些值

2. 确定修改好的事件时间是否大于相对时间

若事件包含 Unix 时间格式的 `scheduled_time` 字段, 请确定计划时间是否大于相对时间。相对时间比当前时间早 1 分钟。本搜索将使用以 `makeresults` 命令开头的子搜索。

```
index=_internal sourcetype=scheduler ( scheduled_time > [ makeresults | eval it=relative_time(now(), "-m") | return $it ] )
```

延伸示例

1. 创建用于测试的每日结果

您可以使用 `makeresults` 命令创建一系列结果以测试您的搜索语法。例如, 以下搜索将创建一组五个结果:

```
| makeresults count=5
```

结果形式如下所示:

_time
2020/1/9 14:35:58
2020/1/9 14:35:58
2020/1/9 14:35:58
2020/1/9 14:35:58
2020/1/9 14:35:58

每个结果都有相同的时间戳, 因此这点本身并不是很有用。但是, 通过添加一些内容, 您可以创建一组唯一的日期。首先添加 `streamstats` 命令来计算结果:

```
| makeresults count=5 | streamstats count
```

结果形式如下所示:

_time	计数
2020/1/9 14:35:58	1
2020/1/9 14:35:58	2
2020/1/9 14:35:58	3
2020/1/9 14:35:58	4
2020/1/9 14:35:58	5

现在, 您可以借助 `eval` 命令使用该计数在 `_time` 字段中创建不同的日期。

```
| makeresults count=5 | streamstats count | eval _time=_time-(count*86400)
```

计算会将 `count` 字段中的值乘以一天的秒数。从原始 `_time` 字段中减去结果, 从而得到等于 24 小时前、48 小时前等等的新日期。日期中的秒数各有不同, 因为 `_time` 是在您运行搜索时计算的。

结果形式如下所示:

_time	计数
2020/1/8 14:45:24	1
2020/1/7 14:45:24	2

2020/1/6 14:45:24	3
2020/1/5 14:45:24	4
2020/1/4 14:45:24	5

日期从原始日期 2020-01-09 的前一天开始，并一直往前五天。

需要超过五个结果？您只需更改 `makergesults` 命令中的计数值即可。

2. 创建用于测试的每小时结果

您可以创建一系列的小时而非一系列的天数来进行测试。使用 3600（一小时的秒数）代替 `eval` 命令中的 86400。

```
| makergesults count=5 | streamstats count | eval _time=_time-(count*3600)
```

结果形式如下所示：

_time	计数
2020/1/9 15:35:14	1
2020/1/9 14:35:14	2
2020/1/9 13:35:14	3
2020/1/9 12:35:14	4
2020/1/9 11:35:14	5

请注意，时间戳中的小时间隔为 1 小时。

3. 添加带字符串值的字段

您可以为字段指定值列表。但是要使值出现在单独的结果中，您需要使列表成为多值字段，然后将该多值列表扩展为单独的结果。使用以下搜索，用您的字符串代替 `buttercup` 和她的朋友们：

```
| makergesults | eval test="buttercup rarity tenderhoof dash mcintosh fleetfoot mistmane" | makemv delim=" " test | mvexpand test
```

结果形式如下所示：

_time	test
2020/1/9 16:35:14	buttercup
2020/1/9 16:35:14	rarity
2020/1/9 16:35:14	tenderhoof
2020/1/9 16:35:14	dash
2020/1/9 16:35:14	mcintosh
2020/1/9 16:35:14	fleetfoot
2020/1/9 16:35:14	mistmane

4. 创建带多个字段的一组事件

让我们从创建一组四个事件开始。其中一个事件的 `age` 字段为空值。

```
| makergesults count=4 | streamstats count | eval age = case(count=1, 25, count=2, 39, count=3, 31, count=4, null()) | eval city = case(count=1 OR count=3, "San Francisco", count=2 OR count=4, "Seattle")
```

- `streamstats` 命令用于创建 `count` 字段。`streamstats` 命令会在处理每个事件时计算该事件的累计数量。
- `eval` 命令用于创建两个新字段：`age` 和 `city`。`eval` 命令使用 `count` 字段中的值。

- `case` 函数采用成对的参数，例如 `count=1, 25`。第一个参数是布尔表达式。当该表达式为 `TRUE` 时，将返回相应的第二个参数。

搜索结果如下所示：

_time	age	city	计数
2020/2/5 18:32:07	25	旧金山	1
2020/2/5 18:32:07	39	西雅图	2
2020/2/5 18:32:07	31	旧金山	3
2020/2/5 18:32:07		西雅图	4

在此示例中，`eventstats` 命令会生成每个城市的平均年龄。生成的平均值放置在一个名为 `avg(age)` 的新字段中。

以下搜索与前一个搜索相同，只是在最后添加了 `eventstats` 命令：

```
| makeresults count=4 | streamstats count | eval age = case(count=1, 25, count=2, 39, count=3, 31, count=4, null()) | eval city = case(count=1 OR count=3, "San Francisco", count=2 OR count=4, "Seattle") | eventstats avg(age) BY city
```

- 对于 `San Francisco`，平均年龄为 $28 = (25 + 31) / 2$ 。
- 对于 `Seattle`，只有一个事件有值。平均值是 $39 = 39 / 1$ 。`eventstats` 命令将该平均值放入西雅图的每个事件中（包括不包含 `age` 值的事件）。

搜索结果如下所示：

_time	age	avg(age)	city	计数
2020/2/5 18:32:07	25	28	旧金山	1
2020/2/5 18:32:07	39	39	西雅图	2
2020/2/5 18:32:07	31	28	旧金山	3
2020/2/5 18:32:07		39	西雅图	4

5. 添加带一组随机数字的字段

如果您需要使用一组数字来测试某物，则有两种选择：

- 您可以添加带一组指定数字的字段。这类似于添加带一组字符串值的字段，这种在上面的示例中有作介绍。
- 您也可以使用 `random` 函数添加带一组随机生成的数字的字段，如下所示：

```
| makeresults count=5 | streamstats count | eval test=random()/random()
```

结果形式如下所示：

_time	计数	test
2020/1/8 14:45:24	1	5. 371091109260495
2020/1/7 14:45:24	2	0. 4563314783228324
2020/1/6 14:45:24	3	0. 804991002129475
2020/1/5 14:45:24	4	1. 4946919835236068
2020/1/4 14:45:24	5	24. 193952675772845

使用 `round` 函数将数字四舍五入。例如，下面的搜索会把数字四舍五入到小数点右边的第四位：

```
...| eval test=round(random()/random(),4)
```

结果形式如下所示：

_time	计数	test

2020/1/8 14:45:24	1	5. 3711
2020/1/7 14:45:24	2	0. 4563
2020/1/6 14:45:24	3	0. 8050
2020/1/5 14:45:24	4	1. 4947
2020/1/4 14:45:24	5	24. 1940

另请参阅

命令

gentimes

map

描述

map 命令是一个针对每个输入事件或结果重复运行搜索的循环运算符。您可以在已保存的搜索或临时搜索上运行 map 命令。

语法

要求的语法以粗体表示。

```
map
(<searchoption> | <savedsplunkoption>
[maxsearches=int])
```

必要参数

您必须指定 <searchoption> 或 <savedsplunkoption>。

<savedsplunkoption>

语法: <string>

描述: 为每个输入结果运行的保存的搜索的名称。

默认值: 无默认值。

<searchoption>

语法: search=<string>"

描述: 每个输入结果的即席搜索。例如:

...| map search="search index=_internal earliest=\$myearliest\$ latest=\$mylatest\$".

默认值: 无默认值。

可选参数

maxsearches

语法: maxsearches=<int>

描述: 要运行的最大搜索数。若搜索结果超过您指定的最大数量, 将会生成一条消息。

默认值: 10

用法

已知限制

您可以在搜索管道中在 append 或 appendpipe 命令之后使用 map 命令。

字段名称的变量

使用保存的搜索或文本搜索时, map 命令支持替代输入结果中匹配字段名称的 \$variable\$ 字符串。例如, 含 \$count\$ 字符串的搜索将在输入搜索结果中用 count 字段的值替换变量。

在仪表板 <form> 中使用 map 命令时, 使用双美元符号 (\$\$) 指定变量字符串。例如, \$\$count\$\$. 请参阅“仪表板和表单”。

搜索 ID 字段

`map` 命令还支持以 `$_serial_id$` 形式提供的搜索 ID 字段。搜索 ID 字段将有一个每次运行搜索时都会逐渐增加的数字。换句话说，第一次运行的搜索 ID 值为 1，第二个为 2，依此类推。

基本示例

1. 使用保存的搜索调用 `map` 命令

```
error | localize | map mytimebased_savedsearch
```

2. 映射开始和结束时间值

```
... | map search="search starttimeu::$start$ endtimeu::$end$" maxsearches=10
```

延伸示例

1. 用 `Sudo` 事件查找用户登录

此示例介绍如何查找 `Sudo` 事件，然后使用 `map` 命令追溯到在 `Sudo` 事件之前用户登录的计算机和登录时间。首先，对 `Sudo` 事件执行以下搜索。

```
sourcetype=syslog sudo | stats count by user host
```

此搜索将返回结果表格。

User	Host	Count
userA	serverA	1
userB	serverA	3
userA	serverB	2

通过管道符将这些结果传递给 `map` 命令时，替换用户名。

```
sourcetype=syslog sudo | stats count by user host | map search="search index=ad_summary username=$user$type_logon=ad_last_logon"
```

它将获取上一次搜索的三个结果并在 `ad_summary` 索引中搜索用户登录事件。结果将以表的形式返回。

_time	computername	computertime	username	usertime
2016 年 10 月 12 日, 上午 8:31:35.00	Workstation\$	2016 年 10 月 12 日, 8:25:42	userA	2016 年 10 月 12 日, 8:31:35

(感谢 Splunk 用户 Alacer cogitatus 提供此示例。)

另请参阅

命令

gentimes
搜索

mcollect

描述

将事件转换为指标数据点，并将指标数据点插入到搜索头上的指标索引。除非您将数据转发到索引器，否则搜索头必须有一个指标索引，这样 `mcollect` 才能正常工作。

如果您正在将数据转发至索引器，您的数据将插在索引器上，而不是搜索头上。

只当您的角色具有 `run_mcollect` 功能时，您才能使用 `mcollect` 命令。请参见《确保 Splunk Enterprise 安全》中的“使用功能定义 Splunk 平台上的角色”。

语法

要求的语法以粗体表示。

```
| mcollect index=<string>
| file=<string> ]
| split=<true | false | allnums> ]
| spool=<bool> ]
| prefix_field=<string> ]
| host=<string> ]
| source=<string> ]
| sourcetype=<string> ]
| marker=<string> ]
[ <field-list> ]
```

必要参数

index

语法: index=<string>

描述: 添加收集的指标数据的指标索引名称。

field-list

语法: <field>, ...

描述: 维度字段列表。如果 split=true, 必填。如果 split=false 或 split=allnums, 可选。如果未指定 (表示 split=false), 则 mcollect 会将所有字段视为数据点的维度, 除了 metric_name、prefix_field 和所有内部字段。

默认值: 没有默认值

可选参数

file

语法: file=<string>

描述: 想要写入收集的指标数据的文件名称。只有在 spool=false 时适用。通过指定 file=\$timestamp\$ 或 file=\$random\$, 您可以在文件名中使用时间戳或任何数字。

默认值: \$random\$_metrics.csv

拆分

语法: split=<true | false | allnums>

描述: 确定 mcollect 如何识别事件中的测量值。请参阅“如何使用拆分参数”。

默认值: false

spool

语法: spool=<bool>

描述: 若设置为 true, 指标数据文件写入 Splunk spool 目录 \$SPLUNK_HOME/var/spool/splunk, 这也是为文件建立索引的目录。为文件建立索引后, 文件将被移除。若设置为 false, 文件将写入 \$SPLUNK_HOME/var/run/splunk 目录。除非进一步执行了某种形式的自动化或管理, 否则文件将留在此目录中。

默认值: true

prefix_field

语法: prefix_field=<string>

描述: 只有在 split=true 时适用。如果指定, 将忽略没有该字段的所有数据点。否则, 字段值将作为指标名称的前缀。请参阅“设置前缀字段”

默认值: 没有默认值

host

语法: host=<string>

描述: 您想要为收集的指标数据指定的主机名称。只有在 spool=true 时适用。

默认值: 没有默认值

source

语法: source=<string>

描述: 您想要为收集的指标数据指定的数据来源名称。

默认值: 如果搜索已列入计划, 则是搜索名称。如果是即席搜索, 则是写入 var/spool/splunk 目录包含搜索结果的文件名称。

sourcetype

语法: sourcetype=<string>

描述: 为收集的指标数据指定的来源类型名称。Splunk 平台不会计算用默认来源类型 mcollect_stash 索引的数据的许可证使用情况。如果您将此设置值更改为其他来源类型, Splunk 平台会计算 mcollect 命令索引的任何数据的许可证使用情况。

默认: mcollect_stash

没有来自 Splunk 专业服务团队或 Splunk 支持，不要更改此设置。更改来源类型需要更改 `props.conf` 文件。

marker

语法：`marker=<string>`

描述：`mcollect` 将一个或多个以逗号分隔的键/值对作为维度添加到它所生成的指标数据点中，以便以后搜索这些指标数据点。例如，您可以添加正在运行的 `mcollect` 搜索的名称，如下所示：`marker=savedsearch=firewall_top_src_ip`。这样一来，以后您只需把 `savedsearch=firewall_top_src_ip` 添加到搜索字符串中，即可运行搜索来隔离由那个 `mcollect` 搜索创建的指标数据点。

用法

您使用 `mcollect` 命令将事件转换为要存储于搜索头上指标索引中的指标数据点。指标数据为指标字段使用特定的格式。请参阅指标中的“指标数据格式”。

`mcollect` 命令导致每次运行搜索时新的命令被写入到指标索引。

所有指标搜索命令均区分大小写。举例来说，`mcollect` 会把以下值视为 `metric_name` 的三个不同值：`cap.gear`、`CAP.GEAR` 和 `Cap.Gear`。

如果 `metric_name` 字段为空或完全由空格组成，则 Splunk 平台无法索引包含这类字段的指标数据点。

如果您正在升级到版本 8.0.0

将搜索头和索引器群集升级为 Splunk Enterprise 8.0.x 之后，编辑各搜索头群集上的 `limits.conf` 并将 `[mcollect]` 段落下的 `always_use_single_value_output` 设置设为 `false`。这样设置允许这些节点在您使用 `mcollect` 命令或使用指标汇总将日志转换为指标时，使用“每个指标数据点有多个度量”方案。此方案可增加您的数据存储容量并提高指标搜索性能。

如何使用拆分参数

`split` 参数确定 `mcollect` 如何识别搜索中的测量字段。默认值为 `false`。

当 `split=false` 时，您的搜索需要显式识别其测量字段。必要时，可以使用 `rename` 或 `eval` 转换操作。

- 如果您有单指标事件，您的 `mcollect` 搜索必须生成带 `metric_name` 字段（该字段提供测量值的名称）和 `_value` 字段（提供测量值的数字值）的结果。
- 如果您有多指标事件，您的 `mcollect` 搜索必须生成遵循以下语法的结果：`metric_name:<metric_name><numeric_value>`。`mcollect` 将每个字段视为一个测量值。`mcollect` 将剩余的字段视为维度。

当您设置 `split=true`，您可以使用 `field-list` 识别搜索中的维度。`mcollect` 将不在 `field-list` 中的任何字段转换为测量值。唯一例外的是以下划线和 `prefix_field` 开头的内部字段，如果您设置了这样的字段。

如果设置了 `split=allnums`，则 `mcollect` 会将所有数字字段视为指标测量值，将所有非数字字段视为维度。您可以选择使用 `field-list` 声明 `mcollect` 应将事件中的某些数字字段视为维度。

设置前缀字段

使用 `prefix_field` 参数将前缀应用于事件数据中的指标字段。

例如，如果您有以下数据：

`type=cpu usage=0.78 idle=0.22`

您有两个指标字段：`usage` 和 `idle`。

假设您在该数据的 `mcollect` 搜索中包含以下内容：

`...split=true prefix_field=type...`

因为您已设置 `split = true`，Splunk 软件会自动将这些字段转换为测量值，因为他们在 `<field-list>` 中不会被识别。然后，会将指定的 `prefix_field` 值作为前缀应用于指标字段名称。在此情况下，因为您已指定 `type` 字段作为前缀字段，其值 `cpu` 会变成指标名称前缀。结果如下所示：

<code>metric_name:cpu.usage</code>	<code>metric_name:cpu.idle</code>
0.78	0.22

时间

如果 `_time` 字段出现在结果中，Splunk 软件会将其用作指标数据点的时间戳。如果 `_time` 字段不出现，则使用当前的时间。

field-list

如果未指定 `field-list`，则 `mcollect` 会将所有字段视作它所生成的指标数据点的维度，除了 `prefix_field` 和内部字段（即有着下划线 ‘_’ 前缀的字段）。如果 `field-list` 已指定，列表必须出现在 `mcollect` 命令参数的结尾。如果 `field-list` 已指定，所有字段被视作指标值，除了 `field-list` 中的字段、`prefix-field` 和内部字段。

每个指标值的名称是以 `prefix_field` 值为前缀的字段名称。

实际上，将为每个包含数字值的符合条件的字段返回一个指标数据点。如果一个搜索结果包含多个符合条件的指标名称/值对，结果将拆分成多个指标数据点。

示例

以下示例说明如何使用 `mcollect` 命令将事件转换为多值指标数据点。

1: 生成按用户拆分任务和延迟的指标数据点

以下示例指定应在生成的指标数据点中显示的指标，并按用户拆分。注意：该指标不使用 `split` 参数，因此搜索必须使用 `rename` 转换以显式识别将在数据点中显示的测量值。

```
index="_audit" search_id info total_run_time | stats count(search_id) as jobs avg(total_run_time) as latency by user | rename jobs as metric_name:jobs latency as metric_name:latency | mcollect index=mcollect_test
```

以下示例为该搜索的结果：

<code>_time</code>	用户	<code>metric_name:jobs</code>	<code>metric_name:latency</code>
1563318689	管理员	25	3.8105555555555575
1563318689	splunk-system-user	129	0.2951162790697676

2: 生成按用户拆分事件计数和总运行时间的指标数据点

此搜索设置 `split=true`，因此会自动将 `<field-list>` 不能另行识别为维度的字段转换为指标。搜索会将 `user` 识别为维度。

```
index="_audit" info=completed | stats max(total_run_time) as runtime max(event_count) as events by user | mcollect index=mcollect_test split=t user
```

以下示例为该搜索的结果：

<code>_time</code>	用户	<code>metric_name:runtim</code>	<code>metric_name:events</code>
1563318968	管理员	0.29	293
1563318968	splunk-system-user	0.04	3

另请参阅

命令

```
collect  
meventcollect
```

metadata

描述

`metadata` 命令返回指定索引或分布式搜索节点的数据来源、来源类型或主机列表。`metadata` 命令返回随着时间累积的信息。您可以使用时间范围挑选器查看特定时间范围（如过去 7 天）的索引快照。

请参阅“用法”。

语法

```
| metadata type=<metadata-type> [<index-specifier>]... [splunk_server=<wc-string>] [splunk_server_group=<wc-string>]...
```

必要参数

type

语法: type= hosts | sources | sourcetypes

描述: 要返回的元数据的类型。必须是三个文本字符串 host、sources 或 sourcetypes 之一。

可选参数

index-specifier

语法: index=<index_name>

描述: 指定返回结果的索引。您可以指定多个索引。可以使用通配符字符 (*)。若要匹配非内部索引, 请使用 index=*. 若要匹配内部索引, 请使用 index=_*。

示例: | metadata type=hosts index=cs* index=na* index=ap* index=eu*

默认值: 默认索引, 通常为主索引。

splunk_server

语法: splunk_server=<wc-string>

描述: 指定返回结果的分布式搜索节点。如果您使用 Splunk Cloud, 请忽略此参数。如果您使用 Splunk Enterprise, 您仅可指定一个 splunk_server 参数。但在指定服务器名称时可以使用通配符, 以指示多个服务器。例如, 您可以指定 splunk_server=peer01 或 splunk_server=peer*. 用 local 表示搜索头。

默认值: 所有配置的搜索节点都会返回信息

splunk_server_group

语法: splunk_server_group=<wc-string>...

描述: 把结果限制到一个或多个服务器组。如果您使用 Splunk Cloud, 请忽略此参数。可以在字符串中指定一个通配符字符, 来指示多个服务器组。

用法

metadata 命令属于报表生成命令。请参阅“命令类型”。

生成命令使用前导管道符且应是搜索中的第一个命令。

尽管 metadata 命令从所有对等节点上获取数据, 在元数据之后运行的命令仅在搜索头上运行。

对于指定的 metadata 类型的每一个值, 该命令将显示见到的第一个、最后一个及最近的事件。例如, 如果搜索:

```
| metadata type=hosts
```

您的结果应类似如下所示:

	firstTime	host	lastTime	recentTime	totalCount	type
1	1293005265	apache3splunk.com	1293609574	1293609574	27888	hosts
2	1293005220	apache2splunk.com	1293609574	1293609574	27705	hosts
3	1293005265	apache1splunk.com	1293609555	1293609555	9199	hosts
4	1293055241	mysqlsplunk.com	1293492848	1293492848	180	hosts

- firstTime 字段是索引器第一次从本主机见到某事件的时间戳。
- lastTime 字段是索引器最后一次从本主机见到某事件的时间戳。
- recentTime 字段是索引器最近一次从本主机上见到某事件的 indexetime。也就是说, 是指最后一次更新的时间。
- totalCount 字段是从本主机上见到的事件的总数。
- type 字段是要显示的元数据的指定类型。由于本搜索指定了 type=hosts, 因此还有一个 host 列。

在大多数情况下, 当数据进行实时流推送时, lastTime 和 recentTime 的字段值相等。但是, 如果数据为历史数据, 则这两个字段的值可能会有所不同。

在小规模测试环境下, 数据会是完整的。然而, 在每个类别都具有大量值的环境中, 数据可能不完整。此为有意为之, 目的是让 metadata 命令在合理的时间和内存消耗范围内运行。

实时搜索

在会返回大量结果的实时搜索中运行 metadata 命令会非常迅速地消耗 Splunk 服务器上的所有可用内存。在实时搜索中使用 metadata 命令时, 请格外小心。

时间范围

使用“时间范围挑选器”设置时间范围。您不能在搜索字符串中使用 `earliest` 或 `latest` 时间范围修饰符。必须在第一个管道符命令之前设置时间范围修饰符，且通常来说生成命令不允许在第一个管道之前指定任何命令。

如果您指定的搜索时间范围不是 `All Time`，搜索结果可能不精确。元数据作为索引上每个数据桶的汇总数量存储。根据您指定的时间范围确定包括还是不包括数据桶。

例如，您运行以下指定时间范围为 `Last 7 days` 的搜索。对应时间范围是 1 月 1 日到 1 月 7 日。

```
| metadata type=sourcetypes index=ap
```

索引上有个包含 12 月 31 和 1 月 1 日的事件的数据桶。来自该数据桶的元数据包括在搜索返回的信息中。

最大结果数

默认情况下，最多返回 10,000 个结果。此最大值受 `limits.conf` 文件中 `[metadata]` 段落的 `maxresultrows` 设置控制。

示例

1. 搜索多个索引

返回代表不同区域的索引元数据。

```
| metadata type=hosts index=cs* index=na* index=ap* index=eu*
```

2. 搜索 `sourcetype`

返回 `_internal` 索引中事件的 `sourcetypes` 值。

```
| metadata type=sourcetypes index=_internal
```

将返回以下报表。

The screenshot shows the Splunk search interface with the 'Statistics' tab selected. The table displays 11 rows of data, each representing a sourcetype with its firstTime, lastTime, recentTime, and type. The columns are labeled: firstTime, lastTime, recentTime, sourcetype, totalCount, and type.

firstTime	lastTime	recentTime	sourcetype	totalCount	type
1480952400	1483458059	1483543802	mongod	3104	sourcetypes
1480952431	1483543801	1483543809	scheduler	15178	sourcetypes
1481816674	1483112457	1483198203	splunk_migration	4	sourcetypes
1481816665	1483112449	1483198203	splunk_version	4	sourcetypes
1480952409	1483458062	1483543802	splunk_web_access	30	sourcetypes
1480952405	1483543803	1483543808	splunk_web_service	5370	sourcetypes
1480952397	1483543808	1483543812	splunkd	4208946	sourcetypes
1480952407	1483543801	1483543808	splunkd_access	12641	sourcetypes
1480952400	1483458056	1483543802	splunkd_conf	30	sourcetypes
1480952399	1483458056	1483543802	splunkd_stderr	56	sourcetypes
1480952409	1483543351	1483543809	splunkd_ui_access	4316	sourcetypes

3. 设置 `metadata` 命令返回的结果的格式

您还可以使用 `fieldformat` 命令将 `firstTime`、`lastTime` 和 `recentTime` 列的结果设置为更方易读的格式。

```
| metadata type=sourcetypes index=_internal | rename totalCount as Count firstTime as "First Event" lastTime as "Last Event" recentTime as "Last Update" | fieldformat Count=toString(Count, "commas") | fieldformat "First Event"=strftime('First Event', "%c") | fieldformat "Last Event"=strftime('Last Event', "%c") | fieldformat "Last Update"=strftime('Last Update', "%c")
```

单击 `Count` 字段标签可对结果进行分类并将计数最高的排在首位。现在，结果更具可读性：

事件							模式	统计信息 (11)	可视化
每页 20 个		✓ 格式	预览						
sourcetype	type	First Event	Last Event	Last Update	Count				
splunkd	sourcetypes	Wed May 24 14:48:29 2017	Thu May 25 14:33:11 2017	Thu May 25 14:33:14 2017	267,038				
splunkd_ui_access	sourcetypes	Wed May 24 14:48:37 2017	Thu May 25 14:33:13 2017	Thu May 25 14:33:14 2017	10,207				
splunkd_access	sourcetypes	Wed May 24 14:48:36 2017	Thu May 25 14:32:12 2017	Thu May 25 14:32:15 2017	2,483				
splunk_web_service	sourcetypes	Wed May 24 14:48:34 2017	Thu May 25 14:32:11 2017	Thu May 25 14:32:12 2017	738				
splunk_web_access	sourcetypes	Wed May 24 14:48:37 2017	Thu May 25 14:32:12 2017	Thu May 25 14:32:12 2017	719				
mongod	sourcetypes	Wed May 24 14:48:31 2017	Wed May 24 15:00:21 2017	Wed May 24 15:00:23 2017	226				
scheduler	sourcetypes	Wed May 24 14:49:02 2017	Thu May 25 14:17:02 2017	Thu May 25 14:17:02 2017	36				
splunkd_stdout-too_small	sourcetypes	Wed May 24 14:50:46 2017	Wed May 24 14:50:49 2017	Wed May 24 14:50:49 2017	26				
splunkd_stderr	sourcetypes	Wed May 24 14:48:31 2017	Wed May 24 15:00:18 2017	Wed May 24 15:00:19 2017	5				
first_install-too_small	sourcetypes	Wed May 24 14:48:27 2017	Wed May 24 14:48:27 2017	Wed May 24 14:48:32 2017	4				
splunkd_conf	sourcetypes	Wed May 24 14:48:31 2017	Wed May 24 15:00:18 2017	Wed May 24 15:00:19 2017	3				

4. 返回特定服务器的特定索引中事件的 "sourcetype" 值

返回服务器 foo 的 "_audit" 索引中事件的 "sourcetype" 值。

```
| metadata type=sourcetypes index=_audit splunk_server=foo
```

另请参阅

[dbinspect](#)
[tstats](#)

metasearch

描述

根据 <logical-expression> 中的条件从索引中检索事件 metadata。

语法

```
metasearch [<logical-expression>]
```

可选参数

<logical-expression>

语法: <time-opts> | <search-modifier> | [NOT] <logical-expression> | <index-expression> | <comparison-expression> | <logical-expression> [OR <logical-expression>]

描述: 包括时间和搜索修饰符, 比较和索引表达式。

逻辑表达式

<comparison-expression>

语法: <field><cmp><value>

描述: 将一个字段与另一个字段的文本值或值进行比较。

<index-expression>

语法: "<string>" | <term> | <search-modifier>

<time-opts>

语法: [<timeformat>] [<time-modifier>]...

比较表达式

<cmp>

语法: = | != | < | <= | > | >=

描述: 比较运算符。

<field>

语法: <string>

描述: metasearch 命令返回的其中一个字段的名称。请参阅[用法](#)。

<lit-value>

语法: <string> | <num>

描述: 字段的确切或文本值，此值用于比较表达式中。

<value>

语法: <lit-value> | <field>

描述: 在比较表达式中，字段的文本值或另一字段名称。<lit-value> 必须是数字或字符串。

索引表达式

<search-modifier>

语法: <field-specifier> | <savedsplunk-specifier> | <tag-specifier>

时间选项

此搜索可以使用许多灵活的选项进行基于时间的搜索。有关时间调节器的列表，请参阅《搜索手册》中的“与搜索结合使用的
时间调节器”主题。

<timeformat>

语法: timeformat=<string>

描述: 设置用于 starttime 和 endtime 条件的时间格式。默认情况下，时间戳的格式
为: timeformat=%m/%d/%Y:%H:%M:%S。

<time-modifier>

语法: earliest=<time_modifier> | latest=<time_modifier>

描述: 指定开始时间和结束时间（使用相对或绝对时间）。有关时间调节器索引的更多信息，请参阅《搜索手册》中
的“在搜索中指定时间调节器”。

用法

metasearch 命令属于事件生成命令。请参阅“命令类型”。

生成命令使用前导管道符且应是搜索中的第一个命令。

metasearch 命令会返回以下字段。

字段	描述
host	一个默认字段，包含生成了事件的网络设备的主机名或 IP 地址。
index	数据存储库。当 Splunk 平台索引原始数据时，会将数据转换为可搜索事件。
source	一个默认字段，用于标识事件的来源，即事件起源于哪里。
sourcetype	一个默认字段，用于标识事件的数据结构。
splunk_server	安装了 Splunk Enterprise 的实例的名称。
_time	_time 字段包含以 UNIX 时间格式表示的事件时间戳。

示例

示例 1:

从主机 "webserver1" 返回含有 "404" 的事件的默认索引上的元数据。

```
| metasearch 404 host="webserver1"
```

另请参阅

命令

 metadata
 搜索

meventcollect

描述

将流搜索命令生成的事件转换为指标数据点，并将数据插入索引器上的指标索引中。

只当您的角色具有 `run_mcollect` 功能时，您才能使用 `meventcollect` 命令。请参见《确保 Splunk Enterprise 安全》中的“使用功能定义 Splunk 平台上的角色”。

语法

要求的语法以粗体表示。

```
| meventcollect index=<string>
| file=<string> ]
| split=<bool> ]
| spool=<bool> ]
| prefix_field=<string> ]
| host=<string> ]
| source=<string> ]
| sourcetype=<string> ]
| <field-list> ]
```

必要参数

`index`

语法: `index=<string>`

描述: 添加收集的指标数据的指标索引名称。

`field-list`

语法: `<field>, ...`

描述: 维度字段列表。如果 `split=true`, 必填。如果 `split=false`, 可选。如果未指定（表示 `split=false`），则 `meventcollect` 会将所有字段视为数据点的维度，除了 `metric_name`、`prefix_field` 和所有内部字段。

默认值: 没有默认值

可选参数

`拆分`

语法: `split=<bool>`

描述: 确定 `meventcollect` 如何识别事件中的测量值。请参阅“如何使用拆分参数”。

默认值: `false`

`spool`

语法: `spool=<bool>`

描述: 若设置为 `true`, `meventcollect` 会把指标数据文件写入 Splunk 后台打印目录 `$SPLUNK_HOME/var/spool/splunk`, 文件会在这里自动建立索引。若设置为 `false`, `meventcollect` 会把该文件写入 `$SPLUNK_HOME/var/run/splunk` 目录。除非进一步执行了某种形式的自动化或管理，否则文件将留在此目录中。

默认值: `true`

`prefix_field`

语法: `prefix_field=<string>`

描述: 只有在 `split=true` 时适用。如果指定，则 `meventcollect` 会忽略所有不带此字段的数据点。如果未指定，则 `meventcollect` 会把字段值以前缀形式添加到指标名称中。请参阅“设置前缀字段”。

默认值: 没有默认值

`host`

语法: `host=<string>`

描述: 您想要为收集的指标数据指定的主机名称。只有在 `spool=true` 时适用。

默认值: 没有默认值

`source`

语法: `source=<string>`

描述: 您想要为收集的指标数据指定的数据来源名称。

默认值: 如果搜索已列入计划，则是搜索名称。如果是即席搜索，则 `meventcollect` 会把文件名称写入包含搜索结果的 `var/spool/splunk` 目录。

`sourcetype`

语法: `sourcetype=<string>`

描述: 您想要为收集的指标数据指定的来源类型名称。

默认: `metrics_csv`

没有来自 Splunk 专业服务团队或 Splunk 支持，不要更改此设置。更改来源类型需要更改 `props.conf` 文件。

用法

您使用 `meventcollect` 命令将流事件转换为要存储于索引器上指标索引中的指标数据。指标数据为指标字段使用特定的格式。请参阅指标中的“指标数据格式”。

只有流命令可以放在 `meventcollect` 命令前，以便结果可以插入索引器。如果您想要运行使用转换命令生成指标数据点的搜索，请使用 `mcollect` 而不是 `meventcollect`。

`meventcollect` 命令导致每次运行搜索时新的命令被写入到指标索引。另外，如果针对大量数据运行 `meventcollect` 搜索，则它可能会使容量没有非常大的索引器和索引器群集不堪重负。

所有指标搜索命令均区分大小写。举例来说，`meventcollect` 会把以下值视为 `metric_name` 的三个不同值：`cap.gear`、`CAP.GEAR` 和 `Cap.Gear`。

如果 `metric_name` 字段为空或完全由空格组成，则 Splunk 平台无法索引包含这类字段的指标数据点。

如何使用拆分参数

`split` 参数确定 `meventcollect` 如何识别搜索中的测量字段。默认值为 `false`。

当 `split=false` 时，您的搜索需要显式识别其测量字段。必要时，可以使用 `rename` 或 `eval` 转换操作。

- 如果您有单指标事件，您的 `meventcollect` 搜索必须生成带 `metric_name` 字段（该字段提供测量值的名称）和 `_value` 字段（提供测量值的数字值）的结果。
- 如果您有多指标事件，您的 `meventcollect` 搜索必须生成遵循以下语法的结果：`metric_name:<metric_name>=<numeric_value>`。各字段将被视为测量值。`meventcollect` 将剩余的字段视为维度。

当您设置 `split=true`，您可以使用 `field-list` 识别搜索中的维度。`meventcollect` 将不在 `field-list` 中的任何字段转换为测量值。唯一例外的是以下划线和 `prefix_field` 开头的内部字段，如果您设置了这样的字段。

设置前缀字段

使用 `prefix_field` 参数将前缀应用于事件数据中的指标字段。

例如，如果您有以下数据：

```
type=cpu usage=0.78 idle=0.22
```

您有两个指标字段：`usage` 和 `idle`。

假设您在该数据的 `mcatalog` 搜索中包含以下内容：

```
...split=true prefix_field=type...
```

因为您已设置 `split = true`，Splunk 软件会自动将这些字段转换为测量值，因为他们在 `<field-list>` 中不会被识别。然后，会将指定的 `prefix_field` 值作为前缀应用于指标字段名称。在此情况下，因为您已指定 `type` 字段作为前缀字段，其值 `cpu` 会变成指标名称前缀。结果如下所示：

metric_name:cpu.usage	metric_name:cpu.idle
0.78	0.22

示例

1: 将 `metrics.log` 数据收集到指标索引

以下示例显示如何将指标日志数据收集到名为 '`my_metric_index`' 的指标索引。

```
index=_internal source=*/metrics.log | eval prefix = group + "." + name | meventcollect index=my_metric_index split=true  
prefix_field=prefix name group
```

另请参阅

命令

`collect`
`mcollect`

mpreview

描述

使用 `mpreview` 可以了解存储在指标索引中的各种指标时间序列的类型，并可以排除指标数据中存在的问题。

`mpreview` 返回指定指标索引中与提供的筛选器匹配的原始指标数据点的预览。默认情况下，`mpreview` 的目标是从与搜索相关联的每个指标时间序列索引文件（`.tsidx` 文件）中，为每个指标时间序列检索五个指标数据点。您可以使用 `target_per_timeseries` 参数更改此目标值。

根据设计，`mpreview` 会以 JSON 格式返回指标数据点。

在您将 Splunk 平台升级到 8.0.x 之前，`mpreview` 命令无法搜索索引的数据。

只当您的角色具有 `run_msearch` 功能时，您才能使用 `mpreview` 命令。请参见《确保 Splunk Enterprise 安全》中的“使用功能定义 Splunk 平台上的角色”。

语法

要求的语法以粗体表示。

```
| mpreview
[filter=<string>]
[<index-opt>]...
[splunk_server=<wc-string>]
[splunk_server_group=<wc-string>]...
[earliest=<time-specifier>]
[latest=<time-specifier>] :
[chunk_size=<unsigned-integer>]
[target_per_timeseries=<unsigned-integer>]
```

必要参数

无。默认返回所有类型术语。

可选参数

chunk_size

语法：`chunk_size=<unsigned-integer>`

描述：高级选项。当 Splunk 软件处理搜索时，此参数控制一次从单个时间序列索引文件（`.tsidx` 文件）中检索多少个指标时间序列。只有当您发现特定的 `mpreview` 搜索使用了过多的内存或不频繁地返回事件时，才可以将其设置从默认值降至更低一点。当搜索接过高的基数维度（具有大量不同值的维度）进行分组时，可能就会发生这种情况。发生这种情况时，较低的 `chunk_size` 值可使 `mpreview` 搜索的响应速度更快，但完成的速度可能会变慢。另一方面，较高的 `chunk_size` 可以帮助长时间运行的搜索完成得更快，但可能也会因此导致搜索响应速度降低。对于 `mpreview`，`chunk_size` 不能设置为小于 10 的值。

有关此设置的更多信息，请参阅“使用 `chunk_size` 来调节 `mpreview` 性能”。

默认值：1000

`chunk_size` 参数的默认值由 `limits.conf` 的 `[msearch]` 段落中的 `chunk_size` 设置进行设置。

earliest

语法：`earliest=<time-specifier>`

描述：指定搜索的时间范围的最早 `_time`。您可以指定精确时间（`earliest="11/5/2016:20:00:00"`）或相对时间（`earliest=-h` 或 `earliest=@w0`）。

有关设置精确时间的更多信息，请参阅“日期和时间格式变量”。有关设置相对时间的更多信息，请参阅“时间调节器”。只有当搜索具有毫秒级时间戳分辨率的指标索引时，亚秒选项才可用。

过滤器

语法：`filter= "<string>"`

描述：维度或 `metric_name` 上的任意布尔表达式。

index-opt

语法：`index=<index-name> (index=<index-name>)...`

描述：把搜索限制到一个或多个索引中的结果。您可以使用通配符字符（*）。若要匹配非内部索引，请使用 `index=*`。若要匹配内部索引，请使用 `index=_*`。

latest

语法: latest=<time-specifier>

描述: 指定搜索的 _time 范围的最晚时间。您可以指定精确时间 (latest="11/12/2016:20:00:00") 或相对时间 (latest=-30m 或 latest=@w6)。

有关设置精确时间的更多信息，请参阅“日期和时间格式变量”。有关设置相对时间的更多信息，请参阅“时间调节器”。只有当搜索具有毫秒级时间戳分辨率的指标索引时，亚秒选项才可用。

splunk_server

语法: splunk_server=<wc-string>

描述: 指定返回结果的分布式搜索节点。如果您使用 Splunk Enterprise，您仅可指定一个 splunk_server 参数。但在指定服务器名称时可以使用通配符，以指示多个服务器。例如，您可以指定 splunk_server=peer01 或 splunk_server=peer*。用 local 表示搜索头。

splunk_server_group

语法: splunk_server_group=<wc-string>

描述: 把结果限制到一个或多个服务器组。如果您使用 Splunk Cloud，请忽略此参数。可以在字符串中指定一个通配符字符，来指示多个服务器组。

target_per_timeseries

语法: target_per_timeseries=<unsigned-integer>

说明 确定要从与 mpreview 搜索关联的每个指标时间序列索引文件 (.tsidx 文件) 中针对每个指标时间序列检索多少个目标数量的指标数据点。如果某个时间序列的 .tsidx 文件中的数据点少于 target_per_timeseries，则搜索头会检索该 .tsidx 文件中该时间序列的所有数据点。

如果将 target_per_timeseries 置为 0，它将返回给定时间范围内每个时间序列的所有可用数据点。此搜索的规模可能很大，因此完成起来很慢。如果必须搜索大量指标数据点，请改用 mstats。

有关此设置的更多信息，请参阅 target_per_timeseries 参数的工作原理。

默认值: 5

target_per_timeseries 参数的默认值由 [msearch] 段落中的 target_per_timeseries 设置进行设置，此段落位于 limits.conf

用法

此搜索命令从与提供的筛选器匹配的指定指标索引生成单个指标数据点列表。筛选器可以是维度或 metric_name 的任意布尔表达式。指定 earliest 和 latest 以覆盖时间范围挑选器设置。

有关设置 earliest 和 latest 的更多信息，请参阅“时间调节器”。

mpreview 命令旨在以 JSON 格式显示单独的指标数据点。如果您想要聚合指标数据点，使用 mstats 命令。

所有指标搜索命令均区分大小写。举例来说，mpreview 会把以下值视为 metric_name 的三个不同值：cap.gear、CAP.GEAR 和 Cap.Gear。

target_per_timeseries 参数的工作原理

未经过滤的 mpreview 搜索可以覆盖非常大量的原始指标数据点。在某些情况下，因为搜索覆盖的数据点数量过多，因此可能导致搜索速度缓慢或无响应。

target_per_timeseries 参数使 mpreview 命令的响应速度更快，同时针对指标数据为您提供范围相对更大的预览。它限制了 mpreview 可以从搜索覆盖的每个 .tsidx 文件中的每个指标时间序列返回的指标数据点的数量。

例如，如果您有 10 个指标 tsidx 文件，每个文件包含 100 个指标时间序列，并且每个时间序列的数据点数不少于 5 个。如果在搜索中设置 target_per_timeseries=5，则搜索应该会返回最多 $10 \times 100 \times 5 = 5000$ 个指标数据点。

另一方面，假设您有 10 个指标 tsidx 文件，每个文件包含 100 个指标时间序列，但是其中 50 个时间序列中有 3 个数据点，而另外 50 个时间序列的数据点数不少于 5 个。如果在搜索中设置 target_per_timeseries=5，则搜索应该会返回 $10 \times ((50 \times 3) + (50 \times 5)) = 4000$ 个数据点。

当 mpreview 搜索覆盖的指标数据点的数量明显大于此搜索覆盖的指标时间序列的数量时，target_per_timeseries 参数就特别有用。如果搜索中的数据点数略大于或等于搜索中的指标时间序列数，则作用不是特别大。

您可以运行此搜索以确定 mpreview 搜索可能覆盖的指标数据点的数量：

```
| metadata index=<metric_index_name> type=hosts datatype=metric | fields totalCount
```

您可以运行此搜索以确定 mpreview 搜索可能覆盖的指标时间序列的数量：

```
| mstats count(*) WHERE index=<metric_index_name> by _timeseries | stats count
```

使用 `chunk_size` 来调节 `mpreview` 性能

如果您发现尽管有 `target_per_timeseries` 参数，但 `mpreview` 的速度仍然较慢或无响应，您也可以使用 `chunk_size` 来调节 `mpreview` 行为。减小 `chunk_size` 可提升搜索的响应速度，但同时可能使搜索更慢完成。增大 `chunk_size` 可帮助 `mpreview` 搜索更快完成，但同时可能会降低响应速度。

示例

1. 返回和特定的筛选器匹配的数据点

此搜索会返回和特定的筛选器匹配的 `_metrics` 索引中的单个数据点。

```
| mpreview index=_metrics filter="group=queue name=indexqueue metric_name=*.current_size"
```

以下是上述搜索的 JSON 格式结果的示例。

i	时间	事件

2. 返回指标索引中的单个数据点

```
| mpreview index=_metrics
```

3. 降低 `chunk_size` 以提高 `mpreview` 性能

以下搜索降低了 `chunk_size`，使其从属于 `_metrics` 索引的 `tsidx` 文件中分批返回 100 个指标时间序列的指标数据点。通常，它会分批返回 1000 个指标时间序列。

```
| mpreview index=_metrics chunk_size=100
```

4. 使用 `target_per_timeseries` 加快 `mpreview` 搜索

以下搜索使用 `target_per_timeseries` 针对 `_metrics` 索引所搜索的每个 `tsidx` 文件中的每个时间序列最多返回五个指标数据点。

```
| mpreview index=_metrics target_per_timeseries=5
```

另请参阅

命令

```
mcatalog  
mcollect  
mstats
```

msearch

`msearch` 命令为 `mpreview` 命令的别名。请参阅 `mpreview` 命令了解语法和示例。

另请参阅

命令

```
mcatalog  
mstats
```

mstats

描述

使用 `mstats` 命令分析指标。此命令在指标索引中的 `measurement`、`metric_name` 和 `dimension` 字段上实施数据。您可以使用历史搜索和实时搜索中的 `mstats`。当您将实时搜索中的 `mstats` 与时间窗口结合使用时，历史搜索会率先运行以回填数据。

当您使用 `mstats` 命令搜索单个 `metric_name` 值或少量 `metric_name` 值时，该命令会提供最佳搜索性能。

语法

要求的语法以粗体表示。

```
| mstats  
[chart=<bool>]  
[<chart-options>]  
[prestats=<bool>]  
[append=<bool>]  
[backfill=<bool>]  
[update_period=<integer>]  
[fillnull_value=<string>]  
[chunk_size=<unsigned int>]  
<stats-metric-term>...  
WHERE [<logical-expression>]...  
[ (BY|GROUPBY) <field-list> ]  
[<span-length>]
```

必要参数

`<stats-metric-term>`

语法： `<stats-func> | <stats-func-value>`

描述： 提供两个选项可用于对指标进行统计计算。使用 `<stats-func>` 对您在参数中指定的一个或多个指标进行统计计算。对于可以使用通配符表示多个指标的情况，请使用 `<stats-func-value>`。不能在单个 `mstats` 搜索中混合使用 `<stats-func>` 语法和 `<stats-func-value>` 语法。

在大多数情况下，请使用 `<stats-func>` 语法。只有在单个指标可能由多个不同的指标名称（例如 `cpu.util` 和 `cpu.utilization`）表示的情况下，才需要使用 `<stats-func-value>` 语法。在这些情况下，您可以应用通配符来捕获 `metric_name` 的所有排列。

有关 `<stats-func>` 和 `<stats-func-value>` 语法选项的详细信息，请参阅“统计指标术语选项”。

可选参数

`append`

语法： `append=<bool>`

描述： 只有当 `prestats=true` 时有效。此参数运行 `mstats` 命令并添加结果到一组现有的结果而不是生成新的结果。

默认值： `false`

`backfill`

语法： `backfill=<bool>`

描述： 仅对有时间窗口的实时搜索有效。`backfill=true` 时，`mstats` 命令在内存实时数据中搜索之前，在历史数据中运行搜索以回填事件。

默认值： `true`

`chart`

语法： `chart=<bool>`

描述： 当设置为 `chart=t` 时，`mstats` 数据输出的格式适用于制作图表。`mstats` 图表模式仅在 `prestats=f` 时有效。

如果提供了 `span`，`mstats` 图表模式格式就类似于 `timechart` 命令的格式，并且最多可以支持一个 `group-by` 字段，该字段用作序列拆分字段。

如果未提供 `span`，则图表模式格式就类似于 `chart` 或 `timechart` 命令的格式。如果没有 `span`，则 `mstats` 图表模式必须有一个或两个分组字段。第一个分组字段代表图表的 `x` 轴。第二个分组字段代表 `y` 轴，并且是一个序列拆分字段。

默认值： `chart=f`

`chart-options`

语法： `chart.limit | chart.agg | chart.usenull | chart.useother | chart.nullstr | chart.otherstr`

描述： 可指定用于优化结果的选项。请参阅本主题中的“图表选项”部分。

`chunk_size`

语法: chunk_size=<unsigned_int>

描述: 高级选项。当 Splunk 软件处理搜索时，此参数控制一次从单个时间序列索引文件 (.tsidx 文件) 中检索多少个指标时间序列。只有当您发现特定的 mstats 搜索使用了过多的内存或不频繁地返回事件时，才可以将其设置从默认值降至更低一点。当搜索按过高的基数维度（具有大量不同值的维度）进行分组时，可能就会发生这种情况。发生这种情况时，较低的 chunk_size 值可使 mstats 搜索的响应速度更快，但完成的速度可能会变慢。另一方面，较高的 chunk_size 可以帮助长时间运行的搜索完成得更快，但可能也会因此导致搜索响应速度降低。对于 mstats，chunk_size 不能设置为小于 10000 的值。

默认值: 10000000 (1 千万)

chunk_size 参数的默认值由 limits.conf 的 [mstats] 段落中的 chunk_size 设置进行设置。

fillnull_value

描述: 此参数设置用户指定的值，mstats 命令用该值替换其 group-by 字段列表中任何字段的空值。空值包括返回的事件的子集中缺少的字段值，以及所有返回的事件中的字段值。如果未提供 fillnull_value 参数，则 mstats 会省略结果中的特定行，这些行包括带一个或多个空字段值的事件。

默认值: 空字符串

<field-list>

语法: <field>, ...

描述: 指定一个或多个字段以不同的子句对结果分组。使用 BY 子句时必需此函数。

<logical-expression>

语法: <time-opts>|<search-modifier>|((NOT)? <logical-expression>) |<index-expression>|<comparison-expression>|(<logical-expression> (OR)? <logical-expression>)

描述: 描述适用于您的搜索的筛选器的表达式。包括时间和搜索调节器，比较和索引表达式。请参阅以下部分查看所有这些逻辑表达式组件的说明。

无法筛选 metric_name。不支持 CASE 或 TERM 指令。也无法使用 WHERE 子句搜索术语或短语。

prestats

语法: prestats=true | false

描述: 指定是否使用 prestats 格式。prestats 格式是 Splunk 内部格式，设计用于可以生成聚合计算的命令使用。当您使用 prestats 格式时，您可以将数据通过管道导入 chart、stats 或 timechart 命令，这些可以接受 prestats 格式。当 prestats 设为 true 时，带有 AS 子句的指令不相关。聚合的字段名称由使用 prestats 格式的命令确定，产生聚合输出。

默认值: false

<span-length>

语法: span=<int><timescale> [every=<int><timescale>]

描述: 每个时间数据箱的跨度。如果和 <timescale> 一起使用，<span-length> 则被视为时间范围。否则，这是一个绝对的数据桶长度。如果未指定 <span-length>，则默认值为 auto，这表示时间数据桶的数字适应于生产一个合理数量的结果。例如，如果最初的秒数用于 <timescale> 且返回的结果过多，则 <timescale> 会改为较长的值（如分钟）以返回较少的时间数据桶。

为了提高 mstats 搜索的性能，您可以选择将 every 参数搭配 span 一起使用，使搜索减少每个跨度的采样数据量。换句话说，您可以设计一个搜索，其中搜索头对搜索覆盖的 every 小时只采样十分钟的 span。请参阅“跨度长度选项”。

update_period

语法: update_period=<integer>

描述: 只有使用实时搜索时有效。以秒为单位指定 mstats 命令实时摘要的更新频率。数字越大代表摘要更新频率越低，对索引处理的影响越小。

默认值: 1000 (1 秒)

统计指标术语选项

<stats-func>

语法: <stats-func> | <mstats-specific-func> ("<metric_name>") [AS <string>]...

描述: 针对一个或多个 metric_name 字段进行统计计算。除非将 prestats 设为 true，否则您可以使用 AS 子句重命名各函数的结果。metric_name 必须用括号括起来。

当您使用 <stats-func> 语法时，WHERE 子句无法根据 metric_name 进行过滤。

<mstats-specific-func>

语法: rate_avg | rate_sum

描述: mstats 特有的两个函数。rate_avg 计算累计计数器指标的每个指标时间序列速率，然后返回这些速率的平均值。rate_sum 的作用与 rate_avg 一样，只是它返回的是速度之和。有关计数器指标和这些函数的更多信息，请参阅《指标》中的“调查计数器指标”。

<stats-func-value>

语法: count(_value) | <function>(_value) [AS <string>] WHERE metric_name=<metric_name>

描述: 指定 _value 字段的基本计数或 _value 字段的函数。_value 字段使用特定的格式存储指标的数值。您可以指定一

个或多个函数。如果不是 `prestats=true`, 您可以使用 AS 重命名函数的结果。

当您使用 `<stats-func-value>` 语法时, WHERE 子句必须根据 `metric_name` 进行过滤。也可以使用通配符。

`stats-func-value` 语法不支持[实时搜索](#)。如果必须运行实时搜索, 请改用 `stats-func` 语法。

下表按函数类型列出了 `mstats` 命令支持的函数。使用表格中的链接查看每个函数的介绍和示例。

函数类型	支持的函数和语法			
聚合函数	<code>avg()</code> <code>count()</code> <code>max()</code> <code>median()</code> <code>min()</code>	<code>perc<num></code> <code>range()</code> <code>stdev()</code> <code>stdevp()</code>	<code>sum()</code> <code>sumsq()</code> <code>upperperc<num></code> <code>var()</code> <code>varp()</code>	
时间函数	<code>earliest()</code> <code>earliest_time()</code> <code>latest()</code>	<code>latest_time()</code> <code>rate()</code>	<code>rate_avg()</code> <code>rate_sum()</code>	

有关使用带有命令的函数的概述, 请参见“[统计和图表函数](#)”。

图表选项

`chart.limit`

语法: `chart.limit=(top | bottom)<int>`

描述: 只有在指定 `column-split` 时有效。使用 `chart.limit` 选项指定应出现于输出中的结果数量。若设置了 `chart.limit=N`, 则会根据每个系列的总和以及您所选择的前缀保留前或后 N 个值。若 `chart.limit=0`, 则将返回所有结果。如果您选择在 `chart.limit` 值之前不提供 `top` 或 `bottom` 前缀, 则 Splunk 软件会提供前 N 个结果。例如, 如果您设置 `chart.limit=10`, 则 Splunk 软件默认会提供前 10 个结果。

此参数与 `chart` 和 `timechart` 命令的 `limit` 参数相同。

默认值: `top10`

`chart.agg`

语法: `chart.agg=(<stats-func> (<evald-field> | <wc-field>) [AS <wc-field>])`

描述: 统计聚合函数。请参阅“[统计指标术语选项](#)”, 获取受支持的函数列表。该函数可应用于 Eval 表达式、字段或字段组。使用 AS 子句将结果放入您已指定其名称的新字段内。可以在字段名称中使用通配符字符。此参数与 `chart` 和 `timechart` 命令的 `agg` 参数相同。

默认值: `sum`

`chart.nullstr`

语法: `chart.nullstr=<string>`

描述: 如果 `chart.usenull` 为 `true`, 则此系列由 `chart.nullstr` 选项值进行标记, 且默认为 `NONE`。此参数与 `chart` 和 `timechart` 命令的 `nullstr` 参数相同。

`chart.otherstr`

语法: `chart.otherstr=<string>`

描述: 如果 `chart.useother` 为 `true`, 则此系列由 `code.otherstr` 选项值进行标记, 且默认为 `OTHER`。此参数与 `chart` 和 `timechart` 命令的 `otherstr` 参数相同。

`chart.usenull`

语法: `chart.usenull=<bool>`

描述: 用于决定是否为不包含 `split-by` 字段的事件创建一个系列。此参数与 `chart` 和 `timechart` 命令的 `usenull` 参数相同。

`chart.useother`

语法: `chart.useother=<bool>`

描述: 指定是否应该为图表中未包含的数据系列 (因为它们未满足 WHERE 子句的条件) 添加一个系列。此参数与 `chart` 和 `timechart` 命令的 `useother` 参数相同。

逻辑表达式选项

`<comparison-expression>`

语法: `<field><comparison-operator><value> | <field> IN (<value-list>)`

描述: 将字段和文本值比较或提供可以出现在字段中的值的列表。

<index-expression>

语法: <term> | <search-modifier>

描述: 使用搜索术语和搜索修饰符描述您想要从索引中检索的事件。

<time-opt>

语法: [<timeformat>] (<time-modifier>)*

描述: 描述搜索的 <starttime> 和 <endtime> 条件的格式

比较表达式选项

<comparison-operator>

语法: = | != | < | <= | > | >=

描述: 您可以在搜索字段-值对时使用比较运算符。含 equal (=) 或 not equal (!=) 运算符的比较表达式比较字符串值。例如, "1" 与 "1.0" 不匹配。含大于或小于运算符 <><= >= 的比较表达式按数字顺序比较两个数字, 并按字典顺序比较其他值。请参阅“用法”。

<field>

语法: <string>

描述: 字段的名称。

<value>

语法: <literal-value>

描述: 在比较表达式中, 这是字段的文本数字或字符串值。

<value-list>

语法: (<literal-value>, <literal-value>, ...)

描述: 和 IN 运算符结合使用, 以指定两个或两个以上值。例如, 用 error IN (400, 402, 404, 406), 而不用 error=400 OR error=402 OR error=404 OR error=406

索引表达式选项

<string>

语法: "<string>"

描述: 指定搜索必须匹配的关键字或加引号的短语。当搜索字符串、带引号的字符串或非搜索修饰符的字符串时, Splunk 软件会搜索 _raw 字段查找匹配事件或结果。

<search-modifier>

语法: <sourcetype-specifier> | <host-specifier> | <source-specifier> | <splunk_server-specifier>

描述: 从指定字段中搜索事件。例如, 搜索一个主机或主机组合、来源和来源类型。请参阅知识管理器手册中的用默认字段搜索。

<sourcetype-specifier>

语法: sourcetype=<string>

描述: 从指定的来源类型字段搜索事件。

<host-specifier>

语法: host=<string>

描述: 从指定的 host 字段搜索事件。

<source-specifier>

语法: source=<string>

描述: 从指定的 source 字段搜索事件。

<splunk_server-specifier>

语法: splunk_server=<string>

描述: 从特定的服务器搜索事件。使用 "local" 代表搜索头。

跨度长度选项

every

语法: every=<int><timescale>

描述: 与 span 结合使用, 用于在搜索的整个时间范围内以离散时间间隔搜索数据。仅当 span 设置为除 auto 以外的有效值时, every 参数才有效。将 every 时间跨度设置为大于 span 时间跨度的值。

这种搜索数据“下采样”法以数据粒度为代价提高了搜索性能。例如, 以下搜索会返回搜索时间范围内每二十秒中的前十秒的 active_logins 测量值的平均值: | mstats avg(active_logins) span=10s every=20s

every 的月间隔恰好是 30 天。every 的年间隔恰好是 365 天。

<timescale>

语法: <sec> | <min> | <hr> | <day> | <month> | <subseconds>

描述: 时间刻度单位。

默认值: sec

时间刻度	语法	描述
<sec>	s sec secs second seconds	时间刻度（秒）。
<min>	m min mins minute minutes	时间刻度（分钟）。
<hr>	h hr hrs hour hours	时间刻度（小时）。
<day>	d day days	时间刻度（天）。
<month>	mon month months	时间刻度（月）。
<subseconds>	us ms cs ds	单位为微秒 (us)、毫秒 (ms)、百分之一秒 (cs) 或十分之一秒 (ds) 的时间刻度。

mstats 搜索的如果是带毫秒级时间戳分辨率配置的指标索引，则只支持亚秒级时间刻度，例如 ms。

有关启用指标索引以便为毫秒级时间戳精度的指标数据点建立索引的更多信息，请参阅：

- 《Splunk Cloud 用户手册》中的“管理 Splunk Cloud 索引”（如果您使用的是 Splunk Cloud）。
- 《管理索引器和索引器群集》中“创建自定义索引”（如果您使用的是 Splunk Enterprise）。

时间选项

<timeformat>

语法: timeformat=<string>

描述: 设置 starttime 和 endtime 条件的时间格式。

默认值: timeformat=%m/%d/%Y:%H:%M:%S。

有关使用可用 timeformat 选项设置精确时间的更多信息，请参阅“日期和时间格式变量”。

只有当搜索具有毫秒级时间戳分辨率的指标索引时，亚秒选项才可用。

<time-modifier>

语法: starttime=<string> | endtime=<string> | earliest=<time_modifier> | latest=<time_modifier>

描述: 指定开始时间和结束时间（使用相对或绝对时间）。

您还可以使用earliest 和 latest 参数来指定搜索的绝对和相对时间范围。

有关相对 <time_modifier> 语法的更多信息，请参阅见“时间调节器”。

有关设置绝对时间范围的更多信息，请参阅“日期和时间格式变量”。只有当搜索具有毫秒级时间戳分辨率的指标索引时，亚秒选项才可用。

starttime

语法: starttime=<string>

描述: 事件必须晚于或等于该时间。starttime 必须符合 timeformat。

endtime

语法: endtime=<string>

描述: 所有事件都必须早于或等于该时间。

用法

mstats 命令是报表生成命令（append=true 时除外）。请参阅“命令类型”。

生成命令使用前导管道符且应是搜索中的第一个命令，当用命令指定 append=true 时除外。

使用 mstats 命令来搜索指标数据。指标数据为指标字段使用特定的格式。请参阅指标中的“指标数据格式”。

所有指标搜索命令均区分大小写。举例来说，mstats 会把以下值视为 metric_name 的三个不同值：cap.gear、CAP.GEAR 和 Cap.Gear。

mstats 对于 metric_name 字段为空或包含空格的指标数据点，搜索无法返回结果。

一起附加 `mstats` 搜索

`mstats` 命令不支持子搜索。您可以使用 `append` 参数将 `mstats` 搜索的结果添加到前一个的 `mstats` 搜索的结果中。有关 `append` 用法示例，请参阅有关 `tstats` 命令的主题。

聚合

如果使用 `<stats-func>` 语法，则只能对 `metric_name` 字段的特定值进行数字聚合。指标名称必须用括号括起来。如果括号中没有特定 `metric_name` 的数据，搜索将仍然有效。

如果使用 `<stats-func-value>` 语法，则只能对 `_value` 字段进行数字聚合。

聚合不允许用于任何其他字段的值，包括 `_time` 字段。

当 `prestats = true` 且运行的是不带聚合字段而使用 `c` 和 `count` 聚合函数的 `mstats` 搜索时，Splunk 软件会将它们视为 `count(_value)` 进行处理。此外，搜索字符串中的后续所有统计函数都必须引用 `_value` 字段。例如：`| mstats count | timechart count(_value)`

通配符字符

`mstats` 命令在任何搜索过滤器中均支持通配符，但以下情况除外：

- 您无法在 `GROUP BY` 子句中使用通配符。
- 如果使用 `<stats_func_value>` 语法，则不能在 `_value` 字段中使用通配符。
- 如果您正在聚合中使用通配符，且正在对它们进行重命名，您的重命名必须有匹配的通配符。

例如，此搜索无效：

```
| mstats sum(*.free) as FreeSum
```

此搜索有效：

```
| mstats sum(*.free) as *FreeSum
```

- 使用 `<stats-func>` 语法时，实时 `mstats` 搜索不能使用通配指标聚合。

例如，如果将以下搜索设置为实时搜索，则搜索无效：

```
| mstats avg(cpu.*) max(cpu.*) where index=sysmetrics
```

以下实时搜索有效：

```
| mstats avg(cpu.sys) max(cpu.usr) where index=sysmetrics
```

WHERE 子句

使用 `WHERE` 子句接受支持的维度筛选。

如果使用 `<stats-func>` 语法，则 `WHERE` 子句无法按 `metric_name` 进行过滤。按 `metric_name` 筛选是根据 `<stats-func>` 参数指定的 `metric_name` 字段进行。

如果使用 `<stats-func-value>` 语法，则 `WHERE` 子句必须按 `metric_name` 进行过滤。

如果您未在 `WHERE` 子句中指定索引名称，`mstats` 命令返回与您的角色相关的默认指标索引结果。如果您不指定索引名称，则没有默认指标索引与您的角色相关，`mstats` 不返回结果。要搜索所有指标索引，请使用 `WHERE index=*`。

如果 `WHERE` 子句和 `BY` 或 `GROUPBY` 子句与 `mstats` 一起使用，则 `WHERE` 子句必须在 `BY` 或 `GROUPBY` 子句之前。

有关为角色定义默认指标索引的更多信息，请参阅确保 *Splunk Enterprise* 安全中的“使用 Splunk Web 添加和编辑角色”。

按指标名称和维度对结果进行分组

您可以按照 `metric_name` 和 `dimension` 字段对结果进行分组。

您还可以按时间进行分组。您必须使用 `<span-length>` 参数指定时间跨度才能按时间数据桶进行分组。例如，`span=1hr` 或 `span=auto`。`<span-length>` 参数独立于 `BY` 子句，可插入到搜索中子句之间的任意位置。

不允许按照 `_value` 或 `_time` 字段进行分组。

按指标时间序列分组

您可以按指标时间序列对结果进行分组。指标时间序列是一组共享相同指标和相同维度字段-值对的指标数据点。按指标时间序列分组可确保在对指标数据源进行统计计算时，不会混淆来自不同指标数据源的数据点。

使用 `BY _timeseries` 按指标时间序列分组。`_timeseries` 字段是内部字段，不会显示在结果中。如果想在搜索中显示 `_timeseries` 值，请在搜索中添加 `| rename _timeseries AS timeseries`。

有关 `_timeseries` 字段的详细概述和示例，请参阅《[指标](#)》中的“对指标时间序列执行统计计算”。

时间维度

`mstats` 命令无法识别以下时间相关的维度。

不受支持的维度		
<code>date_hour</code> <code>date_mday</code> <code>date_minute</code> <code>date_month</code> <code>date_second</code>	<code>date_wday</code> <code>date_year</code> <code>date_zone</code> <code>metric_timestamp</code> <code>time</code>	<code>timeendpos</code> <code>timestamp</code> <code>timestartpos</code>

亚秒级数据箱时间跨度

对于面向已通过配置拥有毫秒级时间戳分辨率的指标索引的 `mstats` 搜索，您只能使用由秒 (ds)、百分之一 (cs)、毫秒 (ms) 或微秒 (us) 组成的亚秒 span 时间刻度。

亚秒 span 时间刻度应为一秒内可均分的数字。例如，`1s = 1000ms`。这意味着有效的毫秒 span 值为 1、2、4、5、8、10、20、25、40、50、100、125、200、250 或 500ms。另外，不允许设置 `span = 1000ms`。改用 `span = 1s`。

有关为索引提供毫秒级时间戳分辨率的更多信息：

- 请参阅《[Splunk Cloud 用户手册](#)》中的“管理 Splunk Cloud 索引”（如果您使用的是 Splunk Cloud）。
- 请参阅《[管理索引和索引群集](#)》中“创建自定义索引”（如果您使用的是 Splunk Enterprise）。

搜索具有不同时间戳分辨率级别的组索引

如果您针对具有不同时间戳分辨率级别的多个指标索引运行 `mstats` 搜索，则搜索结果可能包含具有不同时间戳分辨率的结果。

例如，假设您有两个指标索引。“metrics-second”指标索引具有秒级时间戳分辨率。“metrics-ms”指标索引具有毫秒级时间戳分辨率。您针对这两个索引运行以下搜索：`| mstats count(*) WHERE index=metric* span=100ms`。

此搜索会生成以下结果：

_time	count(cpu.nice)
1549496110	48
1549496110.100	2

1549496110 行计数是来自两个索引的结果。“metric-ms”的计数只包含时间戳为从 1549496110.000 到 1549496110.099 的指标数据点。时间戳为 1549496110.100 到 1549496110.199 的“metric-ms”指标数据点显示在 1549496110.100 行中。

同时，“metric-second”索引中的指标数据点不具有毫秒级的时间戳精度。1549496110 行仅包含那些带 1549496110 时间戳的“metric-second”指标数据点，而且 1549496110.100 行中不包含来自“metric-second”的指标数据点。

`mstats` 搜索任务的时间数据箱限制

Splunk 软件会对使用 `span` 或类似方法按时间对结果进行分组的 `mstats` 搜索任务进行调节。当 Splunk 软件处理这些任务时，它会限制可以在单个 `.tsidx` 文件中分配的“时间数据箱”的数量。

对于具有秒级时间戳分辨率的指标索引，这将会影响具有较大时间范围和非常小的时间跨度的搜索，例如 `span = 1s` 的时间范围为一年的搜索。如果要搜索具有毫秒级时间戳分辨率的指标索引，则可能会在时间范围较短时遇到此限制，例如，`span = 1ms` 的时间范围为一小时的搜索。

此限制由 `limits.conf` 中的 `time_bin_limit` 设置，默认设置为 100 万个数据箱。如果您需要运行这些类型的 `mstats` 搜索任

务，则在每次搜索占用过多内存时降低此值。如果这些类型的搜索任务返回错误，请提高此值。

Splunk 平台会将搜索时间范围除以 group-by 跨度，以此方式估算搜索所需的时间数据箱数。如果结果数字大于 time_bin_limit，则 Splunk 平台会返回错误。

搜索时间范围由搜索的 earliest 和 latest 值决定。有些类型的搜索（如全时搜索）不包含 earliest 和 latest。对于这些搜索，Splunk 平台会在每个单个 TSIDX 文件中进行检查，以得出搜索的时间范围。

指标索引默认情况下拥有秒级时间戳分辨率。您可以在创建指标索引时为其指定毫秒级时间戳分辨率，也可以编辑现有指标索引将其分辨率切换为毫秒级时间戳分辨率。

如果您使用的是 Splunk Cloud，请参阅《Splunk Cloud 用户手册》中的“管理 Splunk Cloud 索引”。如果您使用的是 Splunk Enterprise，请参阅《管理索引和索引群集》中的“创建自定义索引”。

内存和 mstats 搜索性能

一对 limits.conf 设置在 mstats 搜索的性能及其在搜索过程中所占用的 RAM 和磁盘内存量之间取得平衡。如果 mstats 一直完成得都很慢，则可以调整这些设置以提高其性能，但同时会增加搜索时的内存使用率，并可能因此导致搜索失败。

如果您有 Splunk Cloud，则需要提交支持工单，申请更改这些设置。

有关更多信息，请参阅《搜索手册》中的“内存和 stats 搜索性能”。

按字典顺序

基于用于编码计算机内存中的项目的值按字典顺序对这些项目进行排序。在 Splunk 软件中，几乎都是使用 UTF-8 进行编码，这是 ASCII 的超集。

- 数字排在字母前面。根据第一位数字对数字进行排序。例如数字 10、9、70、100，按照字典顺序排序为 10、100、70、9。
- 大写字母排在小写字母前面。
- 符号的排序标准不固定。有些符号排在数字值前面。有些符号排在字母前面或后面。

您可以指定覆盖字典顺序的自定义排序顺序。请参阅博客“向上排序”！自定义排序顺序

示例

1. 计算按时间分组的单个指标

返回 mymetricdata 指标索引中 aws.ec2.CPUUtilization 指标值的平均值。将结果分成 30 秒时间范围。

```
| mstats avg(aws.ec2.CPUUtilization) WHERE index=mymetricdata span=30s
```

2. 将指标与不同的指标名称相结合

返回 aws.ec2.CPUUtilization 指标和 os.cpu.utilization 指标的平均值。将结果按照主机进行分组，将结果分为 1 分钟时间范围。指标组合，并视作为单个指标系列。

```
| mstats avg(aws.ec2.CPUUtilization) avg(os.cpu.utilization) WHERE index=mymetricdata BY host span=1m
```

3. 使用 chart=t 模式来按前十大主机为指标事件计数绘制图表

按前十大主机返回 aws.ec2.CPUUtilization 指标数据点每天数量的图表。

```
| mstats chart=t count(aws.ec2.CPUUtilization) by host WHERE index=mymetricdata span=1d chart.limit=top10
```

4. 筛选维度值的结果并按其他维度的值拆分

返回含 host=foo 的所有计量的 aws.ec2.CPUUtilization 指标平均值，按 app 维度的值拆分结果。

```
| mstats avg(aws.ec2.CPUUtilization) WHERE host=foo BY app
```

5. 指定多个指标的多个聚合

返回常驻设置大小和虚拟内存大小的平均值和最大值。按 metric_name 对结果进行分组，并按 1 分钟这一时间跨度将它们分装到数据桶中

```
| mstats avg(os.mem.rss) AS "AverageRSS" max(os.mem.rss) AS "MaxRSS" avg(os.mem.vsz) AS "AverageVMS" max(os.mem.vsz) AS
```

```
"MaxVMS" WHERE index=mymetricdata BY metric_name span=1m
```

6. 使用下采样加快搜索速度，在所有默认指标索引中汇总一个指标

找到 aws.ec2.CPUUtilization 指标的中间值。搜索与您的角色相关的所有默认指标索引中的计量时不要包含索引筛选器。使用 every 来计算搜索所覆盖的每五分钟中的分钟中值，从而加快搜索速度。

```
| mstats median(aws.ec2.CPUUtilization) span=1m every=5m
```

7. 获取累计计数器的速度并将结果按时间序列分组

有关更多信息，请参阅《指标》中的“对指标时间序列执行统计计算”。

```
| mstats rate(spl.intr.resource_usage.PerProcess.data.elapsed) as data.elapsed where index=_metrics BY _timeseries | rename _timeseries AS timeseries
```

8. Stats-func-value 示例

使用 <stats-func-value> 语法获得 mymetricdata 索引中 aws.ec2.CPUUtilization 指标的所有测量值数量。

```
| mstats count(_value) WHERE metric_name=aws.ec2.CPUUtilization AND index=mymetricdata
```

另请参阅

相关信息

《指标》中的“指标简介”

multikv

描述

从表格形式的事件中提取字段值，比如 top、netstat 和 ps 等的结果。multikv 命令针对每个表格行新建一个事件，并从表格的标题行中指派字段名称。

multikv 设计来处理的数据类型的示例。

Name	Age	Occupation
Josh	42	SoftwareEngineer
Francine	35	CEO
Samantha	22	ProjectManager

关键属性如下：

- 每行文本代表一个概念性记录。
- 列对齐。
- 文本的第一行为列数据提供名称。

multikv 能将此表从一个事件转换为具有相关字段的三个事件。尽管有时能处理仅仅排好序的字段，但处理固定对齐的字段更容易。

通常的策略是确定标头、偏移和字段计数，然后确定后续行的哪个部分应该被包括在那些字段名称中。能处理单个事件中的多个表（如果 multitable=true），但是需要确保辅助表的标头行的名称为大写形式或 ALLCAPS。

标头行的自动检测将偏向于名称为 ALLCAPS 或大写形式的文本行。

如果您有 Splunk Cloud 并要使用此功能，请提交指定要定义的多键值提取的支持工单。

语法

```
multikv [conf=<stanza_name>] [<multikv-option>...]
```

可选参数

conf

语法：conf=<stanza_name>

描述：若您在 multikv.conf 中定义了字段提取，请使用本参数引用搜索中的段落。更多信息请参阅《管理员手册》中的 multikv.conf 配置文件引用。

<multikv-option>
语法: copyattrs=<bool> | fields <field-list> | filter <term-list> | forceheader=<int> | multitable=<bool>
| noheader=<bool> | rmorig=<bool>
描述: 用于从表格形式的事件中提取字段的选项。

multikv 选项的描述

copyattrs
语法: copyattrs=<bool>
描述: 若为 true, 则 multikv 将把原始事件中的所有字段复制到原始事件生成的各事件中。如果为 false, 则不会从原始事件中复制字段。这意味着事件没有 _time 字段, UI 不知道如何显示它们。
默认值: true

字段

语法: fields <field-list>
描述: 将通过 multikv 提取设置的字段限制到此列表中。忽略表中所有未包含在此列表中的字段。

过滤器

语法: filter <term-list>
描述: 若已指定, multikv 将跳过不含过滤器列表中至少一个字符串的表格行。允许带引号的表达式, 例如 "multiple words" 或 "trailing_space"。

forceheader

语法: forceheader=<int>
描述: 强制将给定行号 (以 1 为基数) 作为表的标题。计数中不包括空行。
默认值: multikv 命令将尝试自动确定标头行。

multitable

语法: multitable=<bool>
描述: 控制初始事件的单个 _raw 中是否可以存在多个表?
默认值: true

noheader

语法: noheader=<bool>
描述: 处理没有标头行识别的表。从第一行推断表的大小, 命名字段为 Column_1, Column_2, ... noheader=true 意味着 multitable=false。
默认值: false

rmorig

语法: rmorig=<bool>
描述: 如果为 true, 则在输出结果中不包括原始事件。如果为 false, 则在输出结果中会包括原始事件, 且每个原始事件在一批原始事件的生成结果后发出。
默认值: true

用法

multikv 命令属于可分配的流命令。请参阅 “命令类型”。

示例

示例 1: 如果 “COMMAND” 字段出现在包含 “splunkd” 的行中, 则提取该字段。

```
... | multikv fields COMMAND filter splunkd
```

示例 2: 提取 “pid” 和 “command” 字段。

```
... | multikv fields pid command
```

另请参阅

extract, kvform, rex, xmlkv

multisearch

描述

multisearch 命令是同时运行多个流搜索的生成命令。此命令需要至少两个子搜索, 同时每个子搜索仅允许流搜索。流搜索示例

包括使用以下命令的搜索：search、eval、where、fields 和 rex。更多信息，请参阅《[搜索手册](#)》中的“命令类型”。

语法

```
| multisearch <subsearch1> <subsearch2> <subsearch3> ...
```

必要参数

<subsearch>

语法："["search <logical-expression>"]"

描述：必须至少指定两个流搜索。有关 <logical-expression> 有效参数的详细信息，请参阅“[搜索命令](#)”。

更多信息请参阅《[搜索手册](#)》中的“关于子搜索”。

用法

multisearch 命令属于事件生成命令。请参阅“[命令类型](#)”。

生成命令使用前导管道符且应是搜索中的第一个命令。

子搜索处理和限制

使用 multisearch 命令时，每个子搜索的事件是交错的。因此，multisearch 命令不受子搜索限制的限制。

与 append 命令不同，multisearch 命令不先运行子搜索以完成。以下使用 append 命令的子搜索示例与使用 multisearch 命令的不同。

```
index=a | eval type = "foo" | append [search index=b | eval mytype = "bar"]
```

示例

示例 1：

搜索来自索引 a 和 b 的事件。使用 eval 命令把不同的字段添加到每个结果集。

```
| multisearch [search index=a | eval type = "foo"] [search index=b | eval mytype = "bar"]
```

另请参阅

[append](#), [join](#)

mvcombine

描述

选取一组除了指定字段不同，其他都相同的事件（其中包括单值），并将这些事件组合成单个事件。指定的字段变成包含组合事件中所有单值的多值字段。

mvcombine 命令不适用于内部字段。

请参阅《[知识管理器手册](#)》中的“使用默认字段”。

语法

```
mvcombine [<delim=>string] <field>
```

必要参数

field

语法：<field>

描述：要合并的字段名称（生成多值字段）。

可选参数

delim

语法: `delim=<string>`

描述: 定义字符串用作组合成多值字段的值的分隔符。例如，如果字段值为 "1"、"2" 和 "3"，`delim` 为 "，"，那么组合多值字段将为 "1";"2";"3"。

默认值: 一个空格 (" ")

要查看 `delim` 参数的输出，您必须紧接 `mvcombine` 命令使用 `nomv` 命令。请参阅“用法”

用法

`mvcombine` 命令是一个转换命令。请参阅“命令类型”。

您可以对多值字段使用评估函数和统计函数或返回多值字段。

`mvcombine` 命令接受一组输入结果并找到结果组，在结果组中，除了指定字段，字段值相同。所有这些结果合并成单个结果，在单个结果中，指定的字段现在是多值字段。

由于原始事件包含多个不同的字段，因此用 `fields` 命令削减可用字段集后再使用该命令通常是最有效的。此命令对于操作某些报表命令的结果也很有效。

指定分隔符

`mvcombine` 命令新建您指定的字段的多值版本和字段的单值版本。默认显示多值版本。

字段的单值版本是由空格或您用 `delim` 参数指定的分隔符分隔的平面字符串。

默认情况下，字段的多值版本显示在结果中。要显示用分隔符隔开的单值版本，请将 `| nomv` 命令添加到搜索的结尾。例如，`... | mvcombine delim= " " host | nomv host`。

调查搜索结果的部分模式倾向于单值呈现方式，例如在 UI 中导出到 CSV 或使用 `splunk search "... -output csv` 运行命令进行搜索。部分无法识别 `multivalue` 的命令可能也会使用该单值。

访问搜索结果大部分倾向于多值呈现方式，例如在 UI 中查看结果或导出到 JSON，从含 `splunk search "... -output json` 的命令行搜索中请求 JSON 或从 REST API 中请求 XML。所选择的分隔符对于这些表单没有影响。

将多值字段转换为单值字段的其他方法

如果您的主要目标是将多值字段转换为单值字段，`mvcombine` 可能不是您的最佳选择。`mvcombine` 主要用于创建新的多值字段。相反，请尝试使用 `nomv` 命令或 `mvjoin eval` 函数。

转换选项	描述	相关信息
<code>nomv</code> 命令	用于简单的多值字段到单值字段的转换。在搜索结果中提供多值字段的名称， <code>nomv</code> 会将字段的每个实例转换为单值字段。	<code>nomv</code>
<code>mvjoin eval</code> 函数	当您要执行多值字段到单值字段的转换时使用，其中以前的多值由您提供的分隔符分隔。例如，从包含值 1、2、3、4、5 的多值字段开始。可以使用 <code>mvjoin</code> 将多值字段转换为以 OR 作为分隔符的单值字段。该字段的新单值是 1 OR 2 OR 3 OR 4 OR 5。	多值 <code>eval</code> 函数

示例

1. 新建多值字段

此示例将使用搜索教程中的示例数据。要亲自尝试此示例，请从“将教程数据导入到 Splunk”下载数据集并根据搜索教程中的说明上载数据。

要了解 `mvcombine` 如何运作，我们来探索数据。

1. 时间范围设为所有时间。
2. 运行以下搜索：

```
index=* | stats max(bytes) AS max, min(bytes) AS min BY host
```

结果显示针对以 `www` 开头的主机，`max` 和 `min` 字段有重复条目。其他主机 `max` 和 `min` 字段不显示任何结果。

新搜索

另存为 ▾ 关闭

index=* | stats max(bytes) AS max, min(bytes) AS min BY host

所有时间 ▾

✓ 179,403 个事件 (18/06/05 13:24:49.000 之前) 无事件采样 ▾ 任务 ▾ || ⌂ ⌃ ⌁ ⌄ ⌅ ⌆ ⌇ ⌈ ⌉ 智能模式 ▾

事件 模式 统计信息 (9) 可视化

每页 20 个 ▾ 格式 预览 ▾

host	max	min
127.0.0.1		
UTC		
debianSplunk		
mailsv		
syslogd		
vendor_sales		
www1	4000	200
www2	4000	200
www3	4000	200

- 要从结果中移除其他主机，请修改搜索以将 host=www* 添加至搜索条件。

```
index=* host=www* | stats max(bytes) AS max, min(bytes) AS min BY host
```

因为 max 和 min 列中的值包含完全相同的值，您可以使用 mvcombine 将主机值与多值结果结合起来。

- 将 | mvcombine host 添加到您的搜索并再次运行搜索。

```
index=* host=www* | stats max(bytes) AS max, min(bytes) AS min BY host | mvcombine host
```

会返回一行，而不是三行。主机字段现在是多值字段。

新搜索

另存为 ▾ 关闭

index=* host=www* | stats max(bytes) AS max, min(bytes) AS min BY host | mvcombine host

所有时间 ▾

✓ 69,791 个事件 (18/06/05 13:25:56.000 之前) 无事件采样 ▾ 任务 ▾ || ⌂ ⌃ ⌁ ⌄ ⌅ ⌆ ⌇ ⌈ ⌉ 智能模式 ▾

事件 模式 **统计信息 (1)** 可视化

每页 20 个 ▾ 格式 预览 ▾

host	max	min
www1	4000	200
www2		
www3		

2. 返回用分隔符隔开的值

正如在使用情况部分所介绍，默认情况下，结果的分隔版本不在输出中返回。要返回用分隔符隔开的结果，您必须返回字段的单值字符串版本。

将 nomv 命令添加到您的搜索。例如：

```
index=* host=www* | stats max(bytes) AS max, min(bytes) AS min BY host | mvcombine delim="," host | nomv host
```

返回的搜索结果显示在以下表格中。

host	max	min
www1, www2, www3	4000	200

要返回在每个逗号后面有空格的结果，请指定 `delim=" , "`。

示例 3:

在多值事件中：

```
sourcetype="WMI:WinEventLog:Security" | fields EventCode, Category,RecordNumber | mvcombine delim="," RecordNumber | nomv RecordNumber
```

示例 4:

用冒号分隔符合并 "foo" 的值。

```
... | mvcombine delim=":" foo
```

另请参阅

命令:

makemv
mvexpand
nomv

函数:

多值 eval 函数
多值统计和 chart 函数
拆分

mvexpand

描述

将多值字段的值扩展到多值字段中每个值的某个事件中。mvexpand 命令将为所有结果的每个多值字段新建一个结果。

mvexpand 命令不适用于内部字段。

请参阅《知识管理器手册》中的“使用默认字段”。

语法

```
mvexpand <field> [limit=<int>]
```

必要参数

字段

语法: <field>
描述: 多值字段的名称。

可选参数

limit

语法: limit=<int>
描述: 指定用于每个输入事件的 <field> 值的数量。
默认值: 0, 即无限制

用法

mvexpand 命令属于可分配的流命令。请参阅“命令类型”。

您可以对多值字段使用评估函数和统计函数或返回多值字段。

限制

mvexpand 命令在扩展一批结果时获准使用的 RAM 总量存在限制。默认情况下，限制为 500MB。结果的输入块通常会是 maxresultrows 或较小规模，同时所有那些结果的扩展都驻留在内存中。所需的总内存为将平均结果规模乘以结果块的数量，再乘以扩展的多值字段的平均大小。

如果此尝试超过任何区块上配置的最大值，则该区块将被截断，并发出警告消息。如果您有 Splunk Enterprise，您可以通过

编辑 limits.conf 文件中的 max_mem_usage_mb 设置调整限制。

前提条件

- 只有具有文件系统访问权限的用户，如系统管理员，才能使用配置文件增加 maxresultrows 和 max_mem_usage_mb 的值。
- 请参阅 Splunk Enterprise《管理员手册》中的“如何编辑配置文件”了解具体步骤。
- 您可以有几个具有相同名称的配置文件，分散在默认目录、本地目录和应用目录中。请参阅 Splunk Enterprise《管理员手册》中“在何处放置（或查找）已修改的配置文件”。

不要更改或复制默认目录中的配置文件。默认目录中的文件必须保持原样并位于其原始位置。更改本地目录中的文件。

如果您有 Splunk Cloud 并遇到由于该限制引起的问题，请提交支持工单。

示例

示例 1:

为多值字段 "foo" 的每个值新建事件。

```
... | mvexpand foo
```

示例 2:

为多值字段 "foo" 的前 100 个值新建事件。

```
... | mvexpand foo limit=100
```

示例 3:

mvexpand 命令仅对一个多值字段有效。此示例将阐述如何将具有多个多值字段的一个事件扩展到每个字段值对应一个事件。例如，对于以下事件 (sourcetype=data)：

```
2018-04-01 00:11:23 a=22 b=21 a=23 b=32 a=51 b=24  
2018-04-01 00:11:22 a=1 b=2 a=2 b=3 a=5 b=2
```

首先，使用 rex 命令提取 a 和 b 的字段值。然后，使用 eval 命令和 mvzip 函数基于 a 和 b 的值新建一个字段。

```
source="mvexpandData.csv" | rex field=_raw "a=(?<a>\d+)" max_match=5 | rex field=_raw "b=(?<b>\d+)" max_match=5 | eval fields = mvzip(a,b) | table _time fields
```

统计选项卡中显示的结果如下所示：

_time	字段
2018-04-01 00:11:23	22, 21 23, 32 51, 24
2018-04-01 00:11:22	1, 2 2, 3 5, 2

在新的字段字段上使用 mvexpand 命令和 rex 命令以新建事件并提取 alpha 和 beta 值：

```
source="mvexpandData.csv" | rex field=_raw "a=(?<a>\d+)" max_match=5 | rex field=_raw "b=(?<b>\d+)" max_match=5 | eval fields = mvzip(a,b) | mvexpand fields | rex field=fields "(?<alpha>\d+),(?<beta>\d+)" | table _time alpha beta
```

使用表命令在结果表中只显示 _time、alpha 和 beta 字段。

统计选项卡中显示的结果如下所示：

_time	alpha	beta
-------	-------	------

2018-04-01 00:11:23	23	32
2018-04-01 00:11:23	51	24
2018-04-01 00:11:22	1	2
2018-04-01 00:11:22	2	3
2018-04-01 00:11:22	5	2

(感谢 Splunk 用户 Duncan 提供此示例。)

另请参阅

命令:

makemv
mvcombine
nomv

函数:

多值 eval 函数
多值统计和 chart 函数
拆分

nomv

描述

将指定的多值字段的值转换为单个值。使用新行 "\n" 分隔符分隔值。

覆盖在 fields.conf 文件中设置的多值字段的配置。

语法

nomv <field>

必要参数

字段

语法: <field>

描述: 多值字段的名称。

用法

nomv 命令属于可分配的流命令。请参阅“命令类型”。

您可以对多值字段使用评估函数和统计函数或返回多值字段。

示例

示例 1:

对于 sendmail 事件，将 senders 字段的值合并成单个值。显示前 10 个值。

```
eventtype="sendmail" | nomv senders | top senders
```

另请参阅

命令:

makemv
mvcombine
mvexpand
convert

函数:

多值 eval 函数

多值统计和 chart 函数
拆分

outlier

描述

此命令用于删除离群值，而不检测离群值。将删除或截去所选字段中的无关数字值。若未指定任何字段，outlier 命令将尝试处理所有字段。

筛选是基于四分位差 (IQR) 进行的，而 IQR 是基于数字字段的第 25 个百分点值与第 75 个百分点值之间的差值计算的。若事件中某字段的值小于 (25th percentile) - param*IQR 或大于 (75th percentile) + param*IQR，将根据 action 参数转换该字段或删除该事件。

若要识别离群值并新建离群值告警，请参阅《搜索手册》中的查找和移除离群值。

语法

```
outlier <outlier-options>... [<field-list>]
```

可选参数

<outlier-options>

语法: <action> | <mark> | <param> | <uselower>

描述: 离群值选项。

<field-list>

语法: <field> ...

描述: 字段名称列表，以空格作分隔。

离群值选项

<action>

语法: action=remove | transform

描述: 指定如何处理离群值。remove 选项将删除包含无关数字值的事件。transform 选项将把无关的值截成离群值的阈值。若 action=transform 且 mark=true，则在各值前加 "000"。

缩写: remove 动作可以缩写成 rm。transform 动作可以缩写成 tf。

默认: 转换

<mark>

语法: mark=<bool>

描述: 若 action=transform 且 mark=true，则在各无关的值前加 "000"。若 action=remove，mark 参数将无效。

默认值: false

<param>

语法: param=<num>

描述: 用来控制离群值检测阈值的参数。离群值的定义为：超出 param 与四分位差 (IQR) 乘积的数字值。

默认值: 2.5

<uselower>

语法: uselower=<bool>

描述: 除了以上之外，控制是否查找值低于中值的离群值。

默认值: false

示例

示例 1：对于 webserver 事件的 timechart，转换无关的平均 CPU 值。

```
404 host="webserver" | timechart avg(cpu_seconds) by host | outlier action=tf
```

示例 2：删除所有无关的数字值。

```
... | outlier
```

另请参阅

anomalies, anomalousvalue, cluster, kmeans

查找和移除离群值

outputcsv

描述

如果您有 Splunk Enterprise，则该命令把搜索结果保存到 \$SPLUNK_HOME/var/run/splunk/csv 目录内本地搜索头上指定的 CSV 文件中。更新到 \$SPLUNK_HOME/var/run/*.csv，使用 outputcsv 命令不会在群集中复制。

如果您有 Splunk Cloud，您不能使用此命令。您可以：

- 使用 Splunk Web 导出搜索结果请参阅《[搜索手册](#)》中的“使用 Splunk Web 导出数据”。
- 使用 REST API 导出搜索结果。请参阅《[搜索手册](#)》中的“使用 REST API 导出数据”。
- 创建包含 CSV 文件作为电子邮件附件的告警操作。请参阅《[告警手册](#)》中的“电子邮件通知操作”。

语法

```
outputcsv [append=<bool>] [create_empty=<bool>] [override_if_empty=<bool>] [dispatch=<bool>] [usexml=<bool>]  
[singlefile=<bool>] [<filename>]
```

可选参数

append

语法：append=<bool>

描述：若 append 为 true，该命令将尝试附加一个现有的 CSV 文件（若存在）。若 CSV 文件不存在，将新建一个文件。若已有一个带 csv 标题的现有文件，命令仅发出该标题所引用的字段。该命令无法附加到 .gz 文件。

默认值：false

create_empty

语法：create_empty=<bool>

描述：如果设置为 true 且没有结果，则新建一个长度为 0 的文件。设置为 false 且没有结果，则没有新建的文件。如文件预先存在，则删除文件。

默认值：false

dispatch

语法：dispatch=<bool>

描述：若设置为 true，则引用 \$SPLUNK_HOME/var/run/splunk/dispatch/<job id>/ 中任务目录内的文件。

filename

语法：<filename>

描述：指定要将搜索结果写入其中的 CSV 文件的名称。此文件应位于 \$SPLUNK_HOME/var/run/splunk/csv 中。文件名中不允许使用目录分隔符。若未指定文件名，该命令将把每个结果的内容以 CSV 行的形式重新写入 _xml 字段中。否则，命令写入文件。如果文件名没有文件拓展名，.csv 文件拓展名附加到文件名。

override_if_empty

语法：override_if_empty=<bool>

描述：如果 override_if_empty=true 且没有结果传递到输出文件，将删除现有的输出文件，如果 override_if_empty=false 且没有结果传递到输出文件，命令不会删除现有的输出文件。

默认值：true

singlefile

语法：singlefile=<bool>

描述：若 singlefile 设置为 true 且输出跨多个文件，则将其压缩成一个文件。

默认值：true

usexml

语法：usexml=<bool>

描述：若无文件名，则指定是否将 csv 输出编码为 XML。从 UI 调用 outputcsv 时不可使用该选项。

用法

可保存为 CSV 文件的结果数量没有限制。

内部字段和 outputcsv 命令

使用 outputcsv 命令时，有自动添加到 CSV 文件的内部字段。添加到 CSV 文件中的输出的内部字段是：

- `_raw`
- `_time`
- `_indextime`
- `_serial`
- `_sourcetype`
- `_subsecond`

要将内部字段从输出中排除，请使用 `fields` 命令并指定您想要排除的字段。例如：

```
... | fields - _indextime _sourcetype _subsecond _serial | outputcsv MyTestCsvFile
```

多值字段

`outputcsv` 命令将多值字段中的值合并为以单个空格分隔的值。

分布式部署

`outputcsv` 命令和搜索头合并以及搜索头群集化不兼容。

命令将 `*.csv` 文件保存在 `$SPLUNK_HOME/var/run/splunk/` 目录中的本地搜索头上。`*.csv` 文件不会在其他搜索头中复制。

示例

1. 将搜索结果输出到 CSV 文件

将搜索结果输出至 `mysearch.csv` 文件。如果您不在搜索中指定拓展名，文件名称中自动添加 `csv` 文件拓展名。

```
... | outputcsv mysearch
```

2. 在文件名中添加动态时间戳

您可以使用子搜索在文件名中添加时间戳。

```
... | outputcsv [stats count | eval search=strftime(now(), "mysearch-%y%m%d-%H%M%S.csv")]
```

3. 将内部字段从输出 CSV 文件中排除。

将不想要的内部字段从输出 CSV 文件中排除。在此示例中，要排除的字段是 `_indextime`、`_sourcetype`、`_subsecond` 和 `_serial`。

```
index=_internal sourcetype="splunkd" | head 5 | fields _raw _time | fields - _indextime _sourcetype _subsecond _serial | outputcsv MyTestCsvfile
```

4. 如果没有返回搜索结果，不要删除 CSV 文件。

如果结果从搜索中返回，将搜索结果输出到 `mysearch.csv` 文件中。如果没有返回结果，不要删除 `mysearch.csv` 文件。

```
... | outputcsv mysearch.csv override_if_empty=false
```

另请参阅

`inputcsv`

outputlookup

描述

将搜索结果写入您指定的静态查找表或 KV 存储集合中。

语法

要求的语法以粗体表示。

```
| outputlookup
[append=<bool>]
[create_empty=<bool>]
```

```
[override_if_empty=<bool>]  
[max=<int>]  
[key_field=<field>]  
[create_inapp=<bool>]  
[create_context=<string>]  
[output_format=<string>]  
<filename> | <tablename>
```

必要参数

您必须指定以下必需参数之一： filename 或 tablename。

filename

语法：<string>

描述：查找文件的名称。该文件必须以 .csv 或 .csv.gz 结尾。

tablename

语法：<string>

描述：transforms.conf 中的段落名称所指定的查找表的名称。查找表能配置为任意查找类型（CSV、外部或 KV 存储）。

可选参数

append

语法：append=<bool>

描述：默认 append=false 设置将搜索结果写入 .csv 文件或 KV 存储集合。不在当前搜索结果中的列将从文件中删除。如果设置为 true，请尝试将搜索结果附加到现有的 .csv 文件或 KV 存储集合。否则将会新建一个文件。若 .csv 文件已存在，outputlookup 命令仅写入出现于现有 .csv 文件中的字段。与 append=true 一起运行的 outputlookup 搜索可能导致查找表格或集合只是部分更新的情况。这意味着该查找表或集合上后续的 lookup 或 inputlookup 搜索可能返回新旧掺杂的数据。outputlookup 命令无法附加到 .gz 文件。

默认值：false

create_context

语法：create_context= app | user | system

描述：指定查找表文件的创建位置。如果在搜索中使用了两个参数，则忽略这两个参数，转而使用 creatinapp 参数。有关详细信息，请参阅“用法”。

默认值：app

create_empty

语法：create_empty=<bool>

描述：如果设置为 true 且没有结果，则新建一个长度为 0 的文件。设置为 false 且没有结果，则没有新建的文件。如文件预先存在，则删除文件。

例如，假设有一个名为 "test" 的系统级查找，并且在 "test.csv" 中定义了此查找。还有一个具有相同名称的应用程序级查找。如果应用程序用空文件 create_empty=true 覆盖其应用程序目录中的 "test.csv"，则应用程序级查找就会将查找视为空来执行相关的行为。但是，如果在应用程序级别根本没有文件 create_empty=false，则将使用系统级别的查找文件。

默认值：false

createinapp

语法：createinapp=<bool>

描述：指定查找表文件是在系统目录中还是在当前应用上下文的查找目录中创建。如果在搜索中使用了这两个参数，则覆盖 create_context 参数。有关详细信息，请参阅“用法”。

默认值：true

key_field

语法：key_field=<field>

描述：对于基于 KV 存储的查找，会使用指定的字段名称作为值键并替换此值。outputlookup 搜索使用 key_field 参数，因此可能会导致查找表或集合仅部分更新。针对该集合的后续 lookup 或 inputlookup 搜索可能会同时返回旧数据和新数据。部分更新仅在并发搜索时发生，一个使用 outputlookup 命令，并使用 inputlookup 命令搜索。可能会出现 inputlookup（当 outputlookup 仍在更新某些记录时）。

max

语法：max=<int>

描述：要输出的行数。

默认值：无限制

output_format

语法：output_format=splunk_sv_csv | splunk_mv_csv

描述：控制查找的输出数据格式。要将多值字段输出到查找表文件时，请使用 output_format=splunk_mv_csv，然后使用 inputlookup 命令将这些字段读回到 Splunk。默认情况下，splunk_sv_csv 会输出一个 CSV 文件，其中不包含

`_mv_<fieldname>` 字段。
默认值: `splunk_sv_csv`

`override_if_empty`

语法: `override_if_empty=<bool>`

描述: 如果 `override_if_empty=true` 且没有结果传递到输出文件, 将删除现有的输出文件, 如果 `override_if_empty=false` 且没有结果传递到输出文件, 命令不会删除现有的输出文件。

默认值: `true`

用法

查找表必须为 CSV 或 GZ 文件, 或是一个使用 `transforms.conf` 中的查找表配置指定的表格名称。查找表可指 KV 存储集合或 CSV 查找。`outputlookup` 命令不能用于外部查找。

确定查找表文件的创建位置

对于 CSV 查找, `outputlookup` 会为搜索结果创建一个查找表文件。`outputlookup` 可以在三个位置放置它创建的文件:

- 系统查找目录: `$SPLUNK_HOME/etc/system/local/lookups`
- 当前应用上下文的查找目录: `$SPLUNK_HOME/etc/apps/<app>/lookups`
- 运行搜索的用户的基于应用的查找目录: `etc/users/<user>/<app>/lookups`

您可以使用 `createinapp` 或 `create_context` 参数来确定 `outputlookup` 为给定搜索创建查找表的位置。如果您尝试在同一个搜索中使用这两个参数, 则 `createinapp` 参数会覆盖 `create_context` 参数。

如果您在搜索中没有使用这两个参数, `limits.conf` 中的 `create_context` 设置将确定 `outputlookup` 创建查找表文件的位置。如果运行搜索时有应用上下文, 则此设置默认为 `app`; 如果运行搜索时没有应用上下文, 则此设置默认为 `system`。

要让 `outputlookup` 在系统查找目录中创建查找表文件, 请设置 `createinapp=false` 或设置 `create_context=system`。或者, 如果您在运行搜索时没有应用上下文, 则不要在搜索中使用这两个参数, 而是依靠 `create_context` 的 `limits.conf` 版本将查找表文件放在系统目录中。仅当 `limits.conf` 中的 `create_context` 设置尚未设置为 `user` 时, 最后一种方法才有效。

要让 `outputlookup` 在当前应用上下文的查找目录中创建查找表文件, 请设置 `createinapp=true` 或设置 `create_context=app`。或者, 如果您在运行搜索时确实有应用上下文, 则不要在搜索中使用这两个参数, 而是依靠 `create_context` 的 `limits.conf` 版本将查找表文件放在应用目录中。仅当 `limits.conf` 中的 `create_context` 设置尚未设置为 `user` 时, 最后一种方法才有效。

要让 `outputlookup` 在查找目录中为运行搜索的用户创建查找表文件, 请设置 `create_context=user`。或者, 如果您希望所有 `outputlookup` 搜索默认在用户查找目录中创建查找表文件, 您可以在 `limits.conf` 中设置 `create_context=user`。如果在搜索中使用了 `createinapp` 和 `create_context` 参数, 则可以覆盖此设置。

如果查找表文件已存在于其写入位置, 则将使用 `outputlookup` 搜索的结果覆盖该文件的现有版本。

使用 `check_permission` 限制对查找表文件的写入权限

有关 CSV 查找的权限, 使用 `transforms.conf` 中的 `check_permission` 字段和 `limits.conf` 中的 `outputlookup_check_permission`, 以限制使用 `outputlookup` 命令时, 具有适当权限的用户的写入访问权。`check_permission` 和 `outputlookup_check_permission` 都默认为 `false`。为 Splunk 软件设为 `true`, 可验证用户的查找权限设置。您可以在每个查找文件的 `.meta` 文件或设置 > 查找 > 查找表文件中更改查找表文件权限。默认情况下, 只有具有管理员或高级角色的用户才能向共享的 CSV 查找文件写入内容。

有关新建查找的更多信息, 请参阅知识管理器手册中的“关于查找”。

更多有关“应用键值存储集合”的信息, 请参阅管理员手册中的“关于 KV 存储”。

附加结果

假设您现在有一份 CSV 文件, 其中包含 A、D 和 J 列。搜索的结果是 A、C 和 J 列。如果您运行 `outputlookup append=false` 的搜索, 那么 A、C 和 J 列写入 CSV 文件中。不会保留 D 列。

如果您运行 `outputlookup append=true` 的搜索, 那么将只保留文件中当前存在的列。在此示例中, A 和 J 列将写入 CSV 文件。C 列将丢失, 因为它并未存在于 CSV 文件中。D 列将保留。

您可以在运行搜索之前使用 `eval` 命令向 CSV 文件添加列, 绕过此问题。例如: 如果您的 CSV 文件名为 `foo`, 您可以进行下列操作:

```
| inputlookup foo | eval c=null | outputlookup foo append=false ....
```

然后运行搜索, 并为您想要保留的列将结果导入到 `fields` 命令。

```
... | fields A C J | outputlookup append=true foo
```

多值字段

当您输出到静态查找表时，`outputlookup` 命令将多值字段中的值合并为以单个空格分隔的值。这不适用于 KV 存储集合。

示例

1. 使用 `transforms.conf` 文件中的设置写入查找表

写入到 `usertogroup` 查找表，该表是在 `transforms.conf` 文件中指定的。

```
| outputlookup usertogroup
```

2. 写入特定系统或应用程序目录中的查找文件

写入到 `users.csv` 查找文件，该文件位于 `$SPLUNK_HOME/etc/system/lookups` 或 `$SPLUNK_HOME/etc/apps/*/lookups` 下。

```
| outputlookup users.csv
```

3. 指定如果未返回任何结果则不覆盖查找文件

如果返回结果，在 `$SPLUNK_HOME/etc/system/lookups` 或 `$SPLUNK_HOME/etc/apps/*/lookups` 下写入 `users.csv` 查找文件。如果没有返回结果，不要删除 `users.csv` 文件。

```
| outputlookup users.csv override_if_empty=false
```

4. 写入 KV 存储集合

将 Shalimar 餐馆的食品检查事件写入到名为 `kvstorecoll` 的 KV 存储集合中。在名为 `kvstorecoll_lookup` 的查找表中引用该集合。

```
index=sf_food_health sourcetype=sf_food_inspections name="SHALIMAR RESTAURANT" | outputlookup kvstorecoll_lookup
```

5. 从 CSV 文件写入 KV 存储集合

把 CSV 文件的内容写入 KV 存储集合 `kvstorecoll` 中，执行此操作需使用查找表 `kvstorecoll_lookup`。这需要使用 `inputlookup` 和 `outputlookup` 两个命令。

```
| inputlookup customers.csv | outputlookup kvstorecoll_lookup
```

6. 为单个 KV 存储集合记录更新字段值

为单个 KV 存储集合记录更新字段值。这要求您使用 `inputlookup`、`outputlookup` 和 `eval` 命令。记录由其内部关键 ID (`_key` 字段) 的值指示，并用新的客户名称和客户所在城市更新记录。记录属于 KV 存储集合 `kvstorecoll`，通过查找表 `kvstorecoll_lookup` 可以访问该集合。

```
| inputlookup kvstorecoll_lookup | search _key=544948df3ec32d7a4c1d9755 | eval CustName="Vanya Patel" | eval CustCity="Springfield" | outputlookup kvstorecoll_lookup append=True key_field=_key
```

若需了解如何获取 KV 存储集合中记录的内部关键 ID 值，请参阅 `inputlookup` 命令的示例 5。

另请参阅

命令

```
collect  
inputlookup  
lookup  
inputcsv  
mcollect  
meventcollect  
outputcsv  
outputtext
```

outputtext

描述

将 `_raw` 字段的内容输出到 `_xml` 字段。

将把 `outputtext` 命令新建为内部机制，来呈现输出的事件文本。

语法

```
outputtext [usexml=<bool>]
```

可选参数

usexml

语法: `usexml=<bool>`

描述: 如果设置为 `true`, 则 `_xml` 中 `_raw` 字段的副本会转义为 XML。如果 `usexml` 设置为 `false`, `_xml` 字段就与 `_raw` 一模一样。

默认值: `true`

用法

`outputtext` 命令属于报表命令。

`outputtext` 命令将所有搜索结果写入搜索头。在 Splunk Web 中，结果将显示在“统计”选项卡中。

示例

1. 将 `_raw` 字段输出到转义的 XML 中

将当前搜索的 `"_raw"` 字段输出到 `"_xml"` 中。

```
... | outputtext
```

另请参阅

`outputcsv`

overlap

注意: 我们不建议使用 `overlap` 命令填充或回填摘要索引。Splunk Enterprise 提供一个名为 `fill_summary_index.py` 的脚本，该脚本将回填您的索引或填充摘要索引间隙。如果您有 Splunk Cloud 并需要回填，请打开支持工单，并指定所需的时间范围、应用、搜索名称、用户和任何其他详情，这样 Splunk 支持就可以回填所需数据。有关更多信息，请参阅《知识管理器手册》中的“管理摘要索引间隙”。

描述

在摘要索引中查找时间重叠的事件，或者查找时间间隙，在该间隙内，计划的保存的搜索可能有缺失的事件。

- 如果查找间隙，则在间隙期间内运行搜索，并使用 “`| collect`” 为结果新建摘要索引。
- 如果查找重叠事件，则使用搜索语言从摘要索引中手动删除重叠事件。

`overlap` 命令将调用外部 Python 脚本 `$SPLUNK_HOME/etc/apps/search/bin/sumindexoverlap.py`。此脚本会接收来自摘要索引的输入事件，并在 `'info_search_name'` 相同而 `'info_search_id'` 不同的事件之间查找时间重叠和间隙。

重要提示: 输入事件应具有以下字段: `'info_min_time'` (含)、`'info_max_time'` (不含)、`'info_search_id'` 和 `'info_search_name'` 字段。若该索引包含原始事件 (`_raw`)，`overlap` 命令将失效。反之，该索引应包含 `chart`、`stats` 和 `timechart` 结果等事件。

语法

```
overlap
```

示例

示例 1:

在 "summary" 索引中查找重叠事件。

index=summary | overlap

另请参阅

collect, sistats, sitop, sirare, sichart, sitimechart

pivot

描述

pivot 命令让简单的数据透视表操作非常简单，但是对于更为复杂的数据透视表操作来说比较复杂。基本上，该命令是 stats 和 xyseries 命令的包装。

pivot 命令不会添加新行为，但是如果您已经非常熟悉数据透视表的运作，则使用起来会更简单。请参阅[《数据透视表手册》](#)。同时，请阅读如何在数据透视表中打开非转换搜索的相关内容。

对特定数据模型对象运行数据透视表搜索。这需要大量输入：数据模型、数据模型对象及数据透视表元素。

语法

| pivot <datamodel-name> <object-name> <pivot-element>

必要参数

datamodel-name

语法：<string>

描述：要搜索数据模型的名称。

objectname

语法：<string>

描述：要搜索数据模型对象的名称。

数据透视表元素

语法：(<cellvalue>)* (<SPLITROW <rowvalue>)* (<SPLITCOL colvalue [options]>)* (<FILTER <filter expression>>)* (<LIMIT <limit expression>>)* (<ROWSUMMARY <true | false>>)* (<COLSUMMARY <true | false>>)* (<SHOWOTHER <true | false>>)* (<NUMCOLUMNS <num>>)* (<rowsort [options]>)*

描述：使用数据透视表元素定义您的数据透视表表格或图表。数据透视表元素包括单元格值、拆分行、拆分列、筛选、限制，行和列格式以及行排序选项。单元格值始终优先。后跟可交错的拆分行和拆分列，例如：avg(val), SPLITCOL foo, SPLITROW bar, SPLITCOL baz。

单元格值

<cellvalue>

语法：<function>(<fieldname>) [AS <label>]

描述：定义单元格的值并选择对其进行重命名。这里，`label` 是报表中单元格的名称。

允许的函数的设置取决于 `fieldname` 的数据类型：

- 字符串：list、values、first、last、count 和 distinct_count (dc)
- 数字：sum、count、avg、max、min、stdev、list 和 values
- 时间戳：duration、earliest、latest、list 和 values
- 对象或子计数：count

行拆分元素的描述

SPLITROW <rowvalue>

语法：SPLITROW <field> [AS <label>] [RANGE start=<value> end=<value> max=<value> size=<value>] [PERIOD (auto | year | month | day | hour | minute | second)] [TRUELABEL <label>] [FALSELABEL <label>]

描述：您可以在每个 SPLITROW 中指定一个或多个选项。这些选项可任意排序。您可以用 "AS <label>" 重命名 <field>，其中 "`label`" 为报表中行的名称。

其他选项取决于指定 <field> 的数据类型：

- RANGE 仅适用于数字。您不需要指定所有选项 (start、end、max 和 size)。
- PERIOD 仅适用于时间戳。使用它指定数据桶的周期。

- TRUELABEL 仅适用于布尔。使用它指定 true 值的标签。
- FALSELABEL 仅适用于布尔。使用它指定 false 值的标签。

列拆分元素的描述

`SPLITCOL colvalue <options>`

语法: `fieldname [RANGE start=<value> end=<value> max=<value> size=<value>] [PERIOD (auto | year | month | day | hour | minute | second)] [TRUELABEL <label>] [FALSELABEL <label>]`

描述: 在每个 SPLITCOL 中, 您可以选择不包含这些选项、包含一些或所有这些选项。它们可能以任何顺序显示。

其他选项取决于指定字段 (`fieldname`) 的数据类型:

- RANGE 仅适用于数字。不需要指定所有的选项 (开始、结束、最大和大小)。
- PERIOD 仅适用于时间戳。使用它指定数据桶的周期。
- TRUELABEL 仅适用于布尔。使用它指定 true 值的标签。
- FALSELABEL 仅适用于布尔。使用它指定 false 值的标签。

筛选元素的描述

`Filter <filter expression>`

语法: `<fieldname> <comparison-operator> <value>`

描述: 用于识别字段中各值的表达式。您使用的比较运算符取决于字段值的类型。

- 字符串: `is`、`contains`、`in`、`isNot`、`doesNotContain`、`startsWith`、`endsWith`、`isNull`、`isNotNull`

例如: `... filter fname in (value1, value2, ...)`

- `ipv4`: `is`、`contains`、`isNot`、`doesNotContain`、`startsWith`、`isNull`、`isNotNull`
- 数字: `=`、`!=`、`<`、`<=`、`>`、`>=`、`isNull`、`isNotNull`
- 布尔: `is`、`isNull`、`isNotNull`

限制元素的描述

`Limit <limit expression>`

语法: `LIMIT <fieldname> BY <limittype> <number> <stats-function>(<fieldname>)`

描述: 用于限制数据透视表中的元素数量。`limittype` 参数将指定放置限制的位置。有效的值为 `top` 或 `bottom`。`number` 参数必须为正整数。可以使用任何 `stats` 函数, 例如 `min`、`max`、`avg` 和 `sum`。

示例: `LIMIT foo BY TOP 10 avg(bar)`

用法

`pivot` 命令属于报表生成命令。请参阅“命令类型”。

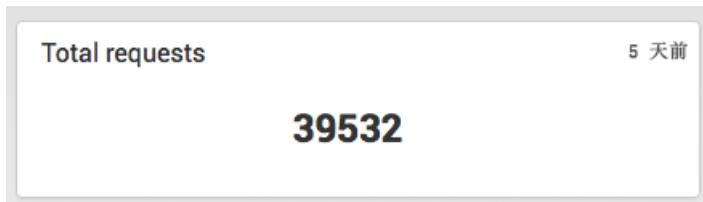
生成命令使用前导管道符且应是搜索中的第一个命令。

示例

示例 1: 本命令将计算“教程”数据模型中的“HTTP Requests”对象中的事件数量。

```
| pivot Tutorial HTTP_requests count(HTTP_requests) AS "Count of HTTP requests"
```

这可以格式化为仪表板面板中的单个值报表:



示例 2: 使用教程数据模型, 为每个主机的“HTTP Requests”计数新建数据透视表表格。

```
| pivot Tutorial HTTP_requests count(HTTP_requests) AS "Count" SPLITROW host AS "Server" SORT 100 host
```

Total requests by server		5 天前
Server	Count	
www1	13628	
www2	12912	
www3	12992	

另请参阅

`datamodel`, `stats`, `xseries`

predict

描述

`predict` 命令预示一个或多个时间系列数据的值。该命令还可以填充时间系列中缺失的数据，并为接下来的几个时间步骤提供预测。

`predict` 命令为所有估算值提供置信区间。此命令为该时间系列中的每个事件添加一个预测值，以及第 95 个上百分范围和下百分范围。请参阅本主题中的用法一节。

语法

`predict <field-list> [AS <newfield>] [<predict_options>]`

必要参数

`<field-list>`

语法: `<field>...`

描述: 要预测的变量的字段名称。可以指定一个或多个字段。

可选参数

`<newfield>`

语法: `<string>`

描述: 重命名 `<field-list>` 中指定的字段。无需重命名 `<field-list>` 中指定的每个字段。但是，您必须为每一个要重命名的字段指定分隔的 `AS <newfield>` 子句。

`<predict_options>`

语法: `algorithm=<algorithm_name> | correlate_field=<field> | future_timespan=<number> | holdback=<number> | period=<number> | suppress=<bool> | lowerXX=<field> | upperYY=<field>`

描述: 您可以指定的、用来控制预测值的选项。您可以指定一个或多个选项，排序不限。各选项在预测选项一节中均有介绍。

预测选项

`algorithm`

语法: `algorithm=LL | LLT | LLP | LLP5 | LLB | BiLL`

描述: 指定要应用的预测算法的名称。LL、LLT、LLP 和 LLP5 为单变量算法。LLB 和 BiLL 为双变量算法。所有算法都在 Kalman 过滤器的基础上改变而来。每种算法都需要最少量的数据点数。如果未提供足够的有效数据点，则显示错误消息。例如，字段本身的数据点数可能很充分，但若你指定的 `holdback` 值很大，有效数据点数可能很少。

默认值: LLP5

算法选项	算法类型	描述
LL	局部等级	没有趋势和季节性的单变量模型。需要最少 2 个数据点。LL 是最简单的算法，可计算时间系列的层级。例如，每个新状态等于旧状态加上高斯噪声。

LLT	局部等级趋势	具有趋势但没有季节性的单变量模型。需要最少 3 个数据点。
LLP	季节性局部等级	具有季节性的单变量模型。数据点的数量必须至少为时段数量的两倍，使用 period 属性。LLP 算法考量到了数据的循环规律（若存在）。如果您知道时段的数量，指定 period 参数。若您不设置 period，该算法将尝试计算时段。若数据并非定期数据，LLP 将返回错误消息。
LLP5	为其预测合并 LLT 和 LLP 模型。	若为定期的时间系列，LLP5 将计算两个预测值，一个用 LLT 另一个用 LLP。该算法然后再获取两个值的加权平均值，并将该值输出为预测值。置信区间也由 LLT 与 LLP 方差的加权平均值决定。
LLB	双变量局部等级	没有趋势也没有季节性的单变量模型。需要最少 2 个数据点。LLB 使用一组数据进行另一组的预测。例如，假设 LLB 使用数据集 Y 预测数据集 X。若 holdback=10，LLB 将用数据集 Y 的最后 10 个数据点来预测数据集 X 的最后 10 个数据点。
BiLL	双变量局部等级	同时预测两个时间系列的双变量模型。把两个系列的协方差纳入考量。

correlate

语法：correlate=<field>

描述：指定 LLB 算法预测其他时间系列时使用的时间系列。指定 LLB 算法时需要使用时间系列。其他任何算法都不会用到时间系列。

默认值：无

future_timespan

语法：future_timespan=<num>

描述：指定 predict 命令要计算的未来预测的数量。该数量不能为负数。勿使用 future_timespan 选项的前提为 algorithm=LLB。

默认值：5

holdback

语法：holdback=<num>

描述：从未尾指定 predict 命令不会使用的数据点数。和 future_timespan 参数一起使用。例如，'holdback=10 future_timespan=10' 计算数据集中最后 10 个值的预测值。计算完毕后，您可以检查实际的数据点值是否落入预测的置信区间，由此来判断预测的精确度。

默认值：0

lowerXX

语法：lower<int>=<field>

描述：指定置信区间的百分比和用于置信区间曲线下端的字段名称。<int> 值是一个指定置信区间的百分比。该整数必须为 0 到 100 之间的数字。<field> 值为字段名称。

默认值：默认的置信区间为 95%。默认字段名称为 'lower95(prediction(X))'，其中 X 是要预测的字段的名称。

period

语法：period=<num>

描述：在时间系列数据中指定时间段或循环期长度。该数字必须至少为 2。若未指定任何值，LLP 和 LLP5 算法将尝试计算时间段的长度。若您用 timechart 命令指定 span 参数，您指定的 span 单位是用于 period 的单位。例如，如果搜索是 ...|timechart span=1d foo2| predict foo2 period=3。跨度是 1 天，预测期限是 3 天。另外，时间段单位是一个数据点。例如，如果有上千个事件，则每个事件都是一个单位。如果您指定 period=7，则表示数据在每 7 个数据点或事件之后循环使用。

默认值：无

suppress

语法：suppress=<field>

描述：和多变量算法一起使用。从多个已预测字段中指定一个，禁止其显示于结果中。很难一次看到所有已预测字段的可视化结果时，请使用 suppress。

默认值：无

upperYY

语法：upper<int>=<field>

描述：指定置信区间的百分比和用于置信区间曲线上端的字段名称。<int> 值是一个指定置信区间的百分比。该值必须为 0 到 100 之间的数字。<field> 值为字段名称。

默认值：默认的置信区间为 95%。默认字段名称为 'upper95(prediction(X))'，其中 X 是要预测的字段的名称。

置信间隔

置信区间上下端参数默认为 lower95 和 upper95。这些值将指定置信间隔，其中 95% 的预测预计会失败。

部分预测会落在置信区间之外，这一点很常见。

- 置信间隔不覆盖 100% 的预测。
- 置信区间是概率预期，同时结果不完全匹配预期。

用法

命令顺序要求

`predict` 命令在顺序上必须次于 `timechart` 命令。`predict` 命令需要时间系列数据。更多详情请参阅示例一节。

如何工作

规定一个不会被察觉到的实体，该实体随时间的推进以不同状态持续工作，由此，`predict` 命令即可为数据建模。

若要预测一个值，该命令会把过去的所有数据纳入考量，由此计算状态的最佳估算值。若要计算各状态的估算值，该命令假设各状态遵守包含高斯噪声的特定线性方程。

在该假设下可以快速地计算出各状态的最小平方估算值。此计算方法名为卡尔曼 (Kalman) 滤波器或卡尔曼-布西 (Kalman-Bucy) 滤波器。计算过程中将获取每个状态估算值的置信区间。该估算值并非点估算值，而是一个值范围，包含观测值和预测值。

测量结果可能仅捕获状态的某个方面，不过也没必须捕获整个状态。

缺失的值

`predict` 命令可以处理缺失了某些值的数据。该命令计算各缺失值的最佳估算值。

勿删除缺失了某些值的事件，删除这些事件可能会扭曲数据的周期。指定 `cont=false` 时勿使用 `timechart` 命令。指定 `cont=false` 将会删除缺失了某些值的事件。

指定范围

用 `timechart` 命令指定的 `span` 的单位必须是秒或更长的时间单位。`predict` 命令在计算 `period` 时，无法接受输入亚秒。

示例

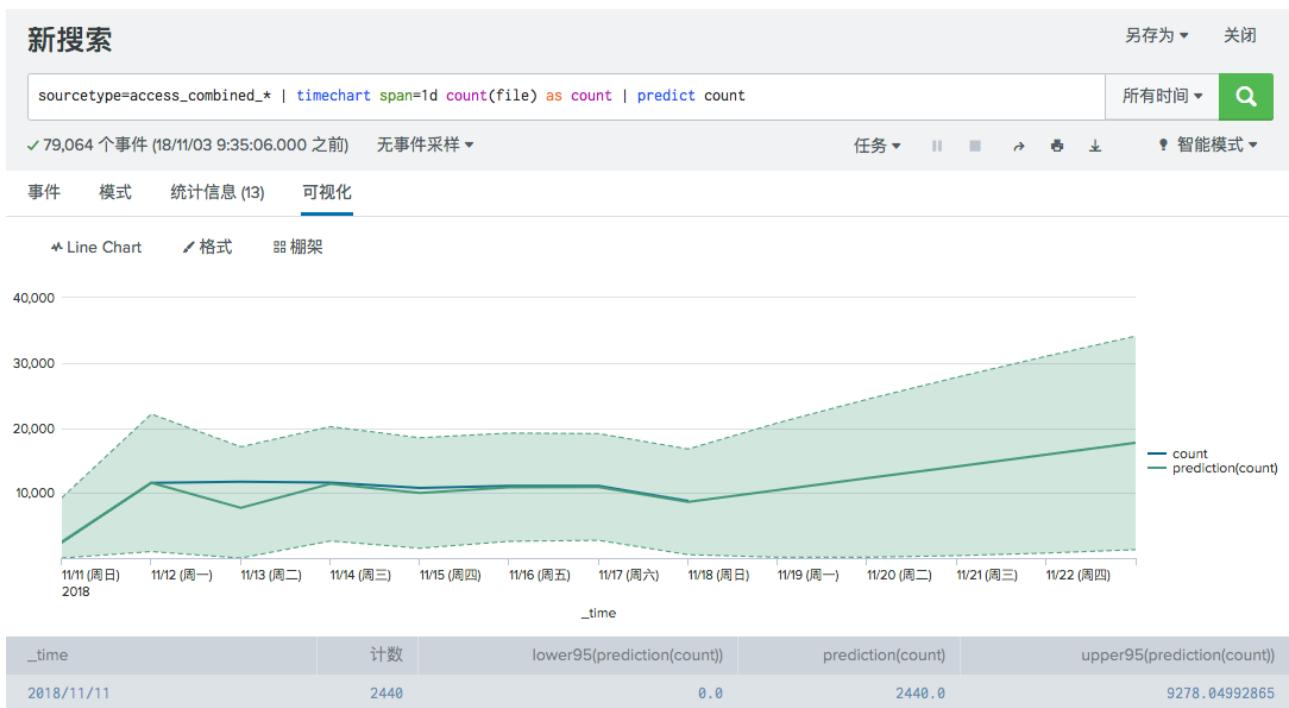
1. 预测未来访问量

此示例使用搜索教程中的示例数据，但应该可以与任意格式的 Apache Web 访问日志配合使用。要在您自己的 Splunk 实例中尝试此示例，您必须下载示例数据并遵循操作以将教程数据导入 Splunk。当运行搜索时，请使用时间范围所有时间。

基于 Apache Web 访问日志文件中存储的之前的访问量预测未来访问量。使用 1 天的时间跨度统计访问次数。

```
sourcetype=access_combined_* | timechart span=1d count(file) as count | predict count
```

结果将显示在“统计”选项卡中。单击“可视化”选项卡。必要时将图表类型修改为折线图。



2. 预测产品未来购买量

此示例使用搜索教程中的示例数据。要在您自己的 Splunk 实例中尝试此示例，您必须下载示例数据并遵循操作以将教程数据导入 Splunk。当运行搜索时，请使用时间范围所有时间。

绘制每天特定产品购买数量的图表。

```
sourcetype=access_* action=purchase arcade | timechart span=1d count
```

- 本示例搜索所有的购物事件，由弧度 `action=purchase` 定义，并通过管道符把结果传递给 `timechart` 命令。
- `span=1day` 参数用数据桶把购物事件计数存储成以天为单位的数据块。

统计选项卡中显示的结果如下所示：

_time	count
2018-06-11	17
2018-06-12	63
2018-06-13	94
2018-06-14	82
2018-06-15	63
2018-06-16	76
2018-06-17	70
2018-06-18	72

将 `predict` 命令添加到搜索以计算不久的将来可能出售的 Arcade 游戏的预计购买量。

```
sourcetype=access_* action=purchase arcade | timechart span=1d count | predict count
```

结果将显示在“统计”选项卡中。单击“可视化”选项卡。必要时将图表类型修改为条形图。



3. 使用默认算法预测值

用默认的 LLP5 算法预测 `foo` 值，该算法结合了 LLP 和 LLT 算法。

```
... | timechart span="1m" count AS foo | predict foo
```

4. 使用同种算法预测多个字段

使用同种算法预测多个字段。本示例中的默认算法。

```
... | timechart ... | predict foo1 foo2 foo3
```

5. 指定不同的置信区间上下限

指定置信区间时，上下限值无需匹配。本示例使用 LL 算法预测某字段的 10 个值，并隐藏数据集中的最后 20 个值。

```
... | timechart span="1m" count AS foo | predict foo AS foobar algorithm=LL upper90=high lower97=low future_timespan=10 holdback=20
```

6. 使用 LLB 算法预测值

本示例介绍 LLB 算法。将 `foo3` 字段与 `foo2` 字段关联起来，由此预测 `foo3` 字段。

```
... | timechart span="1m" count(x) AS foo2 count(y) AS foo3 | predict foo3 AS foobar algorithm=LLB correlate=foo2 holdback=100
```

7. 省略最后 5 个数据点并预测 5 个未来值

在本示例中，搜索放弃使用最后 5 个数据节点，改而预测 5 个未来值。预测值与数据中的最后 5 个值对应。预测完毕后，您可以检查观测值是否落入预测的置信区间，由此来判断预测的精确度。

```
... | timechart ... | predict foo holdback=5 future_timespan=5
```

8. 使用相同的算法、future_timespan 和 holdback 预测多个字段

使用相同的算法、`future_timespan` 和 `holdback` 预测多个字段。

```
... | timechart ... | predict foo1 foo2 foo3 algorithm=LLT future_timespan=15 holdback=5
```

9. 指定字段的别名

指定每个字段的 `AS` 关键字，即可使用这些字段的别名。

```
... | timechart ... | predict foo1 AS foobar1 foo2 AS foobar2 foo3 AS foobar3 algorithm=LLT future_timespan=15 holdback=5
```

10. 使用不同的算法和选项预测多个字段

针对每个字段使用不同的算法和选项预测多个字段。

```
... | timechart ... | predict foo1 algorithm=LL future_timespan=15 foo2 algorithm=LLP period=7 future_timespan=7
```

11. 使用 BiLL 算法预测多个字段

使用双变量算法 BiLL 同时预测 foo1 和 foo2 的值。

```
... | timechart ... | predict foo1 foo2 algorithm=BiLL future_timespan=10
```

另请参阅

trendline, x11

rangemap

描述

使用 rangemap 命令将数字字段中的各值进行分类。此命令将为每个事件添加一个名为 range 的新字段，并在 range 字段中显示其类别。range 字段中的各值基于您指定的数字范围。

把 range 字段设置为任意 attribute_name（包含输入 field 的值）的名称。若未匹配到任何范围，则把 range 值设置为 default 值。

您所设置的范围可以重叠。若有重叠值，将把 range 字段新建为包含所有应用值的多值字段。例如，若 low=1-10, elevated=5-15 且输入字段值为 10，则 range=low 和 code=elevated。

语法

要求的语法以粗体表示。

```
rangemap
field=<string>
[<attribute_name>=<numeric_range>]...
[default=<string>]
```

必要参数

字段

语法: `field=<string>`

描述: 输入字段的名称。该字段必须包含数字值。

可选参数

`attribute_name=numeric_range`

语法: `<string>=<num>-<num>`

描述: `<attribute_name>` 是 `<numeric_range>` 与 `<field>` 中的值匹配时输出的字符串值。`<attribute_name>` 是一个输出到 range 字段的输出。`<numeric_range>` 为范围的起始值和结束值。可以是整数值或浮点数。第一个值必须要小于第二个值。`<numeric_range>` 可以包含负值。

示例: Dislike=-5--1 DontCare=0-0 Like=1-5

默认

语法: `default=<string>`

描述: 如果输入字段与范围不匹配，则使用此参数定义默认值。

默认值: "None"

用法

rangemap 命令属于可分配的流命令。请参阅“命令类型”。

基本示例

示例 1:

若 date_second 介于 1-30 之间，把 range 设置为 "green"；若介于 31-39 之间，设置为 "blue"；介于 40-59 之间则设置为 "red"；若未匹配到任何范围（如 date_second=0）则设置为 "gray"。

```
... | rangemap field=date_second green=1-30 blue=31-39 red=40-59 default=gray
```

示例 2:

把每个事件的 range 字段设置为 "low" 时需要该事件的 count 字段为零 (0)；若 "count" 字段介于 1-100，设置为 "elevated"，其他情况则设置为 "severe"。

```
... | rangemap field=count low=0-0 elevated=1-100 default=severe
```

延伸示例

此示例使用从 USGS 地震网站下载的近期地震数据。该数据是一个逗号分隔的 ASCII 文本文件，其中包含每次记录的地震的震级 (mag)、坐标 (经度、纬度)、区域 (地点) 等。

您可以从 [USGS 地震源](#) 下载当前 CSV 文件并作为输入添加。以下示例使用过去 30 天列表中的所有地震。

此搜索计算了阿拉斯加及其周围发生的地震数和每次地震的震级。然后，使用 rangemap 命令为每个震级分配一种颜色。

```
source=all_month.csv place=*alaska* mag>=3.5 | stats count BY mag | rename mag AS magnitude | rangemap field=magnitude  
light=3.9-4.3 strong=4.4-4.9 severe=5.0-9.0 default=weak
```

结果形式如下所示：

震级	计数	范围
3. 7	15	弱
3. 8	31	弱
3. 9	29	淡色
4	22	淡色
4. 1	30	淡色
4. 2	15	淡色
4. 3	10	淡色
4. 4	22	强
4. 5	3	强
4. 6	8	强
4. 7	9	强
4. 8	6	强
4. 9	6	强
5	2	严重
5. 1	2	严重
5. 2	5	严重

按范围值汇总结果

```
source=all_month.csv place=*alaska* mag>=3.5 | stats count BY mag | rename mag AS magnitude | rangemap field=magnitude  
green=3.9-4.2 yellow=4.3-4.6 red=4.7-5.0 default=gray | stats sum(count) by range
```

结果形式如下所示：

范围	sum(count)
灰色	127
绿色	96
红色	23
黄色	43

按自定义排序顺序排列结果

默认情况下，搜索结果中的值按 sum(count) 字段以降序排列。您可以使用带 case 函数的 eval 命令对结果应用自定义排序顺序。

```
source=all_month.csv place=*alaska* mag>=3.5 | stats count BY mag | rename mag AS magnitude | rangemap field=magnitude  
green=3.9-4.2 yellow=4.3-4.6 red=4.7-5.0 default=gray | stats sum(count) by range | eval sort_field=case(range="red",1,  
range="yellow",2, range="green",3, range="gray",4) | sort sort_field
```

结果形式如下所示：

范围	sum(count)	sort_field
红色	23	1
黄色	43	2
绿色	96	3
灰色	127	4

另请参阅

命令

Eval

博客

排序起来！自定义排序顺序

rare

描述

显示最不常用的字段值。

查找字段列表中所有字段的最不常用的一组值。若指定 <by-clause>，该命令将返回各分组依据字段值所有非重复数据元组的罕见值元组。

此命令的操作方法与 top 命令相同，差别仅在于 rare 命令查找的是最不常用的值，而非最常用的值。

语法

```
rare [<top-options>...]<field-list> [<by-clause>]
```

必要参数

<field-list>

语法：<string>,...

描述：以逗号分隔的字段名称的列表。

可选参数

<top-options>

语法: countfield=<string> | limit=<int> | percentfield=<string> | showcount=<bool> | showperc=<bool>
描述: 指定待显示值类型和数量的选项。这些选项正是 `top` 命令所用的 <top-options>。

<by-clause>

语法: BY <field-list>

描述: 分组所依据的一个或多个字段的名称。

Top 选项

countfield

语法: countfield=<string>

描述: 要将计数值写入其中的新字段的名称。

默认值: "count"

limit

语法: limit=<int>

描述: 指定要返回的元组数量。若指定 `limit=0`, 所有达到 `maxresultrows` 的值都将返回。请参阅限制一节。指定的值若大于 `maxresultrows` 会发生错误。

默认值: 10

percentfield

语法: percentfield=<string>

描述: 要写入百分比值的新字段的名称。

默认值: "percent"

showcount

语法: showcount=<bool>

描述: 指定是否使用该元组的计数新建名为 "count" 的字段（请参阅 "countfield" 选项）。

默认值: true

showperc

语法: showperc=<bool>

描述: 指定是否使用该元组的相对流行度新建名为 "percent" 的字段（请参阅 "percentfield" 选项）。

默认值: true

用法

`rare` 命令是一个转换命令。请参阅“命令类型”。

`rare` 命令返回的结果数受制于 `limit` 参数。`limit` 参数的默认值为 10。您可以将此限制更改为 `limits.conf` 文件中 `[rare]` 段落中 `maxresultrows` 设置中指定的最大值。默认最大值为 50,000, 有效限制了 `rare` 命令使用的内存。

示例

1. 返回字段中最不常用的值

返回 "url" 字段中最不常用的值。限制返回 5 的值的数量。

```
... | rare url limit=5
```

2. 返回由主机组织的最不常用的值

为每个“主机”值找到“用户”字段中最不常用的值。默认情况下，最多返回 10 个结果。

```
... | rare user by host
```

另请参阅

`top`, `stats`, `sirare`

redistribute

描述

`redistribute` 命令实施并行化简搜索处理缩短受支持的 SPL 命令集的搜索运行时间。将 `redistribute` 命令应用到聚合大量搜索结果的高基数数据集搜索。

`redistribute` 命令需要分布式搜索环境，其中，索引器已配置为作为中间化简器操作。只有角色拥有`run_multi_phased_searches` 功能的用户才能在搜索中使用 `redistribute` 命令。

您在搜索中只能使用一次 `redistribute` 命令。

语法

```
redistribute [num_ofReducers=<int>] [<by-clause>]
```

必要参数

无。

可选参数

num_ofReducers

语法: `num_ofReducers=<int>`

描述: 指定被重新调整为中间化简器的索引器池中的索引器数量。

默认值: `num_ofReducers` 的默认值受制于 `limits.conf` 文件中的三个设置: `maxReducersPerPhase`、`winningRate` 和 `reducers`。如果这些设置未更改，默认情况下，Splunk 软件将 `num_ofReducers` 设置为索引器池的 50%，最多 4 个索引器。更多信息请参阅“用法”。

by-clause

语法: `BY <field-list>`

描述: 分组所依据的一个或多个字段的名称。您不能使用通配符字符指定具有相似名称的多个字段。您必须单独指定每个字段。请参阅“使用 by-clause”，了解更多信息。

用法

在具有分布式搜索的 Splunk 部署中，通常使用两个阶段的地图化简过程以确定搜索的最终结果集。索引器层中将进行搜索结果映射，然后在搜索头进行化简。

`redistribute` 命令向映射-化简过程插入中间化简阶段，使它成为一个三个阶段的映射-化简-化简过程。此三阶段过程是并行化简搜索过程。

在中间化简阶段，索引器的子集变成中间化简器。中间化简器为搜索命令执行化简操作，然后将结果传递到搜索头，最终结果化简和聚合操作就在搜索头进行。本来将完全由搜索头完成的并行化简缩短聚合大量搜索结果的高基数数据搜索完成时间。

关于在索引器级别管理并行化简过程，包括配置索引器以中间化简器身份操作，请参阅《分布式搜索》手册中的“并行化简搜索处理概览”。

如果您使用 Splunk Cloud，只能在索引器正在以中低等的平均负载运行时，才能使用 `redistribute`。使用 `redistribute` 命令不需要进行任何配置。

支持的命令

`redistribute` 命令只支持流式命令和以下非流式命令：

- `stats`
- `tstats`
- `streamstats`
- `eventstats`
- `sichart`
- `sitimechart`

当 `transaction` 命令只针对字段运行时，`redistribute` 命令还支持 `transaction` 命令。例如：当以下条件为 `true` 时，`redistribute` 命令无法支持 `transaction` 命令：

- `redistribute` 命令在 `<by-clause>` 参数中有多个字段。
- `transaction` 命令在 `<field-list>` 参数中有多个字段。
- 您在没有指定字段的模式下使用 `transaction` 命令。

为达到最佳性能，将 `redistribute` 放在有高基数输入的首个受支持的非流式命令之前。

搜索处理移至搜索头时

在以下情况下，`redistribute` 命令将搜索字符串处理从中间化简器移至搜索头：

- 遇到了不支持的非流式命令。
- 遇到支持但不包括拆分依据字段的命令。
- 遇到支持命令和包括拆分依据字段，但拆分字段不是 `redistribute` 命令的 `by-clause` 参数中已指定的字段的超集。
- 检测到命令修改 `redistribute` 命令中 `by-clause` 中指定的字段值。

使用 `by-clause` 确定如何在化简器上划分结果

在中间化简阶段开始时，`redistribute` 命令获取映射的搜索结果，根据 `by-clause` 参数指定的字段将它们重新分配到中间化简器的分区。如果您不指定任何 `by-clause` 字段，搜索处理器使用最适合与搜索字符串中 `redistribute` 命令之后的命令一起使用的—一个或多个字段。

命令类型

`redistribute` 命令是编排命令，表示它能够控制搜索运行的方式。不专注于由搜索处理的事件。`redistribute` 命令指导分布式搜索查询计划程序通过将集中式流数据分发在中间化简器，将集中式流数据转换为分布流数据。

关于命令类型的更多信息，请参阅《搜索手册》中的“命令类型”。

设置中间化简器的默认数量

`num_ofReducers` 参数的默认值受 `limits.conf` 文件中的三个设置控制：`maxReducersPerPhase`、`winningRate` 和 `reducers`。

设置名称	定义	默认值
<code>maxReducersPerPhase</code>	可以在中间化简阶段用作中间化简器的索引器的最大数量。	4
<code>winningRate</code>	可以从总计索引器池中选择并在并行化简搜索处理中用作中间化简器的索引器百分比。此设置只在未配置 <code>reducers</code> 设置时适用。	50
<code>reducers</code>	用作并行化简搜索处理专用中间化简器的有效索引器列表。当您使用 <code>redistribute</code> 命令运行搜索时， <code>reducers</code> 列表中的有效索引器仅是用于并行化简操作的那些索引器。如果 <code>reducers</code> 列表中有效索引器的数量超出 <code>maxReducersPerPhase</code> 值，Splunk 平台从 <code>reducers</code> 列表中随机选择一组满足 <code>maxReducersPerPhase</code> 限制的索引器。	" "(空列表)

如果您决定将 7 个索引器添加到 `reducers` 列表中，将停止应用 `winningRate` 设置，`num_ofReducers` 参数默认为 4 个索引器。Splunk 平台从 `reducers` 列表中随机选择四个索引器作为每次运行有效 `redistribute` 搜索时的中间化简器。

如果您提供的 `num_ofReducers` 参数的值超出 `maxReducersPerPhase` 设置的限制，Splunk 平台会将化简器数量设置为 `maxReducersPerPhase` 值。

`Redistribute` 命令和搜索头数据

使用 `redistribute` 命令的搜索将忽视搜索头上的所有数据。如果您计划使用 `redistribute` 命令，最佳做法是将所有搜索头数据转发至索引器层。请参阅“最佳做法”：“将搜索头数据转发到索引器层”，该部分在《分布式搜索》手册中。

在 `chart` 和 `timechart` 搜索中使用重新分布命令

如果您想要向使用 `chart` 或 `timechart` 命令的搜索添加 `redistribute` 命令以产生可以用于图表可视化的统计结果，也需要在搜索中包含 `sichart` 命令或 `sitimechart` 命令。`redistribute` 命令使用 `si-` 命令为中间化简器上的报表命令执行统计计算。`redistribute` 命令将结果移至搜索头，`chart` 或 `timechart` 命令将结果转换成可以用于图表可视化的格式。

最佳做法是为这两个命令使用相同的语法和值。例如：如果您想要在 `redistribute` 搜索中包含 `| timechart count by referrer_domain`，请将 `| sitimechart count by referrer_domain` 插入到搜索字符串：

```
index=main | redistribute | transaction referer_domain | search eventcount>500 | sitimechart count by referrer_domain | search
referer_domain=*.net | timechart count by referrer_domain
```

如果搜索中存在对顺序敏感的命令

`redistribute` 命令支持的命令按顺序显式返回结果。因为 `redistribute` 命令运行时发生的分区，Splunk 平台丢失了排序顺序。如果 Splunk 平台检测到一个顺序敏感型命令，如 `streamstats`，用于 `redistribute` 搜索中，它会在处理时自动将 `sort` 插入搜索。

例如，以下搜索包括顺序敏感型的 `streamstats` 命令：

```
... | redistribute by host | stats count by host | streamstats count by host, source
```

Splunk 平台在它处理搜索时，在 `streamstats` 段之前添加 `sort` 段。如果您在运行后检查搜索任务，您可以看到搜索字符串中的排序段。

```
... | redistribute by host | stats count by host | sort 0 str(host) | streamstats count by host, source
```

`stats` 和 `streamstats` 段在中间化简器上进行处理，因为它们都按 `host` 字段（`redistribute` 命令分布的相同字段）拆分。`sort` 段的工作拆分为两部分，分别是在搜索的映射阶段的索引器和搜索最终化简阶段的搜索头上进行。

如果您需要 `redistribute` 搜索的排序结果

如果您需要 `redistribute` 搜索的结果以该确切顺序排序，请使用 `sort` 在搜索头上实施排序。中间化简器上 `redistribute` 命令对事件进行分区之后，事件排序有其他性能成本。

以下搜索提供排序结果：

```
search * | stats count by foo
```

如果想在将 `redistribute` 添加到搜索以实现加速的同时获得相同的事件排序，请向搜索添加 `sort`：

```
search * | redistribute | stats count by foo | sort 0 str(foo)
```

此搜索的 `stats` 段在中间化简器上进行处理。`sort` 段的工作拆分为两部分，分别是在搜索的映射阶段的索引器和搜索最终化简阶段的搜索头上进行。

`Redistribute` 和虚拟索引

`redistribute` 命令不支持虚拟索引的搜索。如果统一的搜索时间范围长到可以在虚拟归档索引中运行，`redistribute` 命令也不支持统一的搜索。更多信息，请参阅以下 *Splunk Analytics for Hadoop* 主题：

- 关于虚拟索引
- 配置和运行统一的搜索

示例

1. 在大型高基数数据集上加速搜索

在此示例中，`redistribute` 命令应用于在极大的高基数数据集上运行的 `stats` 搜索。`redistribute` 命令可以缩短搜索的完成时间。

```
... | redistribute by ip | stats count by ip
```

中间化简器并行处理搜索的 `| stats count by ip` 部分，缩短搜索的完成时间。搜索头聚合结果。

2. 不声明要 `redistribute` 的 `by-clause` 字段来加速 `timechart` 搜索

此示例在极大的高基数数据集上使用搜索。搜索字符串包括 `eventstats` 命令，该命令使用 `sitimechart` 命令为 `timechart` 操作的执行统计计算。搜索使用 `redistribute` 命令缩短搜索的完成时间。`by-clause` 字段未指定，所以搜索处理器选择一个。

```
... | redistribute | eventstats count by user, source | where count>10 | sitimechart max(count) by source | timechart max(count) by source
```

此搜索运行时，中间化简器并行处理搜索的 `eventstats` 和 `sitimechart` 段，缩短搜索的整体完成时间。搜索头上，`timechart` 命令取化简的 `sitimechart` 计算并将它们转换为可以用于图表和可视化的格式。

因为搜索字符串中未识别出 `by-clause` 字段，中间化简器将针对 `source` 字段上的事件进行重新分布并分区。

3. 加速使用 `tstats` 生成事件的搜索

此示例在极大的高基数数据集上使用搜索。此搜索结合使用 `sitimechart`、`timechart` 和 `tstats` 命令。`redistribute` 命令可以缩短搜索的完成时间。

```
| tstats prestats=t count BY _time span=1d | redistribute by _time | sitimechart span=1d count | timechart span=1d count
```

必须将 `tstats` 命令放在有前导管道符的搜索字符串的开头。当您结合使用 `tstats` 和 `redistribute` 命令时，您必须将 `redistribute` 命令放在搜索的 `tstats` 段之后。

在此示例中，`tstats` 命令使用 `prestats=t` 参数以结合使用 `sitimechart` 和 `timechart` 命令。

`redistribute` 命令使中间化简器并行处理搜索的 `sitimechart` 段，缩短搜索的整体完成时间。然后化简器将结果推送到搜索头，`timechart` 命令将它们处理成可用于图表和可视化的格式。

4. 加速包括支持和不支持命令的混合搜索

此示例在极大的高基数数据集上使用搜索。搜索使用 `redistribute` 命令以缩短搜索完成时间。搜索包括 `redistribute` 命令支持和不支持的命令。在处理完剩余的搜索后，它使用 `sort` 命令对结果进行排序。您需要 `sort` 命令对事件进行排序，因为 `redistribute` 过程会撤消 `stats` 命令家族中的命令原本提供的排序。

```
... | redistribute | eventstats count by user, source | where count >10 | sort 0 -num(count)
```

在此示例中，中间化简器并行处理 `eventstats` 和 `where` 段。不使用 `redistribute` 命令时，搜索的那些部分完成的更快。

Splunk 平台将搜索的 `sort` 部分的处理工作划分为两部分，分别在索引器和搜索头上进行。

5. 加速支持的命令按照不在 `redistribute` 命令 `by-clause` 参数中的字段进行拆分的搜索

在此示例中，`redistribute` 命令按 `source` 字段在中间化简器中重新分布事件。搜索包括 `redistribute` 命令支持的两个命令，但只有其中一个在中间化简器上处理。

```
... | redistribute by source | eventstats count by source, host | where count > 10 | stats count by userid, host
```

在此案例中，搜索的 `eventstats` 段由中间化简器并行处理，因为它将 `source` 作为拆分依据字段。`where` 段也在中间化简器上处理。

但是，搜索的 `stats` 部分在搜索头上处理，因为其拆分依据字段不是事件重新分布依据字段集的超集。也就是说，`stats` 拆分依据字段不包括 `source`。

regex

描述

删除与指定正则表达式不匹配的结果。

语法

要求的语法以粗体表示。

```
regex  
<field>=<regex-expression> | <field>!<regex-expression> | <regex-expression>
```

必要参数

<regex-expression>

语法：<string>"

描述：未定位的正则表达式。该正则表达式必须为受 PCRE 库支持且与 Perl 兼容的正则表达式。需要用双引号引起来。

可选参数

<field>

语法：<field>

描述：指定要与正则表达式进行值匹配的字段名称。

您可以使用 `<field>=<regex-expression>` 指定让 `regex` 命令保留与表达式相匹配的结果。若要保留匹配失败的结果，请指定 `<field>!<regex-expression>`。

默认值：`_raw`

用法

`regex` 命令属于可分配的流命令。请参阅“命令类型”。

使用 `regex` 命令删除不匹配指定正则表达式的结果。

使用 `rex` 命令通过以正则表达式命名的组提取字段，或通过 `Sed` 表达式替换或取代字段中的字符。

在搜索中使用正则表达式时，需注意管道符（|）和反斜杠（\）等字符应如何使用。请参阅《搜索手册》中的“SPL 和正则表达式”。

有关正则表达式的一般信息，请参阅《知识管理器手册》中的“有关 Splunk 正则表达式”。

示例

示例 1：仅保留 “_raw” 字段包含不可路由 A 类中的 IP 地址（10.0.0.0/8）的搜索结果。本示例将在表达式开头使用负推后量断言。

```
... | regex _raw="^(?!\\d)10\\.\\d{1,3}\\.\\d{1,3}\\.\\d{1,3}(?!\\d)"
```

示例 2：只保留匹配有效电子邮件地址的结果。例如：buttercup@example.com。

```
... | regex email="^([a-zA-Z_\\.-]+)@([\\da-zA-Z\\.-]+)\\.([a-zA-Z\\.]{2,6})$"
```

以下表格说明表达式的每个部分。

表达式部分	描述
^	指定字符串的开始。
([a-zA-Z_\\.-]+)	这是表达式中的第一组。指定以匹配一个或多个小写字母、数字、下划线、点或连字符。反斜杠（\）字符用于转义点（.）字符。点字符将转义，因为非转义点匹配任意字符。加（+）号指定匹配此组中的 1 到无限个字符。在这一示例中，这一部分的表达式匹配电子邮件地址 buttercup@example.com 中的 buttercup 。
@	匹配 at 符号。
([\\da-zA-Z\\.-]+)	这是表达式中的第二组。指定匹配域名，可能是一个或多个小写字母、数字、下划线、点或连字符。之后紧跟另一个转义点字符。加（+）号指定匹配此组中的 1 到无限个字符。在这一示例中，这一部分的表达式匹配电子邮件地址 buttercup@example.com 中的 example 。
([a-zA-Z\\.]{2,6})	这是第三组。指定以匹配顶级域（TLD），可以是 2 到 6 个字母或点。此组匹配所有类型的 TLD，如 .co.uk、.edu 或 .asia。此示例中，它将匹配电子邮件地址 buttercup@example.com 中的 .com 。
\$	指定字符串的结束。

另请参阅

命令

```
rex  
search
```

relevancy

描述

根据事件 _raw 字段与 'search' 的关键字的匹配程度来计算事件与查询的匹配程度。结果保存于名为 "relevancy" 的字段中。用于检索匹配程度最高的事件/文档，而不是默认的基于时间的顺序。事件的搜索关键字越稀少、使用越频繁、术语越少，则相关性越高。例如，在搜索 disk error 时，与使用一次 'disk'、使用多次 'error' 的非常大的事件相比，更偏向于使用多次 'disk'（不常用的术语）而只使用一次 'error' 的短的事件/文档。

注意：目前 relevancy 命令不能使用。请参阅下面“已知问题”页面的 SPL-93039：
<http://docs.splunk.com/Documentation/Splunk/latest/ReleaseNotes/KnownIssues>

语法

```
relevancy
```

示例

示例 1：计算搜索的相关性并将结果按降序排序。

```
disk error | relevancy | sort -relevancy
```

另请参阅

```
abstract, highlight, sort
```

reltime

描述

新建一个名为 'reltime' 的相对时间字段，将此字段设置为 'now' 与 '_time' 之间的差值并用易于理解的方式表示。易于理解的值类似于“5 天前”、“1 分钟前”、“2 年前”等。

语法

reltime

用法

reltime 命令属于可分配的流命令。请参阅“命令类型”。

示例

示例 1:

把一个名为 reltime 的字段添加到搜索返回的事件。

```
... | reltime
```

另请参阅

convert

rename

描述

使用 rename 命令重命名一个或多个字段。可使用该命令为字段提供更有意义的名称，例如 "Product ID" 而非 "pid"。如果想要重命名具有相似名称的字段，可以使用通配符。请参阅“用法”一节。

语法

```
rename <wc-field> AS <wc-field>...
```

必要参数

wc-field

语法: <string>

描述: 字段的名称以及用于替换该字段名的名称。含空格的字段名称必须用引号引起来。您可以使用星号 (*) 作为通配符来指定具有相似名称的字段列表。例如，如果要指定所有以 "value" 开头的字段，则可以使用通配符，例如 value*. value*。

用法

rename 命令属于可分配的流命令。请参阅“命令类型”。

用短语重命名

重命名带短语的字段时使用引号。

```
... | rename SESSIONID AS "The session ID"
```

重命名多个类似的已命名的字段

使用通配符重命名多个字段。

```
... | rename *ip AS *IPaddress
```

如果数据来源字段和目标字段均是含有通配符的表达式，并且包含的通配符数量相同，则重命名会在目标表达式中续接通配符

部分。请参阅“示例”。

不能将一个字段重命名为多个名称

不能将一个字段重命名为多个名称。例如，如果您有字段 A，您不能指定 | rename A as B, A as C。此规则还适用于您可以重命名字段的其他命令，如 stats 命令。

以下示例无效。

```
... | stats first(host) AS site, first(host) AS report
```

不能将多个字段合并为一个字段

您不能使用 rename 命令将多个字段合并为一个字段，因为值中存在空值或不存在的字段。

例如，如果您有含 product_id 或 pid 字段的事件，... | rename pid AS product_id 不会把 pid 值合并到 product_id 字段中。事件没有 pid 的情况下，将用空值覆盖 product_id。请参阅 eval 命令和 coalesce() 函数。

重命名不存在的字段

重命名字段可能导致数据丢失。

假设您将 fieldA 重命名为 fieldB，但是 fieldA 不存在。

- 如果 fieldB 不存在，则不会有任何变动。
- 如果 fieldB 存在，重命名的结果是删除的 fieldB 中的数据。fieldB 中的数据将包含空值。

示例

示例 1:

把 "_ip" 字段重命名为 "IPAddress"。

```
... | rename _ip AS IPAddress
```

示例 2:

将以 "foo" 开头的字段重命名为以 "bar" 开头。

```
... | rename foo* AS bar*
```

示例 3:

重命名 "count" 字段。含空格的名称必须用双引号引起来。

```
... | rename count AS "Count of Events"
```

另请参阅

fields、replace、table

replace

描述

将字段值替换为您指定的值。

将只出现一次的第一个字符串替换为指定字段中的另一个字符串。若未指定一个或多个字段，将替换所有字段中的值。

语法

```
replace (<wc-string> WITH <wc-string>)... [IN <field-list>]
```

必要参数

wc-string

语法: <string>

描述: 指定一个或多个字段值及其替换值。可以使用通配符字符匹配一个或多个术语。

可选参数

field-list

语法: <string> ...

描述: 指定一个以逗号或空格分隔的列表, 内容为字段值替换值的一个或多个字段名称。要替换 _internal 字段上的值, 您必须使用 IN <fieldname> 子句指定字段名称。

用法

replace 命令属于可分配的流命令。请参阅“命令类型”。

后指定的无通配符替换值优先于先指定的替换值。若为通配符替换值, 匹配程度高的优先于匹配程度低的。为保证优先关系, 建议将替换操作分解为两个单独的调用。使用通配符替换值时, 结果的通配符数量必须相同, 或者完全没有通配符。可以用通配符 (*) 指定替换操作中的多个替换源值或替换目标值。

示例

1. 替换所有字段中的值

将所有字段中以 "localhost" 结尾的主机值更改为简单的 "localhost"。

```
... | replace *localhost WITH localhost
```

2. 替换特定字段中的值

用 host 字段中更具描述性的名称替换 IP 地址。

```
... | replace 127.0.0.1 WITH localhost IN host
```

3. 更改两个字段的值

替换 start_month 和 end_month 字段中的值。您可以用空格或逗号分隔字段列表中的名称。

```
... | replace aug WITH August IN start_month end_month
```

4. 更改字段中的值的顺序

在主机字段中, 更改包含 localhost 的字符串值的顺序, 这样字符串 "localhost" 会位于其他字符串前面。

```
... | replace "* localhost" WITH "localhost *" IN host
```

5. 替换字段中的多个值

用更具描述性的名称替换字段中的多个值。用逗号分隔替换值。

```
... | replace 0 WITH Critical, 1 WITH Error IN msg_level
```

6. 替换空字符串

搜索错误消息并用空白替换空字符串。

仅当确实存在具有空字符串的值（这与没有值不同）时，此示例才会生效。

```
"Error exporting to XYZ :" | rex "Error exporting to XYZ:(?.*)" | replace "" WITH " " IN errmsg
```

7: 替换内部字段中的值

替换内部字段 _time 的值。

```
sourcetype=* | head 5 | eval _time="XYZ" | stats count BY _time | replace *XYZ* WITH *ALL* IN _time
```

另请参阅

命令
rename

require

描述

如果搜索字符串中位于搜索字符串前面的查询和命令返回零个事件或结果，则会导致搜索失败。

语法

要求的语法以**粗体**表示。

```
| require
```

用法

在搜索字符串中使用 `require` 时，如果搜索字符串中位于该字符串之前的查询和命令返回零个事件或结果，则会导致搜索失败。如果在子搜索中使用，则当子搜索无法返回结果时，它会导致父搜索失败。

使用此命令可以防止 Splunk 平台在可能产生特定负面影响的情况下运行零结果搜索，例如生成误报、运行会产生昂贵 API 调用的自定义搜索命令，或通过子搜索创建空搜索过滤器。

`require` 命令不能用于实时搜索。

require 和后续命令

不要期望 `require` 命令会减轻搜索的所有可能的负面影响。当 `require` 命令导致搜索失败时，它会阻止搜索中的后续命令接收结果，但不会阻止 Splunk 软件在搜索完成之前调用这些命令。这意味着那些后续的搜索命令处理器在搜索完成之前可能会收到空的“数据块”。

如果要执行自定义搜索命令，请确保它与 `require` 命令可以很好地完成互操作。确保它不会因部分输入而产生副作用。

请参阅开发人员门户上《开发人员指南》中的“为 Splunk Cloud 或 Splunk Enterprise 中的应用程序新建自定义搜索命令”。

示例

1. 如果搜索返回零个结果或事件，则停止运行搜索

```
... | require
```

2. 如果子搜索返回零个事件或结果，则会引发异常，且父搜索会停止。

```
... [ search index=other_index NOSUCHVALUE | require ]
```

rest

描述

`rest` 命令读取 Splunk REST API 端点，并将资源数据返回成搜索结果。

关于 REST API 的信息，请参阅《REST API 用户手册》。

语法

要求的语法以**粗体**表示。

```
| rest <rest-uri>
[ count=<int> ]
[ strict=<bool> ]
[ splunk_server=<wc-string> ]
[ splunk_server_group=<wc-string> ]...
[ timeout=<int> ]
[ <get-arg-name>=<get-arg-value> ]...
```

必要参数

`rest-uri`

语法: <uri>

描述: Splunk REST API 端点的 URI 路径。

可选参数

`count`

语法: `count=<int>`

描述: 限制各 REST 调用返回的结果数。例如, 您有四个索引器和一个搜索头。您可将限制设为 `count=25000`。此结果总共限制为 125000, 即 25000×5 。

如果 `count=0`, 则表示无限制。

默认值: 0

`get-arg-name`

语法: <string>

描述: REST 参数名称。

`get-arg-value`

语法: <string>

描述: REST 参数值。

`splunk_server`

语法: `splunk_server=<wc-string>`

描述: 指定返回结果的分布式搜索节点。可以仅指定一个 `splunk_server` 参数, 但在指定服务器名称时可以使用通配符字符, 以指示多个服务器。例如, 您可以指定 `splunk_server=peer01` 或 `splunk_server=peer*`。用 `local` 表示搜索头。

默认值: 所有配置的搜索节点都会返回信息

`splunk_server_group`

语法: `splunk_server_group=<wc-string>...`

描述: 把结果限制到一个或多个服务器组。可以在字符串中指定一个通配符字符, 来指示多个服务器组。

`strict`

语法: `strict=<bool>`

描述: 设置为 `true` 时, 如果 `rest` 引发错误, 则此参数将强制使搜索彻底失败。即使错误适用于子搜索, 也会发生这种情况。设置为 `false` 时, 许多 `rest` 错误条件会返回警告消息, 但不会导致搜索失败。某些错误条件会导致搜索失败, 即使 `strict=false` 时也会失败。

默认值: `false`

`timeout`

语法: `timeout=<int>`

描述: 指定等待 REST 端点响应的超时时间 (秒)。指定超时=0 表示等待 REST 端点响应没有时间限制。

默认值: 60

用法

`rest` 命令使用运行该命令的人员的 ID 执行验证。

`strict` 的错误处理

每当 `rest` 搜索遇到错误情况时, 使用 `strict` 参数使搜索失败。您可以通过更改 `limits.conf` 中的 `restprocessor_errors_fatal`, 为所有 `rest` 搜索在系统级别上进行此设置。

如果您使用的是 Cloud, 请向 Cloud 支持提出请求, 请示更改 `restprocessor_errors_fatal` 设置。

使用 `strict` 参数覆盖 `rest` 搜索的 `restprocessor_errors_fatal` 设置。

示例

1. 访问保存的搜索任务

```
| rest /services/search/jobs count=0 splunk_server=local | search isSaved=1
```

2. 使用包含特定源类型的搜索查找所有保存的搜索

使用包含 `speccsv` 源类型的搜索字符串查找所有保存的搜索。

```
| rest /services/saved/searches splunk_server=local | rename search AS saved_search | fields author, title, saved_search | search saved_search=*speccsv*
```

3. 将当前搜索用户添加到所有事件

将当前搜索用户添加到所有事件。这对于创建只显示与登录用户关联的事件的报告非常有用。

```
* | head 10 | join [ | rest splunk_server=local /services/authentication/current-context | rename username as auth_user_id | fields auth_user_id ]
```

4. 使用 GET 方法分页和筛选参数

大部分 GET 方法支持一组分页和筛选参数。

要确定端点是否支持这些参数，请在《REST API 参考手册》中查找端点。单击 GET 方法上的扩展，查找分页和过滤参数主题的链接。关于分页和筛选参数的更多信息，请参阅《REST API 参考手册》中的“请求和响应详细信息”。

以下示例使用 `search` 参数识别搜索是否是计划搜索，是否禁用。搜索将在和“搜索头”的监视控制台角色匹配的 Splunk 服务器上查找计划搜索。

```
| rest /servicesNS/-/saved/searches splunk_server_group=dmc_group_search_head timeout=0 search="is_scheduled=1" search="disabled=0"
```

以下是对搜索各部分的解释：

描述	搜索部分
REST 调用名称。	<code> rest /servicesNS/-/saved/searches</code>
只查找和“搜索头”的监视控制台角色匹配的 Splunk 服务器。	<code>splunk_server_group=dmc_group_search_head</code>
等待 REST 调用完成不要超时。	<code>timeout=0</code>
仅查找计划搜索。	<code>search="is_scheduled=1"</code>
仅查找活动搜索（非禁用）。	<code>search="disabled=0"</code>

return

描述

返回子搜索中的值。

`return` 命令用于从子搜索向上传递值。该命令使用一个事件替换传入事件，具有一个属性：`"search"`。为了提升性能，`return` 命令使用 `head` 命令自动限制传入结果的数量，并使用 `fields` 命令自动限制生成字段的数量。

根据默认设置，`return` 命令仅使用第一行结果。使用 `count` 参数指定要使用的结果的数量。

语法

```
return [<count>] [<alias>=<field>...][<field>...][$<field>...]
```

必要参数

无。

可选参数

`<count>`

语法：`<int>`

描述：指定行数。

默认值：1， 表示将第一行结果传递给该命令。

<alias>

语法：<alias>=<field>...

描述：指定要返回的字段别名和值。您可以指定多对别名和值，并用空格分隔。

<field>

语法：<field>...

描述：指定一个或多个要返回的字段，并用空格分隔。

<\$field>

语法：<\$field>

描述：指定一个或多个要返回的值，并用空格分隔。

用法

输出字段名称、别名-值对或只是一个字段值时，这个命令使用起来很方便。

输出	示例
字段名称	return source
Alias=value	return ip=srcip
值	return \$srcip

根据默认设置，return 命令仅使用第一行结果。您可以指定多行，例如 'return 2 ip'。每行都被视为一个 OR 子句，亦即，输出可能为 '(ip=10.1.11.2) OR (ip=10.2.12.3)'。可以指定多个值，并将其置于 OR 子句中。因此，'return 2 user ip' 可能会输出 '(user=bob ip=10.1.11.2) OR (user=fred ip=10.2.12.3)'。

大多数情况下，若在子搜索末尾处使用 return 命令，则无需再用 head、fields、rename、format 和 dedup。

重复值

假设您使用以下搜索：

```
sourcetype=WinEventLog:Security | return 2 user
```

从表面上看，您可能会预期该命令返回前两个非重复用户。但事实上，该命令将查看前两个事件，顺序基于隐含的 head 命令指示的排序顺序。return 命令返回这两个事件内的用户。此命令无法确定用户值是否是唯一的。若这些事件中列出了相同的用户，此命令仅返回一位用户。

若要返回唯一值，需要把 dedup 命令包含于您的搜索中。例如：

```
sourcetype=WinEventLog:Security | dedup user | return 2 user
```

返回字段中的引号

return 命令不能对返回的字段中的引号转义。在使用 return 命令前，必须使用 eval 命令对引号进行转义。例如：

```
...[search eval field2=replace(field1,"\"","\\\"") | return field2]
```

示例

示例 1：

搜索 'error ip=<someip>'，其中 <someip> 是用户 'boss' 最近一次使用的 ip。

```
error [ search user=boss | return ip ]
```

示例 2：

搜索 'error (user=user1 ip=ip1) OR (user=user2 ip=ip2)'，其中用户和 IP 来自最近的两次登录。

```
error [ search login | return 2 user ip ]
```

示例 3：

返回对最后用户的 `userid` 执行 `eval`, 并以 1 递增。

```
... | eval nextid = 1 + [ search user=* | return $id ] | ...
```

另请参阅

`format`, `search`

reverse

语法

`reverse`

描述

颠倒结果的顺序。

注意：`reverse` 命令并不影响搜索返回的事件，只影响事件的显示顺序。对于 CLI，这会包括任何默认或显式的 `maxout` 设置。

注意：`Reverse` 超大结果集（即包含数百万或更多结果的集）需要大量临时存储空间、I/O 和时间。

示例

示例 1:

颠倒结果集的顺序。

```
... | reverse
```

另请参阅

`head`, `sort`, `tail`

rex

描述

使用该命令既可以通过以正则表达式命名的群组提取字段，也可以通过 Sed 表达式替换或取代字段中的字符。

`rex` 命令将把指定字段的值与未定位的正则表达式进行匹配，并将命令的组提取到名称对应的字段中。

如果 `mode=sed`，将用于替换或替代字符的给定 `sed` 表达式应用于所选字段的值。此 `sed` 语法还可用于在索引时以掩码显示敏感数据。请参阅《数据导入手册》中的“使用 `sed` 以使数据匿名”。

若未指定字段，正则表达式或 `sed` 表达式将应用于 `_raw` 字段。针对 `_raw` 字段运行 `rex` 命令可能会影响性能。

将 `rex` 命令用于搜索时间字段提取或字符串替换和字符替代。

语法

要求的语法以粗体表示。

```
rex [field=<field>]  
  (<regex-expression> [max_match=<int>] [offset_field=<string>] ) | (mode=sed <sed-expression>)
```

必要参数

您必须指定 `<regex-expression>` 或 `mode=sed <sed-expression>`。

`regex-expression`

语法： "`<string>`"

描述： PCRE 正则表达式，用于定义要匹配和从指定字段中提取的信息。需要用双引号引起来。

`mode`

语法: mode=sed

描述: 指定以指示您使用的是 Sed (UNIX 流编辑器) 表达式。

sed-expression

语法: "<string>"

描述: 当 mode=sed 时, 指定是否在匹配正则表达式中替换字符串 (s) 或替代字符 (y)。不执行其他 sed 命令。需要用双引号引起。Sed 模式支持下列标记: 全局 (g) 和第 N 个出现 (N), 其中 N 是字符串中字符的位置数值。

可选参数

字段

语法: field=<field>

描述: 要从中提取信息的字段。

默认值: _raw

max_match

语法: max_match=<int>

描述: 控制正则表达式的匹配次数。如果大于 1, 则生成的字段为多值字段。使用 0 指定无限匹配。多次匹配适用于整个模式的重复应用。如果您的正则表达式包含一个可在模式中多次匹配的捕获组, 则只有最后一个捕获组可用于多次匹配。

默认值: 1

offset_field

语法: offset_field=<string>

描述: 创建一个字段, 根据 regex-expression 中指定的正则表达式, 列出 field 参数中某些值的位置。例如, 如果 rex 表达式为 "(?<tenchars>.{10})", 则将匹配 field 参数的前十个字符。offset_field 显示 tenchars=0-9。偏移计算始终使用零 (0) 作为第一个位置。有关其他示例, 请参阅“示例”。

默认值: 无默认值

用法

rex 命令属于可分配的流命令。请参阅“命令类型”。

rex 命令还是 regex 命令?

使用 rex 命令通过以正则表达式命名的组提取字段, 或通过 Sed 表达式替换或取代字段中的字符。

使用 regex 命令删除不匹配指定正则表达式的结果。

正则表达式

Splunk SPL 使用 perl 兼容正则表达式 (PCRE)。

在搜索中使用正则表达式时, 需注意管道符 (|) 和反斜杠 (\) 等字符应如何使用。请参阅《搜索手册》中的“SPL 和正则表达式”。

有关正则表达式的一般信息, 请参阅《知识管理器手册》中的“Splunk Enterprise 正则表达式”。

sed 表达式

当在 sed 模式使用 rex 命令时, 您有两个选项: 替换 (s) 或字符替代 (y)。

使用 sed 替换 (s) 您数据中的文本的语法是: "s/<regex>/<replacement>/<flags>"

- <regex> 是一个 PCRE 正则表达式, 其中可包含捕获组。
- <replacement> 是用于替换正则表达式匹配的字符串。\\n 代表反向引用, 其中 "n" 为一位数。
- <flags> 既可以是用于替换所有匹配的 g, 也可以是用于替换一个指定匹配的数字。

使用 sed 替代字符的语法是: "y/<string1>/<string2>/"

- 该语法将使用 <string2> 中的字符替代与 <string1> 相匹配的字符。

示例

1. 使用正则表达式提取电子邮件值

从事件中提取电子邮件值以在事件中新建 from 和 to。例如, 您有如下事件:

```

Mon Mar 19 20:16:27 2018 Info: Bounced: DCID 8413617 MID 19338947 From: <MariaDubois@example.com> To: <zecora@buttercupgames.com> RID 0 - 5.4.7 - Delivery expired (message too old) ('000', ['timeout'])

Mon Mar 19 20:16:03 2018 Info: Delayed: DCID 8414309 MID 19410908 From: <WeiZhang@example.com> To: <mcintosh@buttercupgames.com> RID 0 - 4.3.2 - Not accepting messages at this time ('421', ['4.3.2 try again later'])

Mon Mar 19 20:16:02 2018 Info: Bounced: DCID 0 MID 19408690 From: <Exit_Desk@example.net> To: <lyra@buttercupgames.com> RID 0 - 5.1.2 - Bad destination host ('000', ['DNS Hard Error looking up mahidnrasatyambsg.com (MX): NXDomain'])

Mon Mar 19 20:15:53 2018 Info: Delayed: DCID 8414166 MID 19410657 From: <Manish_Das@example.com> To: <dash@buttercupgames.com> RID 0 - 4.3.2 - Not accepting messages at this time ('421', ['4.3.2 try again later'])

```

索引事件之后，From 和 To 值不会被识别为字段。您可以使用 rex 命令提取字段值并在搜索结果中新建 from 和 to 字段。

_Raw 事件中的 from 和 to 行遵循相同的模式。每个 from 行都是 From:，每个 to 行都是 To:。用尖括号将电子邮件地址括起来。您可以使用此模式新建正则表达式，以提取值和新建字段。

```
source="cisco_esa.txt" | rex field=_raw "From: <(?<from>.*)> To: <(?<to>.*)>"
```

您可删除重复的值并通过向搜索添加 dedup 和 table 命令仅返回地址列表。

```
source="cisco_esa.txt" | rex field=_raw "From: <(?<from>.*)> To: <(?<to>.*)>" | dedup from to | table from to
```

From	To
eduardo.rodriguez@example.net	pinkie@buttercupgames.com
na.lui@example.net	dash@buttercupgames.com
Vanya_Patel@example.com	rutherford@buttercupgames.com
MariaDubois@example.com	zecora@buttercupgames.com
na.lui@example.net	rarity@buttercupgames.com
WeiZhang@example.com	mcintosh@buttercupgames.com
Exit_Desk@example.net	lyra@buttercupgames.com

结果形式如下所示：

2. 使用 max_match 从多值字段中提取

您可以使用 max_match 参数指定正则表达式运行多次，以从一个字段中提取多个值。

例如，使用 makeresults 命令创建具有多个值的字段：

```
| makeresults | eval test="a$1,b$2"
```

结果形式如下所示：

_time	test
2019/12/5 11:15:28	a\$1, b\$2

要分别提取 test 字段中的每个值，请在 rex 命令中使用 max_match 参数。例如：

```
... | rex field=test max_match=0 "((?<field>[$])*\\$(?<value>[^,]*),?)"
```

结果形式如下所示：

_time	字段	test	值
2019/12/5 11:36:57	a	a\$1, b\$2	1
	b		2



3. 从 `scheduler.log` 事件的字段中提取值

从 `scheduler.log` 事件的名为 "savedsearch_id" 的字段中提取 "user"、"app" 和 "SavedSearchName"。如果 `savedsearch_id=bob;search=my_saved_search`, 那么 `user=bob`、`app=search` 且 `SavedSearchName=my_saved_search`

```
... | rex field=savedsearch_id "(?<user>\w+);(?<app>\w+);(?<SavedSearchName>\w+)"
```

4. 使用 `sed` 表达式

使用 `sed` 语法将正则表达式与一系列数字进行匹配，并将其替换为匿名的字符串。

```
... | rex field=ccnumber mode=sed "s/(\d{4}-){3}/XXXX-XXXX-XXXX-/g"
```

5. 使用 `offset_field`

要标识字段中某些值的位置，请使用带有 `offset_field` 参数的 `rex` 命令和正则表达式。

以下示例首先使用 `makeresults` 命令创建带值的字段：

```
| makeresults | eval foo="abcdefghijklmnopqrstuvwxyz"
```

结果形式如下所示：

_time	foo
2021-05-21 11:36:57	abcdefghijklmnopqrstuvwxyz

将带有 `offset_field` 参数的 `rex` 命令添加到搜索中以创建名为 `bar` 的字段。您可以使用正则表达式 "(?<firstfive>abcde)" 确定字段 `foo` 中前五个值的位置。例如：

```
| makeresults | eval foo="abcdefghijklmnopqrstuvwxyz" | rex offset_field=bar field=foo "(?<firstfive>abcde)"
```

结果形式如下所示：

_time	bar	firstfive	foo
2021-05-21 11:36:57	firstfive=0-4	abcde	abcdefghijklmnopqrstuvwxyz

您可以使用正则表达式 "(?<middle>fgh)" 确定字段 `foo` 中几个中间值的位置。例如：

```
| makeresults | eval foo="abcdefghijklmnopqrstuvwxyz" | rex offset_field=bar field=foo "(?<middle>fgh)"
```

结果形式如下所示：

_time	bar	foo	middle
2021-05-21 11:36:57	middle=5-7	abcdefghijklmnopqrstuvwxyz	fgh

6. 显示 IP 地址和潜在攻击者的端口

显示 IP 地址和潜在攻击者的端口。

```
sourcetype=linux_secure port "failed password" | rex "\s+(?<ports>port \d+)" | top src_ip ports showperc=0
```

此搜索使用 `rex` 提取端口字段和值。然后，它显示排名最靠前的数据来源 IP 地址 (`src_ip`) 和端口（使用潜在攻击者的搜索返回）。

另请参阅

`extract`, `kvform`, `multikv`, `regex`, `spath`, `xmlkv`

rtorder

描述

对来自实时搜索的事件进行缓冲，以尽可能按时间顺序的升序发出事件。

`rtorder` 命令将新建获取输入事件的流事件缓冲区，按时间顺序的升序将这些事件存储到缓冲区，并按照相同的顺序从缓冲区发出这些事件。不过，仅当当前时间在事件的时间戳之后至少达到 `buffer_span` 所指定的时间跨度时，这些事件才会发出。

如果超出缓冲区的最大大小，也会从缓冲区发出事件。

如果在先前已发出的事件之前就以输入形式接收某个事件，则除非 `discard` 选项设置为 `true`，否则将立即发出无序事件。如果 `discard` 设置为 `true`，将始终放弃无序事件，从而保证输出始终严格以时间顺序的升序排列。

语法

```
rtorder [discard=<bool>] [buffer_span=<span-length>] [max_buffer_size=<int>]
```

可选参数

buffer_span

语法: `buffer_span=<span-length>`

描述: 指定缓冲区的长度。

默认值: 10 秒

discard

语法: `discard=<bool>`

描述: 指定是否始终丢弃无序的事件。

默认值: `false`

max_buffer_size

语法: `max_buffer_size=<int>`

描述: 指定缓冲区的最大大小。

默认值: 50,000 或 `limits.conf` 中 `[search]` 段落的 `max_result_rows` 设置。

示例

示例 1:

保留事件的最后 5 分钟的缓冲区，当这些事件超出 5 分钟时，立即按照时间顺序的升序发出这些事件。如果某事件在 5 分钟后已发出，则将丢弃超过 5 分钟的新接收事件。

```
... | rtorder discard=t buffer_span=5m
```

另请参阅

`sort`

`run`

`run` 命令为 `script` 命令的别名。请参阅“`script` 命令”查看语法和示例。

savedsearch

描述

运行保存的搜索或报表，并返回已保存搜索的搜索结果。若搜索中包含诸如 `$replace_me$` 等替换占位符术语，该搜索处理器将把占位符替换为您指定的字符串。例如：

```
| savedsearch mysearch replace_me="value"
```

语法

```
| savedsearch <savedsearch_name> [<savedsearch-options>...]
```

必要参数

`savedsearch_name`

语法: <string>

描述: 要运行的保存的搜索的名称。

可选参数

`savedsearch-options`

语法: <substitution-control> | <replacement>

描述: 指定是否允许替代。若允许, 请指定在字符串替代表替换中使用的键值对。

`substitution-control`

语法: `nosubstitution=<bool>`

描述: 如果为 `true`, 则不进行字符串替代表替换。

默认值: `false`

`replacement`

语法: <field>=<string>

描述: 在字符串替代表替换中使用的键/值对。

用法

`savedsearch` 命令属于生成命令, 开头必须为前导管道字符。

`savedsearch` 命令将始终运行新的搜索。要重新处理先前运行的搜索的结果, 请使用 `loadjob` 命令。

时间范围

- 如果在时间范围挑选器中指定所有时间, 则 `savedsearch` 命令使用的时间范围将通过保存的搜索进行保存。
- 如果在时间范围挑选器中指定任何其他时间, 则指定的时间范围将由保存的搜索所保存的时间范围所覆盖。

示例

示例 1

运行保存的搜索 "mysecurityquery"。

```
| savedsearch mysecurityquery
```

Example2

运行保存的搜索 "mysearch"。若替换占位符术语 `$replace_me$` 出现于保存的搜索中, 请改用“值”。

```
|savedsearch mysearch replace_me="value"...
```

另请参阅

`search`, `loadjob`

script

描述

调用可以修改或生成搜索结果的外部 `python` 程序。脚本必须在 `commands.conf` 文件中进行声明且必须位于 `$SPLUNK_HOME/etc/apps/<app_name>/bin/` 目录中。执行此脚本须使用 `$SPLUNK_HOME/bin/python`。

如果您在使用 Splunk Cloud 并要安装自定义脚本, 请提交支持工单。在安装之前, 将检查您的脚本, 以确保它符合 Splunk 对安全性、数据安全等对象的要求。

语法

```
script <script-name> [<script-arg>...] [maxinputs=<int>]
```

必要参数

script-name

语法: <string>

描述: 要运行的脚本式搜索命令的名称, 定义于 commands.conf 文件中。

可选参数

maxinputs

语法: maxinputs=<int>

描述: 指定要传递给脚本的输入结果数量。

默认值: 100

script-arg

语法: <string> ...

描述: 要传递给脚本的一个或多个参数。若传递多个参数, 在每个参数之间用空格分隔。

用法

script 命令实际上是调用自定义搜索命令的替代方法。请参阅开发人员门户上《开发人员指南》中的“[为 Splunk Cloud 或 Splunk Enterprise 中的应用程序新建自定义搜索命令](https://dev.splunk.com/enterprise/docs/devtools/customsearchcommands/)”。

以下搜索

| script commandname

等同于

| commandname

注意: 此脚本命令的部分函数已删除。Perl 或 Python 作为参数的显式选择已失效, 将忽略此类参数。若需要写入 Perl 搜索命令, 您需要在 commands.conf 文件中把这些命令声明为 Perl。不建议采用这种方式, 因为您需要确定有关输入和输出格式的大量未公开的事项。另外, 现已不再支持 etc/searchscripts 目录中的脚本采用的显式文件名引用。现在, 所有搜索命令必须在 commands.conf 文件中进行声明。

示例

示例 1:

使用参数 myarg1 和 myarg2 运行 Python 脚本 "myscript", 然后通过电子邮件发送结果。

... | script myscript myarg1 myarg2 | sendemail to=david@splunk.com

scrub

描述

通过使用保持相同单词长度的虚构值替换标识数据, 即用户名、ip 地址、域名等, 使搜索结果匿名。例如, 它可以将字符串 user=carol@adalberto.com 变为 user=aname@mycompany.com。这允许 Splunk 用户在不泄露机密或私人信息的情况下共享日志数据。

更多信息请参阅用法一节。

语法

scrub [public-terms=<filename>] [private-terms=<filename>] [name-terms=<filename>] [dictionary=<filename>] [timeconfig=<filename>] [namespace=<string>]

必要参数

无

可选参数

public-terms
语法: `public-terms=<filename>`
描述: 指定包括不要匿名的公共术语的文件名。

private-terms
语法: `private-terms=<filename>`
描述: 指定包含要匿名的私有术语的文件名。

name-terms
语法: `name-terms=<filename>`
描述: 指定一个包含要匿名的名称的文件名。

dictionary
语法: `dictionary=<filename>`
描述: 指定包含不要匿名的术语词典的文件名，除非那些术语在 `private-terms` 文件中。

timeconfig
语法: `timeconfig=<filename>`
描述: 指定包含要匿名的时间配置的文件名。

namespace
语法: `namespace=<string>`
描述: 指定包含用于匿名的替代文件的应用程序，而不使用内置的匿名文件。

用法

默认情况下，`scrub` 命令使用位于 `$SPLUNK_HOME/etc/anonymizer` 目录下的词典和配置文件。可以通过指定参数到 `scrub` 命令覆盖这些默认文件。参数与 `splunk anonymize` CLI 命令中的设置完全对应。有关详细信息，请发出 `splunk help anonymize` 命令。

您可以将自己的配置文件添加到默认位置。

另外，您可以指定维护词典和配置文件的副本的应用程序。要指定应用程序，请使用 `namespace=<string>` 参数，`<string>` 是出现在 `$SPLUNK_HOME/etc/apps/<app>/anonymizer` 路径下的名称对应的应用名称。

如果 `$SPLUNK_HOME/etc/apps/<app>/anonymizer` 目录不存在，Splunk 软件查找 `$SPLUNK_HOME/etc/slave-apps/<app>/anonymizer` 目录下的文件。

`scrub` 命令可使所有属性匿名，以下划线（`_`）开头（`_raw` 除外）或以 `date_` 开头除外。此外，以下属性不可匿名：`eventtype`、`linecount`、`punct`、`sourcetype`、`timeendpos`、`timestartpos`。

`scrub` 命令遵循默认的 50000 个结果的 `maxresultrows` 限制。此设置记录在 `limits.conf` 文件的 `[searchresults]` 段落中。请参阅 [管理员手册](#) 中的 `limits.conf`。

示例

1. 使用默认文件匿名当前的搜索结果。

```
... | scrub
```

2. 使用指定的私有术语文件匿名当前搜索结果。

此搜索使用位于 `$SPLUNK_HOME/etc/anonymizer` 目录下的 `abc_private-terms` 文件。

```
... | scrub private-file=abc_private-terms
```

搜索

描述

用 `search` 命令从索引中检索事件或在管道中过滤上一个搜索命令的结果。您可通过关键字、加引号的短语、通配符和字段-值表达式从索引中检索事件。`search` 命令在任何搜索开始时暗示。您不需要在搜索条件的开头指定 `search` 命令。

您还可以稍后在搜索管道中使用 `search` 命令来过滤管道中前一个命令的结果。

搜索命令还可以用于子搜索。请阅读《[搜索手册](#)》中的“[关于子搜索](#)”。

[检索事件](#) 后，可以对其应用各种命令，以进行转换、筛选和报告事件。使用竖线（`|`）或管道符将命令应用到检索的事件。

语法

search <logical-expression>

必要参数

<expression>

语法: <logical-expression> | <time-opt> | <search-modifier> | NOT <logical-expression> | <index-expression> | <comparison-expression> | <logical-expression> [OR] <logical-expression>

描述: 包括用于描述要从索引中检索的事件的所有关键字或字段-值对。包括必要的圆括号。在此参数中使用布尔表达式、比较运算符、时间调节器、搜索修饰符或表达式组合。

两个词和表达之间始终隐含有 AND 运算符。例如，`web error` 与 `web AND error` 是一样的。指定 `clientip=192.0.2.255 earliest=-1h@h` 和指定 `clientip=192.0.2.255 AND earliest=-1h@h` 是一样的。所以有明确的理由将其包括在内，否则您不需要指定 AND 运算符。

逻辑表达式选项

<comparison-expression>

语法: <field><comparison-operator><value> | <field> IN (<value-list>)

描述: 将字段和文本值比较或提供可以出现在字段中的值的列表。

<index-expression>

语法: "<string>" | <term> | <search-modifier>

描述: 使用文本字符串和搜索修饰符描述要从索引中检索的事件。

<time-opt>

语法: [<timeformat>] (<time-modifier>)...

描述: 描述搜索的起始时间和结束时间条件的格式。请参阅“时间选项”。

比较表达式选项

<comparison-operator>

语法: = | != | < | <= | > | >=

描述: 您可以在搜索字段/值对时使用比较运算符。含 equal (=) 或 not equal (!=) 运算符的比较表达式比较字符串值。例如，“1”与“1.0”不匹配。含大于或小于运算符 <><= >= 的比较表达式按数字顺序比较两个数字，并按字典顺序比较其他值。请参阅“用法”。

<field>

语法: <string>

描述: 字段的名称。

<value>

语法: <literal-value>

描述: 在比较表达式中，字段的文本数字或字符串值。

<value-list>

语法: (<literal-value>, <literal-value>, ...)

描述: 和 IN 运算符结合使用，以指定两个或两个以上值。例如，用 `error IN (400, 402, 404, 406)`，而不用 `error=400 OR error=402 OR error=404 OR error=406`

索引表达式选项

<string>

语法: "<string>"

描述: 指定要匹配的关键字或带引号的短语。搜索字符串和带引号的字符串（任何非搜索修饰符的字符串）时，Splunk 软件会搜索 `_raw` 字段查找匹配事件或结果。

<search-modifier>

语法: <sourcetype-specifier> | <host-specifier> | <hosttag-specifier> | <source-specifier> | <savedsplunk-specifier> | <eventtype-specifier> | <eventtype-tag-specifier> | <splunk_server-specifier>

描述: 从指定字段或字段标记中搜索事件。例如，搜索主机、数据来源、来源类型、保存的搜索和事件类型其中之一或各项的组合。此外，还可以通过以下格式搜索字段标记: `tag::<field>=<string>`。

- 请参阅《知识管理器手册》中有关使用默认字段进行搜索的内容。
- 请参阅《知识管理器手册》中有关使用标记和字段别名的内容。

<sourcetype-specifier>

语法: sourcetype=<string>
描述: 从指定的来源类型字段搜索事件。

<host-specifier>
 语法: host=<string>
 描述: 从指定的 host 字段搜索事件。

<hosttag-specifier>
 语法: hosttag=<string>
 描述: 搜索拥有该字符串所标记的主机的事件。

<eventtype-specifier>
 语法: eventtype=<string>
 描述: 搜索与指定的事件类型匹配的事件。

<eventtype-tag-specifier>
 语法: eventtype-tag=<string>
 描述: 搜索与该字符串所标记的所有 eventtype 相匹配的事件。

<savedsplunk-specifier>
 语法: savedsearch=<string> | savedsplunk=<string>
 描述: 搜索将通过指定的保存的搜索找到的事件。

<source-specifier>
 语法: source=<string>
 描述: 从指定的 source 字段搜索事件。

<splunk_server-specifier>
 语法: splunk_server=<string>
 描述: 从特定的服务器搜索事件。使用 "local" 代表搜索头。

时间选项

有关时间调节器的列表，请参阅“用于搜索的时间调节器”。

<timeformat>
 语法: timeformat=<string>
 描述: 设置用于 starttime 和 endtime 条件的时间格式。
 默认值: timeformat=%m/%d/%Y:%H:%M:%S。

<time-modifier>
 语法: starttime=<string> | endtime=<string> | earliest=<time_modifier> | latest=<time_modifier>
 描述: 指定开始时间和结束时间（使用相对或绝对时间）。

您还可以使用 earliest 和 latest 属性来指定搜索的绝对和相对时间范围。有关时间修饰符语法的更多信息，请参阅《搜索手册》中的“在搜索中指定时间修饰符”。

starttime
 语法: starttime=<string>
 描述: 事件必须晚于或等于该时间。必须匹配 timeformat。

endtime
 语法: endtime=<string>
 描述: 所有事件都必须早于或等于该时间。

用法

如果 search 命令是搜索中的第一个命令且在第一个管道符之前，那么该命令属于事件生成命令。如果 search 命令进一步沿管道符向下使用，则为可分配的流命令。请参阅“命令类型”。

隐含的搜索命令

search 命令隐含在每次搜索的开头。

如果 search 命令是搜索中的第一个命令，您可以使用术语，如关键字、短语、字段、布尔表达式和比较表达式来准确指定您希望从 Splunk 索引中检索到的事件。如果您未指定字段，则搜索会查找 _raw 字段中的术语。

下面是一些搜索词的示例：

- 关键字: error login, 这和指定以下内容相同: error AND login
- 加引号的短语: "database error"
- 布尔运算符: login NOT (error OR fail)
- 通配符: fail*
- 字段-值对: status=404, status!=404, or status>200

要搜索属于 SPL 运算符或关键字的字段值, 如 country=IN、country=AS、iso=AND 或 state=OR, 您必须将运算符或关键字包括在引号中。例如: country="IN"。

请参阅《[搜索手册](#)》中的[使用搜索命令](#)。

之后在搜索管道中使用搜索命令

除了在所有搜索开始时使用隐含的 search 命令外, 您还可以在稍后的搜索管道中使用 search 命令。您可以使用的搜索词取决于传递到 search 命令的字段。

如果将 _raw 字段传递到 search 命令中, 则您可以使用与 search 命令是搜索中的第一个命令时相同类型的搜索词。

但是, 如果 _raw 字段未传递到 search 命令中, 则必须指定与传递到 search 命令中的字段匹配的字段值对。转换命令 (例如 stats 和 chart) 不会将 _raw 字段传递给管道中的下一个命令。

布尔表达式

使用 search 评估的布尔表达式的顺序为:

1. 括号中的表达式
2. NOT 子句
3. OR 子句
4. AND 子句

此评估顺序和通过 where 命令使用的顺序不同。where 命令会先评估 AND 子句再评估 OR 子句。

比较两个字段

要比较两个字段, 不要用 search 命令指定 index=myindex fieldA=fieldB 或 index=myindex fieldA!=fieldB。当指定一个 comparison_expression 时, search 命令需要将 <field> 和 <value> 比较。search 命令将 fieldB 解释为值, 而不是字段名称。

使用 where 命令比较两个字段。

```
index=myindex | where fieldA=fieldB
```

对于不对等比较, 您可以以几种方式指定条件。

```
index=myindex | where fieldA!=fieldB
```

或

```
index=myindex | where NOT fieldA=fieldB
```

请参阅《[搜索手册](#)》中的[NOT 和 != 间的差异](#)。

用 IN 运算符进行多个字段值比较

如果您想要确定字段是否包含其中一个值, 使用 IN 运算符。

例如, 使用此语法:

```
... error_code IN (400, 402, 404, 406) | ...
```

不使用此语法:

```
... error_code=400 OR error_code=402 OR error_code=404 OR error_code=406 | ...
```

与 search 命令一起使用时, 您可以使用 IN 操作符值列表中的通配符。例如:

```
... error_code IN (40*) | ...
```

您可以将 NOT 运算符和 IN 运算符结合使用。例如：

```
... NOT clientip IN (211.166.11.101, 182.236.164.11, 128.241.220.82) | ...
```

还有一个和您可以和 eval 以及 where 命令结合使用的 IN 函数。当 IN 函数与 eval 和 where 命令一起使用时，值列表中不允许出现通配符。请参阅“[对比和条件函数](#)”。

按字典顺序

基于用于编码计算机内存中的项目的值按字典顺序对这些项目进行排序。在 Splunk 软件中，几乎都是使用 UTF-8 进行编码，这是 ASCII 的超集。

- 数字排在字母前面。根据第一位数字对数字进行排序。例如数字 10、9、70、100，按照字典顺序排序为 10、100、70、9。
- 大写字母排在小写字母前面。
- 符号的排序标准不固定。有些符号排在数字值前面。有些符号排在字母前面或后面。

您可以指定覆盖字典顺序的自定义排序顺序。请参阅博客“[向上排序](#)”！自定义排序顺序。

引号和转义字符

通常，需要在含有空格、逗号、管道符、引号和方括号的短语和字段值两边加上引号。引号必须平衡。开引号后面紧跟非转义结束引号。例如：

- 诸如 error | stats count 之类的搜索将查找含有字符串错误的事件的数量。
- 诸如 ... | search "error | stats count" 之类的搜索将按顺序返回包含错误、管道符、统计信息和计数的原始事件。

另外，如果您不想搜索关键字和短语的默认含义（例如布尔运算符和字段/值对），请在这些关键字和短语两侧加上引号。例如：

- 搜索不表示布尔运算符的关键字 AND： error "AND"
- 搜索此字段/值短语： error "startswith=foo"

反斜杠字符（\）用于转义引号、管道符及其本身。仍可在引号内部扩展反斜杠转义序列。例如：

- 如果在搜索中使用 \| 序列，则会将管道符发送到命令，而不是用管道符来拆分各个命令。
- \" 序列将向命令发送引号字符，例如，搜索引号字符或使用 rex 将引号字符插入字段中。
- \\ 序列可在命令中用作反斜杠字符。

无法识别的反斜杠序列不会改变：

- 例如，搜索字符串中的 \s 可以 \s 形式发送给命令，因为 \s 属于未知的转义序列。
- 但是，在搜索字符串中，\\s 可以 \s 形式发送给命令，因为 \\ 属于已知的转义序列，它将被转换为 \。

使用 TERM() 进行搜索

您可以使用 TERM() 指令强制 Splunk 软件将括号中的所有内容作为索引中的一个术语进行匹配。当关键字包含次要分隔符（如句点和逗号）并且被主分隔符（如空格或逗号）限制时，关键字会更加有帮助。实际上，TERM 不适用于不被主分隔符限制的术语。

请参阅《[搜索手册](#)》中的[使用 CASE 和 TERM 匹配短语](#)。

使用 CASE() 进行搜索

您可以使用 CASE() 指令搜索区分大小写的术语和字段值。

请参阅《[搜索手册](#)》中的[使用 CASE 和 TERM 匹配短语](#)。

示例

这些示例演示了如何使用 search 命令。您可以在《[搜索教程](#)》的“开始搜索”主题中找到更多示例。

1. 字段-值对匹配

此示例将演示源 IP (src) 和目标 IP (dst) 的特定值的字段-值对匹配。

```
src="10.9.165.*" OR dst="10.9.165.8"
```

2. 使用布尔和比较运算符

此示例将演示使用布尔运算符和比较运算符进行字段-值对匹配。搜索代码值为 10 或 29 的事件、任何非 "localhost" 的主机以及一个大于 5 的 xqp 值。

```
(code=10 OR code=29) host!="localhost" xqp>5
```

在本示例中，您还可以使用 IN 运算符，因为您在相同字段中指定两个字段-值对。修订的搜索是：

```
code IN(10, 29) host!="localhost" xqp>5
```

3. 使用通配符

此示例将演示使用通配符进行字段-值对匹配。从具有 HTTP 客户端或服务器错误状态的所有 Web 服务器中搜索事件。

```
host=webserver* (status=4* OR status=5*)
```

在本示例中，您还可以使用 IN 运算符，因为您在相同字段中指定两个字段-值对。修订的搜索是：

```
host=webserver* status IN(4*, 5*)
```

4. 使用 IN 运算符

本示例显示如何使用 IN 运算符指定字段-值对匹配列表。在来自 access.log 文件的事件中，搜索 action 字段中的值 addtocart 或 purchase。

```
sourcetype=access_combined_wcookie action IN (addtocart, purchase)
```

5. 指定辅助搜索

本示例两次使用 search 命令。在每个搜索开始时通过条件 eventtype=web-traffic 暗示 search 命令。稍后在搜索管道中重新使用 search 命令过滤结果。该搜索使用 transaction 命令定义 Web 会话，并搜索包含三个以上事件的用户会话。

```
eventtype=web-traffic | transaction clientip startswith="login" endswith="logout" | search eventcount>3
```

6. 使用 NOT 或 != 比较

使用比较运算符布尔值 "NOT" 搜索与使用 "!=" 比较运算符有所不同。

以下搜索返回除了 fieldA = "value2" 之外的所有内容，包括所有其他字段。

```
NOT fieldA="value2"
```

以下搜索将返回其内存存在 fieldA 且不含 "value2" 值的事件。

```
fieldA!="value2"
```

如果对该值使用通配符，则 NOT fieldA=* 返回其中 fieldA 为 null 或未定义的事件，而 fieldA!=* 从不返回任何事件。

请参阅《[搜索手册](#)》中的 [NOT 和 != 间的差异](#)。

searchtxn

描述

快速返回匹配到交易类型且包含特定文本的交易事件。如果您有 Splunk Cloud 并要定义交易类型，请提交支持工单。

语法

```
| searchtxn <transaction-name> [max_terms=<int>] [use_disjunct=<bool>] [eventsonly=<bool>] <search-string>
```

必要参数

<transaction-name>
语法：<transactiontype>

描述: 定义于 `transactiontypes.conf` 中的交易类型段落名称。

`<search-string>`

语法: `<string>`

描述: 要在交易事件内搜索的术语。

可选参数

`eventsonly`

语法: `eventsonly=<bool>`

描述: 如果为 `true`, 则只检索相关事件但不运行 "`| transaction`" 命令。

默认值: `false`

`max_terms`

语法: `maxterms=<int>`

描述: 介于 1-1000 之间的整数, 用于确定所有字段可以使用多少个唯一字段值。使用较小的值可以提高搜索速度, 有利于更新的值。

默认值: 1000

`use_disjunct`

语法: `use_disjunct=<bool>`

描述: 指定是否应处理 `<search-string>` 中的每个术语, 相当于这些术语在初始搜索中已用运算符 OR 隔开。

默认值: `true`

用法

`searchtxn` 命令属于事件生成命令。请参阅“命令类型”。

生成命令使用前导管道符且应是搜索中的第一个命令。

交易

该命令只能用于由特定字段值, 而非顺序或时间约束连接起来的交易。

假设您在 `transactiontypes.conf.in` 文件中有一个名为 '`email`' 的 `<transactiontype>` 段落。该段落包含下列设置。

- `fields=qid, pid`
- `search=sourcetype=sendmail_syslog to=root`

`searchtxn` 命令将查找所有与 `sourcetype="sendmail_syslog" to=root` 相匹配的事件。

在找到的结果中, 所有包含已定位 `qid` 或 `pid` 的字段将用于进一步搜索相关交易事件。若找不到更多的 `qid` 或 `pid` 值, 将运行生成的搜索:

`sourcetype="sendmail_syslog" ((qid=val1 pid=val1) OR (qid=valn pid=valm)) | transaction name=email | search to=root`

示例

示例 1:

查找 David Smith 发出的所有电子邮件交易。

`| searchtxn email to=root from="David Smith"`

另请参阅

`transaction`

selfjoin

描述

根据您指定的一个或多个字段, 在相同结果集中将搜索结果行与其他搜索结果行结合起来。

语法

`selfjoin [<selfjoin-options>...] <field-list>`

必要参数

<field-list>

语法: <field>...

描述: 要连接的字段或字段列表。

可选参数

<selfjoin-options>

语法: overwrite=<bool> | max=<int> | keepsingle=<bool>

描述: 控制返回的搜索结果集的选项。您可以指定一个或多个这样的选项。

Selfjoin 选项

keepsingle

语法: keepsingle=<bool>

描述: 控制是否保留连接字段中有着唯一值的结果。当 keepsingle=true, 没有其他结果可以连接的搜索结果将保存在输出中。

默认值: false

max

语法: max=<int>

描述: 指示与每个主结果连接的 'other' 结果的最大数量。若 max=0, 则无限制。此参数设置 "其他" 结果最大值。主要结果的最大数为 100,000。

默认值: 1

overwrite

语法: overwrite=<bool>

描述: 当 overwrite=true, 将导致 "其他" 结果的字段覆盖主要结果字段。主要结果用作连接的基础。

默认值: true

用法

自联接更常用于关系数据库表。它们较少用于事件数据。

事件用例的一个示例就是包含进程相关信息的事件，每个进程都有一个父级进程 ID。您可以使用 `selfjoin` 命令将进程相关信息与父级进程相关信息关联起来。

请参阅 "延伸示例"。

基本示例

1: 使用单个字段加入结果

在 'id' 字段上将结果与自身连接。

```
... | selfjoin id
```

延伸示例

以下示例显示 `selfjoin` 命令如何针对简单结果集运作。您可以在自己的 Splunk 实例中按照此示例操作。

此示例逐渐建立搜索。每次向搜索添加内容，搜索都将重新运行，添加的影响显示在结果图表中。`_time` 字段中的值在您每次重新运行搜索时会发生变化。但是，在此示例中，结果表格中的值没有变化，以便我们可以着重了解搜索的更改将如何影响结果。

1. 通过使用 `makergesults` 命令从新建简单的 5 个结果的结果集开始。

```
| makergesults count=5
```

已新建 5 个结果，每个都有相同的时间戳。

_time
2018/1/18 14:38:59

2018/1/18 14:38:59
2018/1/18 14:38:59
2018/1/18 14:38:59
2018/1/18 14:38:59

2. 要更好地追踪每个结果，请使用 `streamstats` 命令以添加对每个结果计数的字段。

```
| makeresults count=5 | streamstats count as a
```

a 字段添加到结果。

_time	a
2018/1/18 14:38:59	1
2018/1/18 14:38:59	2
2018/1/18 14:38:59	3
2018/1/18 14:38:59	4
2018/1/18 14:38:59	5

3. 另外，使用 `eval` 命令将时间戳改为相隔 60 秒。不同的时间戳使此示例更加逼真。

```
| makeresults count=5 | streamstats count as a | eval _time = _time + (60*a)
```

时间戳的分部分已更新。

_time	a
2018/1/18 14:38:59	1
2018/1/18 14:39:59	2
2018/1/18 14:40:59	3
2018/1/18 14:41:59	4
2018/1/18 14:42:59	5

4. 接下来，使用 `eval` 命令以新建字段用作连接结果的字段。

```
| makeresults count=5 | streamstats count as a | eval _time = _time + (60*a) | eval joiner="x"
```

新字段已添加。

_time	a	连接程序
2018/1/18 14:38:59	1	x
2018/1/18 14:39:59	2	x
2018/1/18 14:40:59	3	x
2018/1/18 14:41:59	4	x
2018/1/18 14:42:59	5	x

5. 使用 `eval` 命令新建带数据的部分字段。

`if` 函数与取模 (`modulus`) 运算一起使用以向每个新字段添加不同的数据。取模运算在一个数字与另一个数字相除后，取模运算算出余数。

- `eval b` 命令处理每个结果并执行取模运算。如果 $a/2$ 的余数为 0，在字段 "b" 中写入 "something"，否则将 "nada" 放入字段 "b"。

- eval c 命令处理每个结果并执行取模运算。如果 $a/2$ 的余数是 1，将 "something else" 放入字段 "c"，否则在字段 "c" 中不放任何内容（空值）。

```
| makeresults count=5 | streamstats count as a | eval _time = _time + (60*a) | eval joiner="x" | eval b = if(a%2==0,"something","nada"), c = if(a%2==1,"somethingelse",null())
```

新字段已添加，除 _time 字段外，其他字段按字段名称以字母顺序排列。

_time	a	b	c	连接程序
2018/1/18 14:38:59	1	nada	somethingelse	x
2018/1/18 14:39:59	2	something		x
2018/1/18 14:40:59	3	nada	somethingelse	x
2018/1/18 14:41:59	4	something		x
2018/1/18 14:42:59	5	nada	somethingelse	x

6. 使用 selfjoin 命令连接 joiner 字段的结果。

```
| makeresults count=5 | streamstats count as a | eval _time = _time + (60*a) | eval joiner="x" | eval b = if(a%2==0,"something","nada"), c = if(a%2==1,"somethingelse",null()) | selfjoin joiner
```

结果已连接。

_time	a	b	c	连接程序
2018/1/18 14:39:59	2	something	somethingelse	x
2018/1/18 14:40:59	3	nada	somethingelse	x
2018/1/18 14:41:59	4	something	somethingelse	x
2018/1/18 14:42:59	5	nada	somethingelse	x

7. 要理解 selfjoin 命令如何将结果连接在一起，请移除搜索的 | selfjoin joiner 部分。然后修改搜索以向 b 和 c 字段中的值添加 a 字段的值。

```
| makeresults count=5 | streamstats count as a | eval _time = _time + (60*a) | eval joiner="x" | eval b = if(a%2==0,"something"+a,"nada"+a), c = if(a%2==1,"somethingelse"+a,null())
```

结果现已将行号附加到 b 和 c 字段中的值。

_time	a	b	c	连接程序
2018/1/18 14:38:59	1	nada1	somethingelse1	x
2018/1/18 14:39:59	2	something2		x
2018/1/18 14:40:59	3	nada3	somethingelse3	x
2018/1/18 14:41:59	4	something4		x
2018/1/18 14:42:59	5	nada5	somethingelse5	x

8. 现在将 selfjoin 命令加回搜索中。

```
| makeresults count=5 | streamstats count as a | eval _time = _time + (60*a) | eval joiner="x" | eval b = if(a%2==0,"something"+a,"nada"+a), c = if(a%2==1,"somethingelse"+a,null()) | selfjoin joiner
```

自联接的结果。

_time	a	b	c	连接程序
2018/1/18 14:39:59	2	something2	somethingelse1	x

2018/1/18 14:40:59	3	nada3	somethingelse3	x
2018/1/18 14:41:59	4	something4	somethingelse3	x
2018/1/18 14:42:59	5	nada5	somethingelse5	x

如果某个字段两行中都有值，根据 `_time` 值，最后的结果行优先。实施的连接如下方表格中所示。

结果行	输出	描述
1	行 1 与行 2 连接并返回为行 2。	在字段 b 中，值 nada1 被丢弃，因为行 2 中的值 something2 优先。在字段 c 中，行 2 中没有值。已返回行 1 中的值 somethingelse1。
2	行 2 与行 3 连接并返回为行 3。	因为行 3 包含字段 b 和字段 c 的值，行 3 中的值优先，行 2 中的值被丢弃。
3	行 3 与行 4 连接并返回为行 4。	在字段 b 中，值 nada3 被丢弃，因为在行 4 中的值 something4 优先。在字段 c 中，行 4 中没有值。行 3 中的值 somethingelse3 已返回。
4	行 4 与行 5 连接并返回为行 5。	因为行 5 包含字段 b 和字段 c 的值，行 5 中的值优先，行 4 中的值被丢弃。
5	行 5 没有其他可连接的行。	没有返回其他结果。

(感谢 Splunk 用户 Alacer cogitatus 协助此示例。)

另请参阅

`join`

`sendemail`

描述

用 `sendemail` 命令生成电子邮件通知。您可以将搜索结果发送至指定的电子邮件地址。

您必须有一个可用于发送电子邮件的简单邮件传输协议 (SMTP) 服务器。Splunk 实例不包含 SMTP 服务器。

语法

要求的语法以粗体表示：

```
sendemail to=<email_list>
[from=<email_list>]
[cc=<email_list>]
[bcc=<email_list>]
[subject=<string>]
[format=csv | table | raw]
[inline= <bool>]
[sendresults=<bool>]
[sendpdf=<bool>]
[priority=highest | high | normal | low | lowest]
[server=<string>]
[width_sort_columns=<bool>]
[graceful=<bool>]
[content_type=html | plain]
[message=<string>]
[sendcsv=<bool>]
[use_ssl=<bool>]
[use_tls=<bool>]
[pdfview=<string>]
[papersize=letter | legal | ledger | a2 | a3 | a4 | a5]
[paperorientation=portrait | landscape]
[maxinputs=<int>]
[maxtime=<int> m | s | h | d]
```

[footer=<string>]

必要参数

to

语法: to=<email_list>

描述: 要将搜索结果发送到的电子邮件地址的列表。以引号在列表中列出指定的所有电子邮件地址，并用逗号分隔。例如: "alex@email.com, maria@email.com, wei@email.com"

您可以向其发送电子邮件的域集受“电子邮件设置”页面中的允许的域设置限制。例如，该设置可能会限制您只能将电子邮件发送到组织的电子邮件域中的地址。

有关更多信息，请参阅《告警手册》中的“电子邮件通知操作”。

可选参数

bcc

语法: bcc=<email_list>

描述: 密件抄送行。以引号在列表中列出指定的所有电子邮件地址，并用逗号分隔。

cc

语法: cc=<email_list>

描述: 抄送行。以引号在列表中列出指定的所有电子邮件地址，并用逗号分隔。

content_type

语法: content_type=html | plain

描述: 电子邮件的格式类型。

默认值: content_type 参数的默认值在 alert_actions.conf 文件的 [email] 段落中设置。新的或升级的 Splunk 安装版的默认值是 html。

format

语法: format=csv | raw | table

描述: 指定内联结果采用的格式。

默认值: format 参数的默认值在 alert_actions.conf 文件的 [email] 段落中设置。新的或升级的 Splunk 安装版的默认值是 table。

footer

语法: footer=<string>

描述: 指定替代电子邮件页脚。

默认值: 默认值页脚为:

如果您认为您收到了错误的电子邮件，请参阅您的 Splunk 管理员。
splunk > 机器数据引擎。

若想在页脚强制插入一行，请使用 Shift+Enter。

from

语法: from=<email_list>

描述: 电子邮件地址的 from 行。

默认值: "splunk@<hostname>"

inline

语法: inline=<boolean>

描述: 指定在邮件正文中发送结果还是以附件形式发送。默认情况下，附件以 CSV 文件形式提供。请参阅“用法”一节。

默认值: inline 参数的默认值在 alert_actions.conf 文件的 [email] 段落中设置。新的或升级的 Splunk 安装版的默认值是 false。

graceful

语法: graceful=<boolean>

描述: 若设置为 true，任何原因导致电子邮件发送失败都不会返回错误。剩余的搜索将继续运行，相当于 sendmail 命令不再是搜索的一部分。若 graceful=false 且电子邮件发送失败，搜索将返回一个错误。

默认值: false

maxinputs

语法: maxinputs=<integer>

描述: 设置通过告警发送的搜索结果的最大数量。

默认值: 50000

maxtime

语法: maxtime=<integer>m | s | h | d

描述: 在操作终止前允许操作执行的最长时间。

示例: 2m

默认值: 无限制

message

语法: message=<string>

描述: 指定通过电子邮件发送的消息。

默认值: 默认消息取决于使用 sendemail 命令指定的其他参数。

- 如果 sendresults=false，则消息默认为“搜索完成”。
- 如果 sendresults=true、inline=true 和 sendpdf=false 或 sendcsv=false，则消息默认为“搜索结果”。
- 如果 sendpdf=true 或 sendcsv=true，则消息默认为“附加的搜索结果”。

paperorientation

语法: paperorientation=portrait | landscape

描述: 纸张方向。

默认值: portrait

papersize

语法: papersize=letter | legal | ledger | a2 | a3 | a4 | a5

描述: PDF 默认的纸张大小。可接受值: letter、legal、ledger、a2、a3、a4、a5。

默认值: letter

pdfview

语法: pdfview=<string>

描述: 要以 PDF 形式发送 view.xml 文件的名称。例如，mydashboard.xml、search.xml 或 foo.xml。通常来说，这是仪表板的名称，但也是单个页面应用程序或某些其他对象的名称。仅指定名称。请勿指定文件扩展名。view.xml 文件位于 <SPLUNK_HOME>/data/ui/views 中。

priority

语法: priority=highest | high | normal | low | lowest

描述: 设置电子邮件在电子邮件客户端中显示时的优先级。Lowest 或 5, low 或 4, high 或 2, highest 或 1。

默认值: 正常或 3

sendcsv

语法: sendcsv=<boolean>

描述: 指定是否在电子邮件中以 CSV 文件附件的形式发送结果。

默认值: sendcsv 参数的默认值在 alert_actions.conf 文件的 [email] 段落中设置。新的或升级的 Splunk 安装版的默认值是 false。

sendpdf

语法: sendpdf=<boolean>

描述: 指定是否在电子邮件中以 PDF 附件的形式发送结果。有关生成 PDF 的更多信息，请参阅《报告手册》中的“生成报表和仪表板的 PDF”。

默认值: sendpdf 参数的默认值在 alert_actions.conf 文件的 [email] 段落中设置。新的或升级的 Splunk 安装版的默认值是 false。

sendresults

语法: sendresults=<boolean>

描述: 确定是否应在邮件中包含结果。请参阅“用法”一节。

默认值: sendresults 参数的默认值在 alert_actions.conf 文件的 [email] 段落中设置。新的或升级的 Splunk 安装版的默认值是 false。

server

语法: server=<host>[:<port>]

描述: 如果 SMTP 服务器不是本地服务器，请使用此参数指定发送电子邮件时要使用的 SMTP 邮件服务器。<host> 可以是主机名或 IP 地址。您可以选择指定 Splunk 实例应连接到的 SMTP <port>。

如果设置 use_ssl=true，则必须在 server 参数中同时指定 <host> 和 <port>。

此设置优先于 alert_actions.conf 文件中的 mailserver 设置。mailserver 的默认设置是 localhost:25。

如果告警操作配置为在告警触发时发送电子邮件通知，则 sendemail 命令可能无法使用您在 server 参数中指定的服务器。“电子邮件设置”页面上电子邮件域设置中的值可能会限制您可以使用的服务器。sendemail 命令使用在“电子邮件设置”页面上设置的邮件主机。有关更多信息，请参阅《告警手册》中的“电子邮件通知操作”。

默认值: localhost

subject

语法: subject=<string>
描述: 指定主题行。
默认值: “Splunk 结果”

use_ssl

语法: use_ssl=<boolean>
描述: 指定与 SMTP 服务器通信时是否使用 SSL。如果设置为 true，则还必须在 server 参数中指定 <host> 和 <port>。
默认值: false

use_tls

语法: use_tls=<boolean>
描述: 当与 SMTP 服务器 (starttls) 通信时，指定是否使用 TLS（传输层安全）。
默认值: false

width_sort_columns

语法: width_sort_columns=<boolean>
描述: 仅适用于简单文本电子邮件。指定是否应按宽度对列进行排序。
默认值: true

用法

如果您设置 sendresults=true 和 inline=false 且不指定 format，电子邮件将附上 CSV 文件。

如果在 sendemail 消息中使用字段作为标记，请在使用 sendemail 命令处理之前，先使用 rename 命令删除它们包含的大括号字符（例如 { 和 }）。当大括号字符出现在 \$results\$ 之类的标记中时，sendemail 命令无法解读大括号字符。

权限要求

要使用 sendemail，您的角色必须具有 schedule_search 和 list_settings 权限。

示例

1: 将搜索结果发送到指定的电子邮件

将搜索结果发送到指定的电子邮件。默认情况下，结果以表的形式显示。

```
... | sendemail to="elvis@splunk.com" sendresults=true
```

2: 以表格格式发送搜索结果

以原始格式发送主题为 "myresults" 的搜索结果。

```
... | sendemail to="elvis@splunk.com,john@splunk.com" format=raw subject=myresults server=mail.splunk.com sendresults=true
```

3. 包含 PDF 附件、消息和原始嵌入结果

使用 PDF 附件、消息和原始嵌入结果发送电子邮件通知。

```
index=_internal | head 5 | sendemail to@example@splunk.com server=mail.example.com subject="Here is an email from Splunk" message="This is an example message" sendresults=true inline=true format=raw sendpdf=true
```

设置

描述

对子搜索执行 set 操作。

语法

要求的语法以粗体表示。

```
| set (union | diff | intersect) subsearch1 subsearch2
```

必要参数

`union | diff | intersect`

语法: `union | diff | intersect`

描述: 执行两个子搜索, 然后对两个搜索结果集执行指定的 `set` 操作。

操作	描述
<code>union</code>	返回一个集合, 该集合结合了两个子搜索生成的结果。提供两个子集共有的结果一次。
<code>diff</code>	返回一个集合, 该集合结合了两个子搜索生成的结果, 并排除了两者共有的事件。不指示结果源自哪个子搜索。
<code>intersect</code>	返回一个包含两个子搜索共有的结果的集合。

`subsearch`

语法: `"[<string>]"`

描述: 指定一个子搜索。子搜索必须用方括号括起来。有关子搜索语法的更多信息, 请参阅《搜索手册》中的“关于子搜索”。

用法

`set` 命令属于事件生成命令。请参阅“命令类型”。

生成命令使用前导管道符且应是搜索中的第一个命令。

结果

若结果中包含的所有字段均互相匹配, `set` 命令将把结果视为相同。搜索所生成的部分内部字段, 如 `_serial`, 会随搜索变化。若您对原始事件执行 `set` 命令, 需要过滤出部分内部字段, 而非诸如 `stats` 命令所生成的转换结果。在这些情况中, 不同搜索的所有字段均相同。

输出限制

对于所调用的用于执行 `set` 命令的子搜索, 其生成的结果数量有一个限制值。若超过此限制值, 为 `diff` 命令设置的输入结果将在没有告警的情况下直接遭截断。

如果您有 Splunk Enterprise, 则可以通过编辑 `limits.conf` 文件并更改 `[subsearch]` 段落中的 `maxout` 值来调整此限制。如果此值有所改变, 各种子搜索方案生成的默认结果数量也会改变。请注意, 非常大的值可能会在搜索的‘分析’阶段、在子搜索运行时导致广泛的停滞。该限制值的默认值为 10,000。

只有有着文件系统访问权限的用户, 如系统管理员, 才能编辑配置文件。不要更改或复制默认目录中的配置文件。默认目录中的文件必须保持原样并位于其原始位置。在本地目录进行更改。

请参阅“如何编辑配置文件”。

如果您正在使用 Splunk Cloud, 想要编辑配置文件, 请向 Splunk 支持提交工单。

结果行限制

默认情况下, `set` 命令尝试遍历每个子搜索的最多 50,000 个项目。若任一搜索的输入结果数量超过此限制值, `set` 命令将在没有告警的情况下直接忽略剩余事件。默认情况下, `limits.conf` 中的子搜索的 `maxout` 设置可防止结果数超过此限制。

此最大值受 `limits.conf` 文件中 `[set]` 段落的 `maxresultrows` 设置控制。提高此限制会导致占用更多内存。

只有有着文件系统访问权限的用户, 如系统管理员, 才能编辑配置文件。不要更改或复制默认目录中的配置文件。默认目录中的文件必须保持原样并位于其原始位置。在本地目录进行更改。

请参阅“如何编辑配置文件”。

如果您正在使用 Splunk Cloud, 想要编辑配置文件, 请向 Splunk 支持提交工单。

示例

示例 1:

返回包含字符串 “404” 或 “303”（不同时包含）的 “URL” 的值。

```
| set diff [search 404 | fields url] [search 303 | fields url]
```

示例 2:

返回含有错误 404 和错误 303 的所有 url。

```
| set intersect [search 404 | fields url] [search 303 | fields url]
```

注意：根据默认设置，在子搜索中使用 `fields` 命令时，不会过滤出内部字段。若不希望 `set` 命令比较 `_raw` 或 `_time` 等内部字段，您需要明确地将它们从子搜索中排除：

```
| set intersect [search 404 | fields url | fields - _*] [search 303 | fields url | fields - _*]
```

另请参阅

`append`、`appendcols`、`appendpipe`、`join`、`diff`

setfields

描述

将所有结果的字段值设置为常用值。

对于结果集中的每个事件，将给定字段值设置为指定值。用逗号分隔多个定义。将添加缺失的字段，覆盖现有字段。

无论何时需要更改或定义字段值，您都可以使用更通用的 `eval` 命令。请参阅 `eval` 表达式的用法，设置示例 1 中字段的值。

语法

```
setfields <setfields-arg>, ...
```

必要参数

<setfields-arg>

语法： `string="<string>"`, ...

描述：含带引号值的键值对。若指定了多个键值对，请用逗号隔开每个对。将执行标准键清理。即，所有非字母数字字符将替换为 `'_'`，并将删除前导 `'_'`。

示例

示例 1:

指定 `ip` 和 `foo` 字段的值。

```
... | setfields ip="10.10.10.10", foo="foo bar"
```

用 `eval` 命令来实现：

```
... | eval ip="10.10.10.10" | eval foo="foo bar"
```

另请参阅

`eval`、`fillnull`、`rename`

sichart

摘要索引是一种可用来加速长期运行的、不符合报表加速条件的搜索的方法，如在报表命令之前使用非流式命令的搜索。有关更多信息，请参阅《知识管理器手册》中的“关于报表加速和摘要索引”和“使用摘要索引提高报表效率”。

描述

`chart` 命令的摘要索引版本。`sichart` 命令将使用生成图表可视化所需的统计数据填充摘要索引。例如，它可以新建柱状图、折线图、面积图或饼图。填充好摘要索引后，您可以把 `chart` 命令与跟 `sichart` 命令一起使用的同一道搜索结合使用，即可对摘要索引执行搜索。

语法

要求的语法以粗体表示。

```
sichart
[sep=<string>]
[format=<string>]
[cont=<bool>]
[limit=<int>]
[agg=<stats-agg-term>]
( <stats-agg-term> | <sparkline-agg-term> | "("<eval-expression>"")" )...
[ BY <field> [<bins-options>...] [<split-by-clause>] ] | [ OVER <field> [<bins-options>...] [BY <split-by-clause>] ]
```

有关语法的描述，请参阅“chart 命令”。

用法

支持的函数

您可以将 `sichart` 命令与一系列函数结合使用。有关使用函数的一般信息，请参阅“统计和图表函数”。

- 若需按类别排列的函数列表，请参阅“按类别排列的函数列表”
- 若需按字母顺序排列的函数列表，请参阅“按字母顺序排列的函数列表”

示例

示例 1:

计算稍后对摘要索引结果执行 'chart avg(foo) by bar' 所需的信息。

```
... | sichart avg(foo) by bar
```

另请参阅

`chart`, `collect`, `overlap`, `sirare`, `sistats`, `sitimechart`, `sitop`

sirare

摘要索引是一种可用来加速长期运行的、不符合报表加速条件的搜索的方法，如在报表命令之前使用非流式命令的搜索。有关更多信息，请参阅《知识管理器手册》中的“关于报表加速和摘要索引”和“使用摘要索引提高报表效率”。

描述

`sirare` 命令是 `rare` 命令的摘要索引版本，负责返回一个字段或字段组合最不常用的值。`sirare` 命令使用生成罕见报表所需的统计数据填充摘要索引。填充好摘要索引后，再把常规的 `rare` 命令与 `rare` 命令搜索使用的同一道搜索结合使用，即可针对摘要索引提交报表。

语法

```
sirare [<top-options>...]<field-list> [<by-clause>]
```

必要参数

<field-list>
语法：<string>, ...
描述：以逗号分隔的字段名称的列表。

可选参数

<by-clause>
语法：BY <field-list>
描述：分组所依据的一个或多个字段的名称。

<top-options>
语法：countfield=<string> | limit=<int> | percentfield=<string> | showcount=<bool> | showperc=<bool>
描述：指定待显示值类型和数量的选项。这些选项正是 `rare` 和 `top` 命令使用的 <top-options>。

Top 选项

countfield

语法: countfield=<string>
描述: 要写入计数值的新字段的名称。
默认值: "count"

limit

语法: limit=<int>
描述: 指定要返回的元组的数量 (0 表示返回所有值)。

percentfield

语法: percentfield=<string>
描述: 要写入百分比值的新字段的名称。
默认值: "percent"

showcount

语法: showcount=<bool>
描述: 指定是否使用该元组的计数新建名为 "count" 的字段 (请参阅 "countfield" 选项)。
默认值: true

showpercent

语法: showpercent=<bool>
描述: 指定是否使用该元组的相对流行度新建名为 "percent" 的字段 (请参阅 "percentfield" 选项)。
默认值: true

示例

示例 1:

计算稍后对摘要索引结果执行 'rare foo bar' 所需的信息。

```
... | sirare foo bar
```

另请参阅

collect, overlap, sichtart, sistats, sitimechart, sitop

sistats

描述

您可以使用很多命令新建摘要索引, sistats 命令就是其中之一。可以用很多方法加速搜索运行, 缩短时间, 摘要索引就是其中之一。

sistats 命令是 stats 命令的摘要索引版本, 负责计算数据集的聚合统计数据。

sistats 命令填充摘要索引。然后, 您必须新建报表以生成摘要统计信息。请参阅“用法”一节。

语法

```
sistats [allnum=<bool>] [delim=<string>] ( <stats-agg-term> | <sparkline-agg-term> ) [<by clause>]
```

- 本语法中各参数的相关描述, 请参阅“stats 命令”。
- 欲了解可与 sistats 命令结合使用的函数, 请参阅“统计和图表函数”。

用法

摘要索引与主要索引互不影响。

新建好摘要索引后, 针对该摘要索引运行搜索即可新建报表。借由您用来填充摘要索引的搜索字符串, 用 stats 命令替代 sistats 命令, 即可新建您的报表。

更多信息请参阅《知识管理器手册》中的“关于报表加速和摘要索引”以及“使用摘要索引提高报表效率”。

不适用于特定字段的统计函数

除 count 函数外，当您将 sistats 命令与未应用于特定字段的函数或解析为字段的 eval 表达式配对使用时，搜索头对它的处理方式类似于为它应用了一个针对所有字段的通配符。换句话说，当搜索中有 | sistats avg，则会返回针对 | sistats avg(*) 的搜索结果。

但是，这种“隐式通配符”语法已正式弃用。使通配符采用显式形式。若想让一个函数应用于所有可能的字段时，请使用 | sistats <function>(*)。

内存和 sistats 搜索性能

一对 limits.conf 设置在 sistats 搜索的性能及其在搜索过程中所占用的 RAM 和磁盘内存量之间取得平衡。如果 sistats 一直完成得都很慢，则可以调整这些设置以提高其性能，但同时会增加搜索时的内存使用率，并可能因此导致搜索失败。

如果您有 Splunk Cloud，则需要提交支持工单，申请更改这些设置。

有关更多信息，请参阅《搜索手册》中的“内存和 stats 搜索性能”。

示例

示例 1:

以字符串 "lay" 结尾的唯一字段具有每小时平均值，利用有关该平均值的统计数据新建摘要索引。例如 delay、xdelay、relay 等。

```
... | sistats avg(*lay) BY date_hour
```

若要新建报表，请使用该搜索针对摘要索引运行搜索。

```
index=summary | stats avg(*lay) BY date_hour
```

另请参阅

collect, overlap, sichart, sirare, sitop, sitimechart

有关摘要索引的详细解释和示例，请参阅《知识管理器手册》中的“使用摘要索引提高报表效率”。

sitimechart

摘要索引是一种可用来加速长期运行的、不符合报表加速条件的搜索的方法，如在转换命令之前使用非流式命令的搜索。有关更多信息，请参阅《知识管理器手册》中的“关于报表加速和摘要索引”和“使用摘要索引提高报表效率”。

描述

sitimechart 命令是 timechart 命令的摘要索引版本，负责使用对应的统计数据表格新建时间系柱状图可视化。sitimechart 命令使用生成时间表报表所需的统计数据填充摘要索引。在使用 sitimechart 搜索填充好摘要索引后，使用常规的 timechart 命令，而且采用与 sitimechart 搜索同样的搜索字符串，以针对摘要索引提交报表。

语法

要求的语法以**粗体**表示。

```
sitimechart
[sep=<string>]
[partial=<bool>]
[cont=<bool>]
[limit=<int>]
[agg=<stats-agg-term>]
[<bin-options>...]
<single-agg> [BY <split-by-clause>] | <eval-expression> BY <split-by-clause>
```

指定 sitimechart 命令参数时，必须要有 <single-agg> 或 <eval-expression> BY <split-by-clause> 两者之一。

有关每个参数的说明，请参阅 timechart 命令。

用法

支持的函数

您可以将 `sitimechart` 命令与一系列函数结合使用。有关使用函数的一般信息，请参阅“统计和图表函数”。

- 若需按类别排列的函数列表，请参阅“按类别排列的函数列表”
- 若需按字母顺序排列的函数列表，请参阅“按字母顺序排列的函数列表”

示例

示例 1：

使用 `collect` 命令使用主机组织的 CPU 使用情况统计信息来填充名为 `mysummary` 的摘要索引，

```
... | sitimechart avg(cpu) BY host | collect index=mysummary
```

`collect` 命令将搜索结果添加到指定的摘要索引中。在调用 `collect` 命令前必须创建摘要索引。

然后对相同的搜索使用 `timechart` 命令来生成时间表报表。

```
index=mysummary | timechart avg(cpu) BY host
```

另请参阅

`collect`, `overlap`, `sichart`, `sirare`, `sistats`, `sitop`

sitop

摘要索引是一种可用来加速长期运行的、不符合报表加速条件的搜索的方法，如在报表命令之前使用非流式命令的搜索。有关更多信息，请参阅《知识管理器手册》中的“基于摘要的搜索加速概述”以及“使用摘要索引提高报表效率”。

描述

`sitop` 命令是 `top` 命令的摘要索引版本，负责返回一个字段或字段组合最常用的值。`sitop` 命令使用生成 `top` 报表所需的统计数据填充摘要索引。填充好摘要索引后，再把常规的 `top` 命令与 `sitop` 命令搜索使用的同一道搜索结合使用，即可针对摘要索引提交报表。

语法

```
sitop [<N>] [<top-options>...]<field-list> [<by-clause>]
```

注意：该语法与 `top` 命令的语法完全相同。

必要参数

<field-list>
 语法：<field>, ...
 描述：以逗号分隔的字段名称的列表。

可选参数

<N>
 语法：<int>
 描述：要返回的结果数。

<top-options>
 语法：countfield=<string> | limit=<int> | otherstr=<string> | percentfield=<string> | showcount=<bool> | showperc=<bool> | useother=<bool>
 描述：`sitop` 命令的选项。请参阅 **Top 选项**。

<by-clause>
 语法：BY <field-list>
 描述：分组所依据的一个或多个字段的名称。

Top 选项

countfield
 语法：countfield=<string>
 描述：要将计数值写入其中的新字段的名称。

默认值: count

limit

语法: limit=<int>

描述: 指定要返回的元组的数量（0 表示返回所有值）。

默认值: "10"

otherstr

语法: otherstr=<string>

描述: 如果 useother 为 true, 请指定数值, 并写入到代表所有其他值的行中。

默认值: "OTHER"

percentfield

语法: percentfield=<string>

描述: 要写入百分比值的新字段的名称。

默认值: "percent"

showcount

语法: showcount=<bool>

描述: 指定是否使用该元组的计数新建名为 "count" 的字段（请参阅 "countfield" 选项）。

默认值: true

showperc

语法: showperc=<bool>

描述: 指定是否使用该元组的相对流行度新建名为 "percent" 的字段（请参阅 "percentfield" 选项）。

默认值: true

useother

语法: useother=<bool>

描述: 指定是否添加一行来表示由于限制截止点的原因而未包括的所有值。

默认值: false

示例

示例 1:

计算稍后对摘要索引结果执行 'top foo bar' 所需的信息。

```
... | sitop foo bar
```

示例 2:

使用每天运行计划的搜索中的顶级源 IP 地址填充摘要索引：

```
eventtype=firewall | sitop src_ip
```

另存搜索为 "Summary - firewall top src_ip"。

稍后, 当希望检索它上面的信息和报表时, 对过去一年的数据运行搜索:

```
index=summary search_name="summary - firewall top src_ip" |top src_ip
```

此外, 因为此搜索指定了搜索名称, 所以它会筛选掉由其他摘要索引搜索填入到摘要索引中的其他数据。

另请参阅

collect, overlap, sichart, sirare, sistats, sitimechart

snowincident

snowincident 命令与 Splunk Add-on for ServiceNow 结合使用。

如需此命令的相关信息, 请参阅 *Splunk Add-on for ServiceNow* 中的“将自定义生成搜索命令用于 Splunk Add-on for ServiceNow”。

snowincidentstream

`snowincidentstream` 命令与 Splunk Add-on for ServiceNow 结合使用。

如需此命令的相关信息，请参阅 *Splunk Add-on for ServiceNow* 中的“将自定义流搜索命令用于 Splunk Add-on for ServiceNow”。

snowevent

`snowevent` 命令与 Splunk Add-on for ServiceNow 结合使用。

如需此命令的相关信息，请参阅 *Splunk Add-on for ServiceNow* 中的“将自定义生成搜索命令用于 Splunk Add-on for ServiceNow”。

snoweventstream

`snoweventstream` 命令与 Splunk Add-on for ServiceNow 结合使用。

如需此命令的相关信息，请参阅 *Splunk Add-on for ServiceNow* 中的“将自定义流搜索命令用于 Splunk Add-on for ServiceNow”。

sort

描述

`sort` 命令按指定字段对所有结果进行排序。对于缺少给定字段的结果，如果排序顺序为降序或升序，则认为分别具有该字段的最小或最大可能值。

若 `sort` 命令的第一个参数为数字，则按顺序返回的结果数量不会超过该数字。如果未指定数字，则使用默认限制值 10000。如果指定数字 0，则返回全部结果。有关更多信息，请参阅 `count` 参数。

语法

要求的语法以粗体表示。

```
sort
[<count>]
<sort-by-clause>...
[desc]
```

必要参数

<sort-by-clause>

语法： (- | +) <sort-field>, (- | +) <sort-field> ...

描述： 排序所依据的字段的列表及其排序顺序。用减号 (-) 表示降序，加号 (+) 表示升序。指定一个以上字段时，用逗号分隔各字段名称。请参阅**排序字段选项**。

可选参数

<count>

语法： <int> | limit=<int>

描述： 指定要从排好序的结果中返回的结果数。如果未指定计数，则使用默认限制值 10000。如果指定为 0，则返回全部结果。您可以使用整数指定计数或在计数之前使用标签，例如 `limit=10`。

使用 `sort 0` 可能会对性能产生负面影响，具体取决于返回的结果数。

默认值： 10000

desc

语法： d | desc

描述： 颠倒结果的顺序。如果指定多个字段，将字段中的值按与指定字段的顺序相反的顺序放置。例如，如果指定了三个字段，`desc` 参数会颠倒第一个字段中值的顺序。对于第一个字段中的每组重复值，颠倒第二个字段中相应值的顺序。对于第二个字段中的每组重复值，颠倒第三个字段中相应值的顺序。

排序字段选项

<sort-field>

语法： <field> | auto(<field>) | str(<field>) | ip(<field>) | num(<field>)

描述: 可以使用 <sort-field> 指定的选项。

<field>

语法: <string>

描述: 要排序的字段的名称。

auto

语法: auto(<field>)

描述: 自动确定如何排序字段的值。

ip

语法: ip(<field>)

描述: 将字段的值解释成 IP 地址。

num

语法: num(<field>)

描述: 将字段的值解释成数字。

str

语法: str(<field>)

描述: 将字段的值解释成字符串，并按字母顺序排序各值。

用法

根据默认设置，sort 将尝试自动确定排序的内容。如果字段采用数字值，则排序顺序为数字顺序。如果字段采用 IP 地址值，则排序顺序为 IP 顺序。否则，排序顺序按词典顺序。下面列出了一些具体示例：

- 字母字符串按字典顺序排序。
- 标点字符串按字典顺序排序。
- 数值型数据将按您的预期方式对数字进行排序，可以将排序顺序指定为升序或降序。
- 字母数字字符串将基于第一个字符的数据类型进行排序。如果字符串以数字开头，则此字符串将单独按此数字进行数字排序。否则，字符串按字典顺序排序。
- 由字母数字和标点字符组成的字符串的排序方式与字母数字字符串的排序方式相同。

如果字段采用默认自动模式，则排序顺序将由当时所比较的每个值对来确定。这表示对于某些值对，可能采用字典顺序排序，而对于其他值对，则可能采用数字顺序排序。例如，如果以降序排序 10.1 > 9.1，但 10.1.a < 9.1.a：

按字典顺序

基于用于编码计算机内存中的项目的值按字典顺序对这些项目进行排序。在 Splunk 软件中，几乎都是使用 UTF-8 进行编码，这是 ASCII 的超集。

- 数字排在字母前面。根据第一位数字对数字进行排序。例如数字 10、9、70、100，按照字典顺序排序为 10、100、70、9。
- 大写字母排在小写字母前面。
- 符号的排序标准不固定。有些符号排在数字值前面。有些符号排在字母前面或后面。

自定义排序顺序

您可以指定覆盖字典顺序的自定义排序顺序。请参阅博客“向上排序”！自定义排序顺序

基本示例

1. 使用分类字段选项指定字段类型

先按照 "ip" 值以升序排序对结果排序，再按照 "url" 值以降序排序进行排序。

```
... | sort num(ip), -str(url)
```

2. 指定要排序的结果数

先按照 "size" 字段以降序排序对前 100 个结果进行排序，再按照 "source" 值以升序排序。本示例指定每个字段中的数据类型。"size" 字段包含数字，"source" 字段则包含字符串。

```
... | sort 100 -num(size), +str(source)
```

3. 指定降序和升序排列顺序

先按 “_time” 字段以升序顺序对结果进行排序，再按 “host” 值以降序顺序排序。

```
... | sort _time, -host
```

4. 更改事件的时间格式进行分类

更改事件的时间格式，并按照使用 eval 命令新建的“时间”字段降序排列结果。

```
... | bin _time span=60m | eval Time=strftime(_time, "%m/%d %H:%M %Z") | stats avg(time_taken) AS AverageResponseTime BY Time | sort - Time
```

(感谢 Splunk 用户 Ayn 提供此示例。)

5. 返回最近的事件

返回最近的事件：

```
... | sort 1 -_time
```

6. 使用带 <count> 的标签

您可以使用标签来标识要返回的结果数：返回前 12 个结果，按 “host” 字段降序排序。

```
... | sort limit=12 host
```

延伸示例

1. 指定自定义分类顺序

按字典或数字以外的特定顺序（如星期或月份）对结果表进行排序。例如，假设某个搜索生成了下表：

Day	Total
Friday	120
Monday	93
Tuesday	124
Thursday	356
Weekend	1022
Wednesday	248

按 Day 字段进行排序将返回一个按字母顺序排序的表格，该表格是没有任何意义的。您希望按 Monday 到 Friday 的星期顺序对该表进行排序，而 Weekend 在列表的末尾。

要创建自定义排序顺序，首先需要新建一个定义此顺序的字段，名为 sort_field。然后，您可以按该字段进行排序。

```
... | eval wd=lower(Day) | eval sort_field=case(wd=="monday",1, wd=="tuesday",2, wd=="wednesday",3, wd=="thursday",4, wd=="friday",5, wd=="weekend",6) | sort sort_field | fields - sort_field
```

此搜索使用 eval 命令新建 sort_field 字段，然后使用 fields 命令从最终的结果表中移除 sort_field 字段。

结果形式如下所示：

Day	Total
Monday	93
Tuesday	124
Wednesday	248
Thursday	356

Friday	120
Weekend	1022

(感谢 Splunk 用户 Ant1D 和 Ziegfried 提供此示例。)

有关自定义排序顺序的其他示例，请参阅博客“排序起来！” 。rangemap 命令中的“自定义排序顺序”和“扩展”示例。

另请参阅

reverse

spath

描述

spath 命令可以从结构化数据格式 XML 和 JSON 中提取信息。此命令将此信息储存于一个或多个字段中。该命令还会将所示事件列表中的语法突出显示。

您还可以使用带 eval 命令的 spath() 函数。有关更多信息，请参阅“评估函数”。

语法

spath [input=<field>] [output=<field>] [path=<datapath> | <datapath>]

可选参数

输入

语法: input=<field>

描述: 要读入和提取值的字段。

默认值: _raw

output

语法: output=<field>

描述: 如果指定此参数，则从 path 提取的值将写入此字段名中。

默认值: 如果未指定 output 参数，path 参数的值变成提取的值的字段名称。

path

语法: path=<datapath> | <datapath>

描述: 要提取的值所在的位置路径。位置路径可以指定为 path=<datapath> 或只是 datapath。若没有指定 path=，则第一个没有标签的参数会用作位置路径。位置路径由一个或多个由句点分隔的位置步骤组成。如 'foo.bar.baz'。位置步骤由字段名以及括在大括号内的可选索引组成。索引可以是整数，用于指代数据在数组中的位置（JSON 与 XML 之间会有所不同）；也可以是字符串，用于指代 XML 属性。如果索引指代 XML 属性，则使用 @ 符号指定属性名。

用法

spath 命令属于可分配的流命令。请参阅“命令类型”。

省略的位置路径

当不使用 path 参数时，spath 命令在“自动提取”模式下运行。在“自动提取”模式下，spath 命令将于输入字段的前 5,000 个字符中查找并提取所有字段。若未指定其他输入数据来源，这些字段将默认为 raw。如果提供了路径，则该路径的值将提取到由路径命名的字段，而如果提供的是输出参数，则提取到由输出参数指定的字段。

包含一个或多个位置步骤的位置路径

位置路径包含一个或多个位置步骤，每个位置步骤都具有一个由前面的位置步骤所指定的上下文。顶级位置步骤的上下文隐含为整个 XML 或 JSON 文档的顶级节点。

位置步骤由字段名以及可选数组索引组成。

位置步骤由字段名以及括在大括号内的整数或字符串所表示的可选数组索引组成。

数组索引在 XML 和 JSON 中的含义并不相同。例如，JSON 使用基于零的索引。在 JSON 中，foo.bar{3} 表示 bar 子项的第四个元素，而该子项属于 foo 元素。而在 XML 中，相同的路径则表示第三个 bar 子项，该子项属于 foo。

使用通配符代替数组索引

在 JSON 中，`spath` 命令允许您使用通配符来代替数组索引。现在，您可以使用位置路径 `entities.hashtags{}.text` 获取所有 `hashtags` 的文本，不必依次指定 `entities.hashtags{0}.text`、`entities.hashtags{1}.text` 等等。引用的路径，亦即此处的 `entities.hashtags`，必须引用该路径的一个数组才会具有意义。否则您将收到一个错误，就像常规数组索引一样。

这也适用于 XML。例如，`catalog.book` 和 `catalog.book{}` 是等效的。它们都会给出目录中的所有书籍。

`spath` 命令的替代方法

如果使用 `auto_kv` 或索引时间字段提取，则在索引时间为执行路径提取。

您无需显式使用 `spath` 命令来提供路径。

如果使用 `indexed_extractions=JSON` 或使用 `props.conf` 文件中的 `KV_MODE=JSON`，则无需显式使用 `spath` 命令。

基本示例

1. 指定输出字段和路径

此示例显示了如何指定输出字段和路径。

```
... | spath output=myfield path=foo.bar.baz
```

2. 仅指定 <datapath>

对于 `path` 参数，您可以指定位置路径，无论是否包含 `path=`。在本例中，`<datapath>` 为 `server.name`。

```
... | spath output=myfield server.name
```

3. 基于阵列指定输出字段和路径

例如，您有这个数组。

```
{
  "foo" : [1,2]
}
```

要指定输出字段和路径，请使用此语法。

```
... | spath output=myfield path=foo{1}
```

4. 指定使用嵌套数组的输出字段和路径

例如，您有这个嵌套数组。

```
{
  "foo" : {
    "bar" : [
      {"zoo" : 1},
      {"baz" : 2}
    ]
  }
}
```

要指定此嵌套数组的输出和路径，请使用此语法。

```
... | spath output=myfield path=foo.bar{}.baz
```

5. 指定 XML 属性的输出字段和路径

使用 `@` 符号指定 XML 属性。考虑以下书籍和作者的 XML 列表。

```
<?xml version="1.0">
<purchases>
```

```

<book>
    <author>Martin, George R.R.</author>
    <title yearPublished=1996>A Game of Thrones</title>
    <title yearPublished=1998>A Clash of Kings</title>
</book>
<book>
    <author>Clarke, Susanna</author>
    <title yearPublished=2004>Jonathan Strange and Mr. Norrell</title>
</book>
<book>
    <author>Kay, Guy Gavriel</author>
    <title yearPublished=1990>Tigana</title>
</book>
<book>
    <author>Bujold, Lois McMaster</author>
    <title yearPublished=1986>The Warrior's Apprentice</title>
</book>
</purchases>

```

使用此搜索以返回书籍的路径和出版年份。

```
... | spath output=dates path=purchases.book.title{@yearPublished} | table dates
```

在本示例中，输出是列出书籍所有出版年份的单个多值结果。

延伸示例

1: GitHub

作为许多大型 Git 存储库的管理员，您想：

- 看看谁提交了最多的更改以及提交到了哪个存储库
- 生成每个用户的提交列表

假设你使用 GitHub PushEvent webhook 索引 JSON 数据。您可以使用 spath 命令提取名为 `repository`、`commit_author` 和 `commit_id` 的字段：

```
... | spath output=repository path=repository.url
... | spath output=commit_author path=commits{}.author.name
... | spath output=commit_id path=commits{}.id
```

想要查看谁向存储库提交了最多的更改，运行此搜索。

```
... | top commit_author by repository
```

要查看每个用户的提交列表，运行此搜索。

```
... | stats values(commit_id) by commit_author
```

2: 提取 XML 属性的子集

此示例显示如何从 XML 属性和元素提取值。

```

<vendorProductSet vendorID="2">
    <product productID="17" units="mm" >
        <prodName nameGroup="custom">
            <locName locale="all">API 01209</locName>
        </prodName>
        <desc descGroup="custom">
            <locDesc locale="es">Precios</locDesc>
            <locDesc locale="fr">Prix</locDesc>
            <locDesc locale="de">Preise</locDesc>
            <locDesc locale="ca">Preus</locDesc>
            <locDesc locale="pt">Preços</locDesc>
        </desc>
    </product>

```

要提取 locDesc 元素 (Precios、Prix、Preise 等) 的值, 请使用:

```
... | spath output=locDesc path=vendorProductSet.product.desc.locDesc
```

要提取 locale 属性 (es、fr、de 等) 的值, 请使用:

```
... | spath output=locDesc.locale path=vendorProductSet.product.desc.locDesc{@locale}
```

要提取第四个 locDesc (ca) 的属性, 可使用:

```
... | spath path=vendorProductSet.product.desc.locDesc{4}{@locale}
```

3: 提取和扩展具有多值字段的 JSON 事件

mvexpand 命令仅对一个多值字段有效。此示例将阐述如何将具有多个多值字段的一个 JSON 事件扩展到每个字段值对应一个事件。例如, 假设存在以下事件, 且 sourcetype=json:

```
{
  "widget": {
    "text": [
      {
        "data": "Click here",
        "size": 36
      },
      {
        "data": "Learn more",
        "size": 37
      },
      {
        "data": "Help",
        "size": 38
      }
    ]
  }
}
```

首先, 使用一个搜索从 JSON 中提取字段。因为没有指定 path 参数, spath 命令将在“自动提取”模式中运行并从输入字段中的前 5000 个字符中提取所有字段。然后对字段进行重命名并放在表中。

```
sourcetype=json | spath | rename widget.text.size AS size, widget.text.data AS data | table _time,size,data
```

```
_time          size   data
-----
2018-10-18 14:45:46.000 BST 36 Click here
                           37 Learn more
                           38 Help
```

然后, 使用 eval 函数 mvzip() 新建一个名为 x 的多值字段, 该字段具有 size 和 data 值:

```
sourcetype=json | spath | rename widget.text.size AS size, widget.text.data AS data | eval x=mvzip(data,size) | table _time,data,size,x
```

```
_time          data   size     x
-----
2018-10-18 14:45:46.000 BST Click here 36  Click here,36
                               Learn more 37  Learn more,37
                               Help       38  Help,38
```

接下来, 使用 mvexpand 命令新建基于 x 和 eval 函数 mvindex() 的单个事件, 以重新定义数据和大小值。

```
sourcetype=json | spath | rename widget.text.size AS size, widget.text.data AS data | eval x=mvzip(data,size) | mvexpand x | eval x = split(x,",") | eval data=mvindex(x,0) | eval size=mvindex(x,1) | table _time,data, size
```

```
_time          data   size
-----
2018-10-18 14:45:46.000 BST Click here 36
```

2018-10-18 14:45:46.000 BST Learn more 37
2018-10-18 14:45:46.000 BST Help 38

(感谢 Splunk 用户 G. Zaimi 提供此示例。)

另请参阅

extract, kvform, multikv, regex, rex, xmlkv, xpath

stats

描述

计算结果集的聚合统计，如平均数、计数和总和。类似于 SQL 聚合。如果 stats 命令在没有 BY 子句的情况下使用，将只返回一行，也就是整个进来的结果集的聚合。如果使用了 BY 子句，将为 BY 子句中指定的每个唯一值返回一行。

stats 命令可以用于多个如 SQL 的操作。如果您对 SQL 很熟悉，但对 SPL 较为陌生，请参阅“面向 SQL 用户的 Splunk SPL”。

Stats 和 Eval 命令的区别

stats 命令根据事件中的字段计算统计信息。eval 命令使用现有字段和任意表达式在您的事件中新建字段。

时间	事件
7/18/15 6:20:56 ...PM	182.236.164.11 - - [18/July/2015:18:20:56] " GET /cart.do? action=addtocart&itemId=EST-14...
7/18/15 6:21:04 ...PM	182.236.164.11 - - [18/July/2015:18:21:04] " POST /oldlink? itemId=EST18&JSESSIONID=SD6SL8FF10...

Stats 命令前面的结果集

... | 按主机作为 GET 的 stats count(eval(method="GET"))

host	GET
www1	8413
www2	4654

Stats 命令后面的结果集

语法

简单版：

stats (stats-function(*field*) [AS *field*])... [BY *field-list*]

完成：

要求的语法以粗体表示。

```
| stats  
[partitions=<num>]  
[allnum=<bool>]  
[delim=<string>]  
( <stats-agg-term>... | <sparkline-agg-term>... )
```

[<by-clause>
[<dedup_splitvals>]

必要参数

stats-agg-term

语法: <stats-func>(<evalued-field> | <wc-field>) [AS <wc-field>]

描述: 统计聚合函数。请参阅“Stats 函数”选项。该函数可应用于 Eval 表达式、字段或字段组。使用 AS 子句将结果放入您已指定其名称的新字段内。可以在字段名称中使用通配符字符。更多有关 Eval 表达式的信息，请参阅《搜索手册》中的“Eval 表达式的类型”。

sparkline-agg-term

语法: <sparkline-agg> [AS <wc-field>]

描述: 迷你图聚合函数。使用 AS 子句将结果放入您已指定其名称的新字段内。可以在字段名称中使用通配符字符。

可选参数

allnum

语法: allnum=<bool>

描述: 如果为 true，则计算每个字段的数字统计信息（当且仅当该字段的所有值均为数字时）。

默认值: false

by-clause

语法: BY <field-list>

描述: 分组所依据的一个或多个字段的名称。您不能使用通配符字符指定具有相似名称的多个字段。您必须单独指定每个字段。BY 子句为 BY 子句字段中的每个不同值返回一行。如果未指定 BY 子句，stats 命令只返回一行，即整个进来的结果集的聚合。

dedup_splitvals

语法: dedup_splitvals=<boolean>

描述: 指定是否删除多值 BY 子句中的重复值。

默认值: false

delim

语法: delim=<string>

描述: 指定 list() 或 values() 聚合中各个值的分隔方式。

默认值: 一个单独的空格

partitions

语法: partitions=<num>

描述: 如果指定，基于多线程减少的拆分字段分区输入数据。分区参数在同一台机器上的同一搜索进程中使用多个线程运行简化步骤（在并行简化处理中）。使用重新分布命令与在多个计算机上并行运行简化步骤的并行简化比较。

默认值: 1

Stats 函数选项

stats-func

语法: 语法取决于您使用的函数。请参考下方表格。

描述: 可与 stats 命令一起使用的统计和图表函数。每次调用 stats 命令时都可以使用一个或多个函数。但是，只能使用一个 BY 子句。请参阅“用法”。

以下表格按照函数类型列出支持的函数。使用表格中的链接查看每个函数的介绍和示例。有关使用带有命令的函数的概述，请参见统计和图表函数。

函数类型	支持的函数和语法			
聚合函数	avg() count() distinct_count() estdc() estdc_error()	exactperc<num>() max() median() min() mode()	perc<num>() range() stdev() stdevp()	sum() sumsq() upperperc<num>() var() varp()
事件顺序函数	first()	last()		
多值统计和 chart 函数	list()	values()		
时间函数	earliest()	latest()	rate()	

最早时间	<code>earliest_time()</code>	最新时间	<code>latest_time()</code>	范围	<code>range()</code>
------	------------------------------	------	----------------------------	----	----------------------

Sparkline 函数选项

迷你图是内联图表，位于搜索结果的表单元格内，显示与每一行的主键相关联的基于时间的趋势。有关更多信息，请参阅《[搜索手册](#)》中的“为您的搜索结果添加迷你图”。

sparkline-agg

语法: `sparkline (count(<wc-field>), <span-length>) | sparkline (<sparkline-func>(<wc-field>), <span-length>)`

描述: 迷你图说明符，它将获取某一个字段上的聚合函数的第一个参数和一个可选的时间跨度说明符。如果未使用时间跨度说明符，将根据搜索的时间范围选择合适的时间跨度。如果迷你图的范围未被限制在某一字段，只允许使用 `count` 聚合器。可以在字段名称中使用通配符字符。请参阅“用法”一节。

sparkline-func

语法: `c() | count() | dc() | mean() | avg() | stdev() | stdevp() | var() | varp() | sum() | sumsq() | min() | max() | range()`

描述: 用来生成迷你图值的聚合函数。每个迷你图值都是通过将此聚合应用到落入每个特定时间数据桶范围内的事件而生成。

用法

`stats` 命令是一个转换命令。请参阅“[命令类型](#)”。

结合使用统计函数和 `eval` 表达式

使用 `stats` 命令时，您必须指定一个统计函数或迷你图函数。使用统计函数时，您可以使用 `eval` 表达式作为统计函数的一部分。例如：

```
index=* | stats count(eval(status="404")) AS count_status BY sourcetype
```

不适用于特定字段的统计函数

除 `count` 函数外，当您将 `stats` 命令与未应用于特定字段的函数或解析为字段的 `eval` 表达式配对使用时，搜索头对它的处理方式类似于为它应用了一个针对所有字段的通配符。换句话说，当搜索中有 `| stats avg`，则会返回针对 `| stats avg(*)` 的搜索结果。

但是，这种“隐式通配符”语法已正式弃用。使通配符采用显式形式。若想让一个函数应用于所有可能的字段时，请使用 `| stats <function>(*)`。

数值计算

在计算期间，数字被视为双精度浮点数，受浮点数的所有通常行为制约。如果计算得出浮点特殊值 `Nan`，则在结果中会表示为“`nan`”。正无穷大和负无穷大的特殊值在结果中会分别表示为“`inf`”和“`-inf`”。除以零会导致空字段。

在某些情况下，计算结果包含的位数可能超过浮点数所代表的位数。此时，最低有效数字的精度可能会丢失。有关如何修正此错误的示例，请参阅 `sigfig(X)` 函数的基本示例中的“[示例 2](#)”。

函数和内存用法

就内存而言，某些函数本身就比其他函数占用更多内存。例如，`distinct_count` 函数需要的内存比 `count` 函数大得多。`values` 和 `list` 函数也会消耗大量的内存。

如果您使用的是未结合 `split-by` 字段或结合按字段的低基数 `split-by` 的 `distinct_count` 函数，考虑用 `estdc` 函数替换 `distinct_count` 函数（估计非重复计数）。`estdc` 函数可降低内存使用和减少运行时间。

内存和 `stats` 搜索性能

一对 `limits.conf` 设置在 `stats` 搜索的性能及其在搜索过程中所占用的 RAM 和磁盘内存量之间取得平衡。如果 `stats` 一直完成得都很慢，则可以调整这些设置以提高其性能，但同时会增加搜索时的内存使用率，并可能因此导致搜索失败。

如果您有 Splunk Cloud，则需要提交支持工单，申请更改这些设置。

有关更多信息，请参阅《[搜索手册](#)》中的“[内存和 stats 搜索性能](#)”。

事件顺序函数

基于时间搜索时使用 `first` 和 `last` 函数不会产生精确结果。

- 要根据时间顺序查找第一个值，请使用 `earliest` 函数，而非 `first` 函数。
- 要根据时间顺序查找最后一个值，请使用 `latest` 函数，而非 `last` 函数。

例如，看一下以下搜索。

```
index=test sourcetype=testDb | eventstats first>LastPass as LastPass, last(_time) as mostRecentTestTime BY testCaseId | where startTime==LastPass OR _time==mostRecentTestTime | stats first(startTime) AS startTime, first(status) AS status, first(histID) AS currentHistId, last(histID) AS lastPassHistId BY testCaseId
```

使用 `stats` 和 `eventstats` 命令按照时间顺序排列事件时，替换 `first` 和 `last` 函数。以下搜索显示函数变化。

```
index=test sourcetype=testDb | eventstats latest>LastPass AS LastPass, earliest(_time) AS mostRecentTestTime BY testCaseId | where startTime==LastPass OR _time==mostRecentTestTime | stats latest(startTime) AS startTime, latest(status) AS status, latest(histID) AS currentHistId, earliest(histID) AS lastPassHistId BY testCaseId
```

BY 子句中的通配符

`stats` 命令在 `BY` 子句中的字段值中不支持通配符字符。

例如，您无法指定 `| stats count BY source*`。

重命名字段

不能将一个字段重命名为多个名称。例如，如果您有字段 `A`，您不能将 `A` 重命名为 `B` 或 `C`。以下示例无效。

```
... | stats first(host) AS site, first(host) AS report
```

基本示例

1. 返回每个主机的平均传输速率

```
sourcetype=access* | stats avg(kbps) BY host
```

2. 搜索访问日志，并从 "referer_domain" 的前 100 个值中返回总点击次数

搜索访问日志，并从 "referer_domain" 的前 100 个值中返回总点击次数。`"top"` 命令将返回每个 "referer_domain" 的计数和百分比值。

```
sourcetype=access_combined | top limit=100 referer_domain | stats sum(count) AS total
```

3. 使用通配符字符计算相似字段的每小时平均值

返回以字符串 "lay" 结尾的唯一字段的每小时平均值。例如 `delay`、`xdelay`、`relay` 等。

```
... | stats avg(*lay) BY date_hour
```

4. 删除结果集中的重复结果，并返回唯一结果的总数

删除具有相同 "host" 值的重复结果，并返回剩余结果的总数。

```
... | stats dc(host)
```

5. 在多值 BY 字段中，删除重复的值

对于每个唯一的 `mvfield` 值，返回 `field` 的平均值。删除重复的 `mvfield` 中的值。

```
... | stats avg(field) BY mvfield dedup_splitvals=true
```

延伸示例

1. 比较使用 `stats` 和 `chart` 命令的差异

此示例使用搜索教程中的示例数据，但应该可以与任意格式的 Apache Web 访问日志配合使用。要在您自己的 Splunk 实例中尝试此示例，您必须下载

示例数据并遵循操作以将教程数据导入 Splunk。当运行搜索时，请使用时间范围所有时间。

此搜索使用 stats 命令计算 HTTP 状态代码值和主机的事件数量：

```
sourcetype=access_* | stats count BY status, host
```

BY 子句为 BY 子句字段中的每个唯一值返回一行。在此搜索中，因为 BY 子句中指定两个字段，状态和主机的每个唯一组合列在单独的行上。

统计选项卡中显示的结果如下所示：

status	host	计数
200	www1	11,835
200	www2	11,186
200	www3	11,261
400	www1	233
400	www2	257
400	www3	211
403	www2	228
404	www1	244
404	www2	209

如果您单击可视化选项，status 字段形成 X 轴，host 和 count 字段形成数据系列。此图表的问题在于主机（www1, www2, www3）是字符串且无法用图表测量。

用搜索中的 stats 命令代替 chart 命令。

```
sourcetype=access_* | chart count BY status, host
```

使用 chart 命令，在 BY 子句之后指定的两个字段改变统计选项卡中的结果外观。BY 子句还使结果适合在图表可视化中显示结果。

- 您指定的首个字段被称为 `<row-split>` 字段。在表格中，此字段中的值变成每行的标签。图表中，此字段形成 X 轴。
- 您指定的第二个字段被称为 `<column-split>` 字段。在表格中，此字段的值用作每列的标题。图表中，此字段形成数据系列。

统计选项卡中显示的结果如下所示：

status	www1	www2	www3
200	11,835	11,186	11,261
400	233	257	211
403	0	288	0
404	244	209	237
406	258	228	224
408	267	243	246
500	225	262	246
503	324	299	329
505	242	0	238

如果您单击可视化选项卡，status 字段形成 X 轴，host 字段中的值形成数据系列，Y 轴显示 count。

2. 使用 *Eval* 表达式计算对每个 Web 服务器发出的不同类型的请求数

此示例使用搜索教程中的示例数据，但应该可以与任意格式的 Apache Web 访问日志配合使用。要在您自己的 Splunk 实例中尝试此示例，您必须下载示例数据并遵循操作以将教程数据导入 Splunk。当运行搜索时，请使用时间范围所有时间。

运行以下搜索以使用 stats 命令确定每个 Web 服务器上发生的不同页面请求（GET 和 POST）的数量。

```
sourcetype=access_* | stats count(eval(method="GET")) AS GET, count(eval(method="POST")) AS POST BY host
```

本示例使用 eval 表达式来指定 stats 命令要统计的不同字段值。

- 第一个子句使用 count() 函数来统计包含 method 字段值 GET 的 Web 访问事件。然后，使用 AS 关键字，代表这些结果的字段重命名为 GET。
- 第二个子句对 POST 事件执行相同的操作。
- 然后两种类型的事件计数都由 web 服务器分隔，使用有 host 字段的 BY 子句。

统计选项卡中显示的结果如下所示：

host	GET	POST
www1	8,431	5,197
www2	8,097	4,815
www3	8,338	4,654

您可以用此搜索中的 stats 命令替换 chart 命令。然后您可以单击可视化选项卡以查看结果图表。

3. 按特定字段计算广泛的统计信息

统计每个震级范围出现的地震次数

此搜索使用从 USGS 地震网站下载的近期地震数据。该数据是一个逗号分隔的 ASCII 文本文件，其中包含每次记录的地震的震级（mag）、坐标（经度、纬度）、区域（地点）等。

您可以从 USGS 地震源下载当前 CSV 文件，然后将文件上载到 Splunk 实例。此示例使用过去 30 天的所有地震数据。

运行以下搜索计算发生在每个震级范围内的地震数量。该数据集由 30 天之内的事件组成。

```
source=all_month.csv | chart count AS "Number of Earthquakes" BY mag span=1 | rename mag AS "Magnitude Range"
```

- 此搜索使用 span=1 定义震级字段 mag 的每个范围。
- 然后，使用 rename 命令将字段重命名为 "Magnitude Range"（震级范围）。

统计选项卡中显示的结果如下所示：

震级范围	地震数量
-1-0	18
0-1	2,088
1-2	3,005
2-3	1,026
3-4	194
4-5	452
5-6	109
6-7	11
7-8	3

单击可视化选项卡以图表形式查看结果。

计算一个地区地震震级的统计汇总统计数据

搜索加利福尼亚及周边地区的地震。计算已记录的地震数量。使用统计函数计算最小、最大范围（最小和最大之间的差异）以及最近地震的平均震级。按震级类型列出值。

```
source=all_month.csv place=*California* | stats count, max(mag), min(mag), range(mag), avg(mag) BY magType
```

统计选项卡中显示的结果如下所示：

magType	计数	max (mag)	min (mag)	range (mag)	avg (mag)
H	123	2.8	0.0	2.8	0.549593
MbLg	1	0	0	0	0.000000
Md	1,565	3.2	0.1	3.1	1.056486
Me	2	2.0	1.6	.04	1.800000
Ml	1,202	4.3	-0.4	4.7	1.226622
Mw	6	4.9	3.0	1.9	3.650000
ml	10	1.56	0.19	1.37	0.934000

查找近期地震震级的平均值、标准偏差和方差

搜索加利福尼亚及周边地区的地震。计算已记录的地震数量。使用统计函数计算最近地震震级的平均值、标准偏差和方差。按震级类型列出值。

```
source=usgs place=*California* | stats count mean(mag), stdev(mag), var(mag) BY magType
```

统计选项卡中显示的结果如下所示：

magType	计数	mean (mag)	std (mag)	var (mag)
H	123	0.549593	0.356985	0.127438
MbLg	1	0.000000	0.000000	0.000000
Md	1,565	1.056486	0.580042	0.336449
Me	2	1.800000	0.346410	0.120000
Ml	1,202	1.226622	0.629664	0.396476
Mw	6	3.650000	0.716240	0.513000
ml	10	0.934000	0.560401	0.314049

mean 值应与使用 avg() 计算出的值完全相同。

4. 以表格形式按商品 ID、类型和名称列出已售出的商品，并计算每个产品的收入

此示例将使用搜索教程中的示例数据，并使用字段查找向事件数据中添加更多信息。

- 从添加数据教程下载数据集，并按照说明加载教程数据。
- 从使用字段查找教程下载 CSV 文件，并遵照说明设置查找定义，以添加价格和 productName 到事件。

字段查找配置完毕后，可以使用时间范围所有时间运行此搜索。

新建一张表，用于按商品 ID、类型和名称显示在 Buttercup Games 线上商店售出的商品。此外，还要计算每种产品的收入。

```
sourcetype=access_* status=200 action=purchase | stats values(categoryId) AS Type, values(productName) AS "Product Name", sum(price) AS "Revenue" by productId | rename productId AS "Product ID" | eval Revenue="$ ".toString(Revenue,"commas")
```

本示例先使用 `values()` 函数显示 `categoryId` 和 `productName` 值，这些值与每个 `productId` 相对应。然后再使用 `sum()` 函数计算 `price` 字段的累计值。

另外，此示例还对各个字段重命名，以便更好地显示。`stats` 函数的重命名借由 "AS" 子句以内嵌的方式实现。由于语法禁止重命名 `split-by` 字段，请使用 `rename` 命令更改 `product_id` 字段的名称。

最后，通过管道符把结果传递给 `eval` 表达式，即可重新设置 `Revenue` 字段值的格式，以带美元符号和逗号的货币格式显示这些字段值。

将返回下表结果：

事件	模式	统计信息 (14)	可视化
每页 20 个	格式	预览	
Product ID	Type	Product Name	Revenue
BS-AG-G09	ARCADE		
CU-PG-G06	SPORTS		
DB-SG-G01	STRATEGY		
DC-SG-G02	STRATEGY		
FI-AG-G08	ARCADE		
FS-SG-G03	STRATEGY		
MB-AG-G07	ARCADE		

5. 确定有多少电子邮件来自每个域

此示例使用示例电子邮件数据。您可以将 `sourcetype=cisco:esa` 替换为 `sourcetype` 值并将 `mailfrom` 字段替换为您数据的电子邮件地址字段名，从而实现对任何电子邮件数据运行此搜索。例如：电子邮件可能为 `To`、`From` 或 `Cc`）。

查找组织内的电子邮件有多少是来自 `.com`、`.net`、`.org` 或其他顶级域。

本搜索中的 `eval` 命令包含两个以逗号隔开的表达式。

```
sourcetype="cisco:esa" mailfrom=* | eval accountname=split(mailfrom,"@"), from_domain=mvindex(accountname,-1) | stats count(eval(match(from_domain, "[^\\n\\r\\s]+\\.com"))) AS ".com", count(eval(match(from_domain, "[^\\n\\r\\s]+\\.net"))) AS ".net", count(eval(match(from_domain, "[^\\n\\r\\s]+\\.org"))) AS ".org", count(eval(NOT match(from_domain, "[^\\n\\r\\s]+\\.(com|net|org)"))) AS "other"
```

- 此搜索的第一部分使用 `eval` 命令拆分 `mailfrom` 字段中的电子邮件地址。`from_domain` 被定义为 @ 符号之后的 `mailfrom` 字段的一部分。
 - `split()` 函数用于将 `mailfrom` 字段分成为 `accountname` 的多值字段。`accountname` 的第一个值是 "@" 符号前的所有内容，第二个值则为该符号后的所有内容。
 - `mvindex()` 函数用于将多值字段 `accountname` 中的 `from_domain` 设置为第二个值。
- 然后，通过管道符将结果传递给 `stats` 命令。`count()` 函数用于计算 `eval` 表达式的结果数。
- `eval` 使用 `match()` 函数将 `from_domain` 与查找域中各种后缀的正则表达式进行比较。如果 `from_domain` 的值与正则表达式匹配，则更新 `count`，且是为每个后缀更新，包括 `.com`、`.net` 和 `.org`。其他域名后缀将按 `other` 统计。

统计选项卡中显示的结果如下所示：

.com	.net	.org	其他
4, 246	9, 890	0	3, 543

6. 搜索 Web 访问日志，以确定前 10 个引用域的总点击次数

此示例使用搜索教程中的示例数据，但应该可以与任意格式的 Apache Web 访问日志配合使用。要在您自己的 Splunk 实例中尝试此示例，您必须下载示例数据并遵循操作以将教程数据导入 Splunk。当运行搜索时，请使用时间范围昨天。

这一示例搜索 web 访问日志并返回前 10 引用域的点击数总数。

```
sourcetype=access_* | top limit=10 referer
```

此搜索使用 `top` 命令查找十个最常用的引用站点域，即 `referer` 字段的值。部分事件可能使用 `referer_domain` 而不是 `referer`。`top` 命令将返回每个 `referer` 的计数和百分比值。

referer	count	percent
http://www.buttercupgames.com/category.screen?categoryId=STRATEGY	265	6.032324
http://www.buttercupgames.com	209	4.757569
http://www.google.com	163	3.710448
http://www.buttercupgames.com/category.screen?categoryId=NULL	152	3.460050
http://www.buttercupgames.com/product.screen?productId=SF-BVS-G01	127	2.890963
http://www.buttercupgames.com/category.screen?categoryId=ARCADE	122	2.777145
http://www.buttercupgames.com/category.screen?categoryId=ACCESSORIES	118	2.686092
http://www.buttercupgames.com/product.screen?productId=WC-SH-G04	93	2.117004
http://www.buttercupgames.com/category.screen?categoryId=TEE	93	2.117004
http://www.buttercupgames.com/category.screen?categoryId=SHOOTER	81	1.843842

然后，您可以使用 stats 命令以计算前 10 个引用网站访问的总数。

```
sourcetype=access_* | top limit=10 referer | stats sum(count) AS total
```

sum() 函数在 count 中添加值以生成前 10 个引用网站访问 web 站点的总计次数。

新搜索	
<pre>sourcetype=access_* top limit=10 referer stats sum(count) AS total</pre>	
✓ 4,393 个事件 (18/06/04 0:00:00.000 至 18/06/05 0:00:00.000) 无事件采样 ▾	
事件	模式
每页 20 个	格式
total	预览
1423	

另请参阅

函数

统计和图表函数

命令

eventstats
rare
sistats
streamstats
top

博客

从 stats、eventstats 和 streamstats 开始
搜索命令优于 stats、chart 和 timechart
Smooth 运算符|搜索多个字段值

strcat

描述

连接 2 个或更多字段中的字符串值。把字符串值和文字合并成新的字段。目标字段名称指定于 strcat 命令的末尾。

语法

```
strcat [all|required=<bool>] <source-fields> <dest-field>
```

必要参数

<dest-field>
语法: <string>

描述：根据 `<source-fields>` 参数的定义，这是一个用于保存已连接的字符串值的目标字段。目标字段始终位于一系列数据来源字段之后。

`<source-fields>`

语法： (`<field>` | `<quoted-str>`)...

描述：指定字段名称和您想要连接的文字字符串值。文字值必须用双引号引起起来。

`quoted-str`

语法： "`<string>`"

描述：带引号的字符串文字。

示例： "/" 或 ":"

可选参数

`allrequired`

语法： `allrequired=<bool>`

描述：指定是否所有数据来源字段必须存在于每个事件中，才能将值写入目标字段。若 `allrequired=f`，将始终写入目标字段，并将不存在的数据来源字段按空字符串处理。若 `allrequired=t`，只有当所有数据来源字段均存在时才将值写入目标字段。

默认值： false

用法

`strcat` 命令属于可分配的流命令。请参阅“命令类型”。

示例

示例 1：

添加一个名为 `comboIP` 的字段，该字段将合并数据来源和目标 IP 地址。用正斜线字符分隔各地址。

```
... | strcat sourceIP "/" destIP comboIP
```

示例 2：

添加一个名为 `comboIP` 的字段，该字段将合并数据来源和目标 IP 地址。用正斜线字符分隔各地址。新建一个表示字段值出现次数的图表。

```
host="mailserver" | strcat sourceIP "/" destIP comboIP | chart count by comboIP
```

示例 3：

添加一个名为 `address` 的字段，该字段将把主机和端口值合并成 `<host>::<port>` 格式。

```
... | strcat host "::" port address
```

另请参阅

`Eval`

streamstats

描述

以流化方式将累计的摘要统计数据添加到所有搜索结果。`streamstats` 命令将在看到事件后计算该事件的统计数据。例如，您可以计算某个特定字段的累计值。计算累计值时需要使用所有已处理事件指定字段中的值，直到累算到当前事件为止。

语法

要求的语法以**粗体**表示。

streamstats

```
[reset_on_change=<bool>]  
[reset_before="(<eval-expression>)" ]  
[reset_after="(<eval-expression>)" ]  
[current=<bool>]
```

```
[window=<int>]  
[time_window=<span-length>]  
[global=<bool>]  
[allnum=<bool>]  
<stats-agg-term>...  
[<by-clause>]
```

必要参数

stats-agg-term

语法: <stats-func> (<evalued-field> | <wc-field>) [AS <wc-field>]

描述: 统计聚合函数。请参阅“Stats 函数”选项。该函数可应用于 Eval 表达式、字段或字段组。使用 AS 子句将结果放入您已指定其名称的新字段内。可以在字段名称中使用通配符字符。更多有关 Eval 表达式的信息，请参阅《搜索手册》中的“Eval 表达式的类型”。

可选参数

allnum

语法: allnum=<boolean>

描述: 若为 true, 仅当一字段的所有值均为数字时才计算该字段的数字统计数据。

默认值: false

by-clause

语法: BY <field-list>

描述: 分组所依据的一个或多个字段的名称。

current

语法: current=<boolean>

描述: 若为 true, 该搜索将把给定或当前事件纳入摘要计算。若为 false, 该搜索将使用上一个事件的字段值。

默认值: true

global

语法: global=<boolean>

描述: 仅在设置 window 参数时使用。定义使用单个窗口 global=true 还是基于 by clause 使用单独的窗口。若 global=false 且 window 设置为非零值, 则 by clause 中指定的字段值的每个组都将使用单独的窗口。

默认值: true

reset_after

语法: reset_after="(<eval-expression>)"

描述: 为每个事件生成 streamstats 计算结果之后, reset_after 将指定: 若 eval-expression 返回 true, 将重新设置所有的累计统计数据。eval-expression 的求值结果必须为 true 或 false。eval-expression 可以引用 streamstats 命令返回的字段。若 reset_after 参数与 window 参数结合使用, 则累计统计数据重置后, 窗口也会跟着重置。

默认值: false

reset_before

语法: reset_before="(<eval-expression>)"

描述: 为每个事件生成 streamstats 计算结果之前, reset_before 将指定: 若 eval-expression 返回 true, 将重新设置所有的累计统计数据。eval-expression 的求值结果必须为 true 或 false。若 reset_before 参数与 window 参数结合使用, 则累计统计数据重置后, 窗口也会跟着重置。

默认值: false

reset_on_change

语法: reset_on_change=<bool>

描述: 设置为: 若分组依据字段更改, 将重置所有累计统计数据。重置后, 相当于不曾见过任何旧事件。只有包含所有分组依据字段的事件才能触发重置。仅包含部分分组依据字段的事件将被忽略。若 reset_on_change 参数与 window 参数结合使用, 则累计统计数据重置后, 窗口也会跟着重置。请参阅用法一节。

默认值: false

time_window

语法: time_window=<span-length>

描述: 根据时间指定 streamstats 计算的窗口大小。time_window 参数受事件 _time 字段中的值范围的限制。要使用 time_window 参数, 事件必须按时间升序或降序排序。您可以使用 window 参数和 time_window 参数来指定窗口中的最大事件数。对于 <span-length>, 要指定五分钟, 请使用 time_window=5m。要指定 2 天, 请使用 time_window=2d。

默认值: 无。但是, max_stream_window 属性值 (在 limits.conf 文件中) 适用。默认值为 10,000 个事件。

window

语法: window=<integer>

描述: 指定计算统计信息时使用的事件的数量。

默认值: 0, 表示使用了所有的旧事件和当前事件。

Stats 函数选项

stats-func

语法：语法取决于您使用的函数。请参考下方表格。

描述：可与 streamstats 命令一起使用的统计和图表函数。每次调用 streamstats 命令时都可以使用一个或多个函数。但是，只能使用一个 BY 子句。请参阅“用法”。

以下表格按照函数类型列出支持的函数。使用表格中的链接查看每个函数的介绍和示例。有关使用带有命令的函数的概述，请参见统计和图表函数。

函数类型	支持的函数和语法			
聚合函数	avg() count() distinct_count() estdc() estdc_error()	exactperc<int>() max() median() min() mode()	perc<int>() range() stdev() stdevp()	sum() sumsq() upperperc<int>() var() varp()
事件顺序函数	earliest()	first()	last()	latest()
多值统计和 chart 函数	list(X)	values(X)		

用法

streamstats 命令属于中央流命令。请参阅“命令类型”。

streamstats 命令与 eventstats 命令相似，差别在于前者使用当前事件之前的事件计算应用到各事件的聚合统计数据。若您想将当前事件纳入统计计算，请使用 current=true，亦即默认设置。

streamstats 命令也与 stats 相信，因为 streamstats 命令也计算搜索结果的摘要统计数据。不同于将结果组视为一个整体进行计算的 stats 命令，streamstats 命令将在看到事件后计算该事件的统计数据。

不适用于特定字段的统计函数

除 count 函数外，当您将 streamstats 命令与未应用于特定字段的函数或解析为字段的 eval 表达式配对使用时，搜索头对它的处理方式类似于为它应用了一个针对所有字段的通配符。换句话说，当搜索中有 | streamstats avg，则会返回针对 | stats avg(*) 的搜索结果。

但是，这种“隐式通配符”语法已正式弃用。使通配符采用显式形式。若想让一个函数应用于所有可能的字段时，请使用 | streamstats <function>(*)。

转义字符串值

如果您的 <eval-expression> 包含值而不是字段名称，您必须转义值两侧的引号。

以下示例是了解这个内容的简单方式。使用 makeresults 命令新建 3 个事件。使用 streamstats 命令产生事件的累计数量。然后使用 eval 命令新建简单测试。如果 count 字段的值等于 2，在 test 字段中显示 yes。否则，在 test 字段中显示 no。

```
| makeresults count=3 | streamstats count | eval test;if(count==2,"yes","no")
```

结果出现形式如下所示：

_time	count	test
2017-01-11 11:32:43	1	no
2017-01-11 11:32:43	2	yes
2017-01-11 11:32:43	3	no

若匹配为 true，使用 streamstats 命令重置计数。您必须转义 yes 两侧的括号。下面示例显示完整搜索。

```
| makeresults count=3 | streamstats count | eval test;if(count==2,"yes","no") | streamstats count as testCount  
reset_after="(`match(test,\\"yes\\")`)"
```

以下是另一个示例。您想要查找描述字段中的 session is closed 值。因为值是一个字符串，您必须用括号括起来。然后，您必

须转义这些括号。

```
... | streamstats reset_after="("description==\"session is closed\"")"
```

reset_on_change 参数

若您有一个带 "shift" 字段的数据集，且该字段包含 DAY 值或 NIGHT 值。请运行以下搜索：

```
... | streamstats count BY shift reset_on_change=true
```

若数据集为：

```
shift  
DAY  
DAY  
NIGHT  
NIGHT  
NIGHT  
NIGHT  
DAY  
NIGHT
```

同时运行前述命令与 reset_on_change=true 将生成下列 streamstats 结果：

```
shift, 计数  
DAY, 1  
DAY, 2  
NIGHT, 1  
NIGHT, 2  
NIGHT, 3  
NIGHT, 4  
DAY, 1  
NIGHT, 1
```

内存和最大结果

streamstats 搜索处理器使用两个 limits.conf 设置来确定可以存储在内存中以用于计算统计信息的最大结果数。

maxresultrows 设置为 window 参数指定一个上限。这设置了 streamstats 命令处理器可以存储在内存中的结果行数。max_mem_usage_mb 设置可限制 streamstats 命令用于追踪信息的内存量。

当达到 max_mem_usage_mb 限制时，streamstats 命令处理器会停止将请求的字段添加到搜索结果中。

不要设置 max_mem_usage_mb=0，因为这会取消对 streamstats 命令处理器可以使用的内存量的限制。这可能会导致搜索失败。

前提条件

- 只有具有文件系统访问权限的用户，如系统管理员，才能使用配置文件增加 maxresultrows 和 max_mem_usage_mb 的值。
- 请参阅 Splunk Enterprise 《管理员手册》中的“如何编辑配置文件”了解具体步骤。
- 您可以有几个具有相同名称的配置文件，分散在默认目录、本地目录和应用目录中。请参阅 Splunk Enterprise 《管理员手册》中“在何处放置（或查找）已修改的配置文件”。

不要更改或复制默认目录中的配置文件。默认目录中的文件必须保持原样并位于其原始位置。更改本地目录中的文件。

如果您有 Splunk Cloud 并想更改这些限制，请提交支持工单。

基本示例

1. 计算最后 5 个事件的字段平均值

对于每个事件，计算最后 5 个事件（包括当前事件）的字段 foo 平均值。类似于执行 trendline sma5(foo)

```
... | streamstats avg(foo) window=5
```

2. 用 by 子句计算最后 5 个事件的字段平均值

针对每个事件，为每个仅包含 5 个事件的 bar 值计算 foo 的平均值，5 个事件由窗口大小指定。

```
... | streamstats avg(foo) by bar window=5 global=f
```

3. 为每个事件添加已处理事件的数量计数

此示例将向每个事件添加一个表示到目前为止所看到的事件数的 `count` 字段（含该事件）。例如，它会向第一个事件中添加 1，向第二个事件中添加 2，依此类推。

```
... | streamstats count
```

如果您不希望包含当前事件，那么可以指定：

```
... | streamstats count current=f
```

4. 将基于时间的窗口应用于 `streamstats`

假设 `limits.conf` 文件中的 `max_stream_window` 参数是默认值 10000 个事件。

以下搜索使用五分钟的时间窗口计数事件。

```
... | streamstats count time_window=5m
```

此搜索将为每个事件添加一个计数字段。

- 如果事件按时间降序排列（最近到最早），则计数字段中的值表示未来 5 分钟内的事件数。
- 如果事件按时间升序排列（最早到最近），则计数字段表示先前 5 分钟内的事件数。

如果在基于时间的窗口中有比 `max_stream_window` 参数的值更多的事件，则 `max_stream_window` 参数优先。即使在任何 5 分钟时间内实际有超过 10,000 个事件，计数也永远不会大于 10000。

延伸示例

1. 创建测试事件

可以将 `streamstats` 命令与 `makergesults` 命令一起使用，以创建系列事件。此技巧通常用于测试搜索语法。`eval` 命令用于创建时间不同的事件。您在 `eval` 命令中使用 3600（一小时的秒数）。

```
| makergesults count=5 | streamstats count | eval _time=_time-(count*3600)
```

`streamstats` 命令用于创建 `count` 字段。`streamstats` 命令会在处理每个事件时计算该事件的累计数量。

结果形式如下所示：

_time	count
2020/1/9 15:35:14	1
2020/1/9 14:35:14	2
2020/1/9 13:35:14	3
2020/1/9 12:35:14	4
2020/1/9 11:35:14	5

请注意，时间戳中的小时间隔为 1 小时。

您可以使用 `eval` 命令创建其他字段。

```
| makergesults count=5 | streamstats count | eval _time=_time-(count*3600) | eval age = case(count=1, 25, count=2, 39, count=3, 31, count=4, 27, count=5, null()) | eval city = case(count=1 OR count=3, "San Francisco", count=2 OR count=4, "Seattle", count=5, "Los Angeles")
```

- `eval` 命令用于创建两个新字段：`age` 和 `city`。`eval` 命令使用 `count` 字段中的值。
- `case` 函数采用成对的参数，例如 `count=1, 25`。第一个参数是布尔表达式。当该表达式为 TRUE 时，将返回相应的第二个参数。

搜索结果如下所示：

_time	age	city	计数
2020/1/9 15:35:14	25	San Francisco	1

2020/1/9 15:35:14	25	旧金山	1
2020/1/9 14:35:14	39	西雅图	2
2020/1/9 13:35:14	31	旧金山	3
2020/1/9 12:35:14	27	西雅图	4
2020/1/9 11:35:14		洛杉矶	5

2. 计算摘要统计的快照

此示例使用搜索教程中的示例数据，但应该可以与任意格式的 Apache Web 访问日志配合使用。要在您自己的 Splunk 实例中尝试此示例，您必须下载示例数据并遵循操作以将教程数据导入 Splunk。当运行搜索时，请使用时间范围所有时间。

您想要确定设置时间段内的字节数量。以下搜索将使用前 5 个事件。因为搜索结果通常先显示最近事件，sort 命令用于以升序对 5 个事件进行排序以先查看最旧事件，最后查看最近事件。升序顺序启用 streamstats 命令计算随时间变化的统计信息。

```
sourcetype=access_combined* | head 5 | sort _time
```

时间	事件
18/06/04 18:20:54.000	182.236.164.11 - - [04/Jun/2018:18:20:54] "GET /category.screen?categoryId=ACCESSORIES&JSESSIONID=SD6SL8FF10ADFF53101 HTTP 1.1" 200 3920 "http://www.buttercupgames.com/oldlink?itemId=EST-17" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_7_4) AppleWebKit/536.5 (KHTML, like Gecko) Chrome/19.0.1084.46 Safari/536.5" 648 host = www1 source = tutorialdata.zip.:www1/access.log sourcetype = access_combined_wcookie
18/06/04 18:20:55.000	182.236.164.11 - - [04/Jun/2018:18:20:55] "POST /oldlink?itemId=EST-18&JSESSIONID=SD6SL8FF10ADFF53101 HTTP 1.1" 408 893 "http://www.buttercupgames.com/product.screen?productId=SF-BVS-G01" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_7_4) AppleWebKit/536.5 (KHTML, like Gecko) Chrome/19.0.1084.46 Safari/536.5" 134 host = www1 source = tutorialdata.zip.:www1/access.log sourcetype = access_combined_wcookie
18/06/04 18:20:56.000	182.236.164.11 - - [04/Jun/2018:18:20:56] "GET /cart.do?action=addtocart&itemId=EST-15&productId=BS-AG-G09&JSESSIONID=SD6SL8FF10ADFF53101 HTTP 1.1" 200 2252 "http://www.buttercupgames.com/oldlink?itemId=EST-15" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_7_4) AppleWebKit/536.5 (KHTML, like Gecko) Chrome/19.0.1084.46 Safari/536.5" 506 host = www1 source = tutorialdata.zip.:www1/access.log sourcetype = access_combined_wcookie
18/06/04 18:22:15.000	91.205.189.15 - - [04/Jun/2018:18:22:15] "GET /category.screen?categoryId=SHOOTER&JSESSIONID=SD6SL7FF7ADFF53113 HTTP 1.1" 200 1369 "http://www.google.com" "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/536.5 (KHTML, like Gecko) Chrome/19.0.1084.46 Safari/536.5" 779 host = www2 source = tutorialdata.zip.:www2/access.log sourcetype = access_combined_wcookie
18/06/04 18:22:16.000	91.205.189.15 - - [04/Jun/2018:18:22:16] "GET /oldlink?itemId=EST-14&JSESSIONID=SD6SL7FF7ADFF53113 HTTP 1.1" 200 1665 "http://www.buttercupgames.com/oldlink?itemId=EST-14" "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/536.5 (KHTML, like Gecko) Chrome/19.0.1084.46 Safari/536.5" 159 host = www2 source = tutorialdata.zip.:www2/access.log sourcetype = access_combined_wcookie

向搜索添加 streamstats 命令以生成 5 个事件的字节累计值并按 clientip 组织结果。

```
sourcetype=access_combined* | head 5 | sort _time | streamstats sum(bytes) AS ASimpleSumOfBytes BY clientip
```

当您点击感兴趣的字段列表中的 ASimpleSumOfBytes 字段时，信息窗口显示字节的累计总计，如此图像中所示：

列表 ▾ 格式 每页 20 个 ▾

隐藏字段 所有字段

选定字段
`a host` 2
`a source` 2
`a sourcetype` 1

感兴趣的字段
`a action` 1
`# ASimpleSumOfBytes 5`

`# bytes` 5
`a categoryid` 2
`a clientip` 2
`# date_hour` 1
`# date_mday` 1
`# date_minute` 2
`a date_month` 1
`# date_second` 5
`a date_wday` 1
`# date_year` 1

ASimpleSumOfBytes

5 值, 100% 的事件 选定的 是 否

报表 时段平均值 时段最大值 时段最小值
 上限值 时段上限值 罕见值
 具有此字段的事件

平均: 4040.2 最小值: 1369 最大值: 7065 标准偏差: 2115.94132716387

值	计数	%
1369	1	20%
3034	1	20%
3920	1	20%
4813	1	20%
7065	1	20%

streamstats 命令将统计信息聚合到原始数据，这表明所有原始数据都用于进一步计算。

向搜索添加 table 命令以仅显示 _time、clientip、bytes 和 ASimpleSumOfBytes 字段中的值。

```
sourcetype=access_combined* | head 5 | sort _time | streamstats sum(bytes) as ASimpleSumOfBytes by clientip | table _time, clientip, bytes, ASimpleSumOfBytes
```

每个事件显示了事件的时间戳、clientip 和所用的 bytes 的数量。ASimpleSumOfBytes 字段显示了每个 clientip 的字节累计的摘要。

每页 20 个 ▾ 格式 预览 ▾

_time	clientip	bytes	ASimpleSumOfBytes
2018/06/04 18:20:54	182.236.164.11	3920	3920
2018/06/04 18:20:55	182.236.164.11	893	4813
2018/06/04 18:20:56	182.236.164.11	2252	7065
2018/06/04 18:22:15	91.205.189.15	1369	1369
2018/06/04 18:22:16	91.205.189.15	1665	3034

3. 计算一段时间内的非重复用户累计值

每天您都要对唯一用户进行跟踪，并且您想要跟踪非重复用户的累计数量。此示例将计算非重复用户随时间的累计值。

```
eventtype="download" | bin _time span=1d as day | stats values(clientip) as ips dc(clientip) by day | streamstats dc(ips) as "Cumulative total"
```

bin 命令把时间划分为天。stats 命令计算每天的非重复用户 (clientip) 和用户计数。streamstats 命令查找用户的非重复累计值。

此搜索将返回包含下列信息的表: day、ips、dc(clientip) 和 Cumulative total。

4. 计算每小时的累计总数

此示例使用 streamstats 生成每小时累计总数。

```
... | timechart span=1h sum(bytes) as SumOfBytes | streamstats global=f sum(*) as accu_total_*
```

此搜索返回 3 列: _time、SumOfBytes 和 accu_total_SumOfBytes。

timechart 命令将数据放入跨度为 1 小时的存储桶，并统计每个类别的总计值。timechart 命令还会填充空值，以确保不存在缺失值。然后再使用 streamstats 命令计算累计值。

此示例使用 streamstats 生成类别值的每小时累计值。

```
... | timechart span=1h sum(value) as total by category | streamstats global=f | addtotals | accum Total | rename Total as accu_total
```

5. 计算为特定的 MAC 地址更改 DHCP IP 租用地址的时间

此示例使用 `streamstats` 计算出为 MAC 地址更改 DHCP IP 租用地址的时间，即 54:00:00:00:00:00。

```
source=dhcp MAC=54:00:00:00:00:00 | head 10 | streamstats current=f last(DHCP_IP) as new_dhcp_ip last(_time) as time_of_change by MAC
```

您还可清理演示内容，以显示 DHCP IP 地址更改表和发生的时间。

```
source=dhcp MAC=54:00:00:00:00:00 | head 10 | streamstats current=f last(DHCP_IP) as new_dhcp_ip last(_time) as time_of_change by MAC | where DHCP_IP!=new_dhcp_ip | convert ctime(time_of_change) as time_of_change | rename DHCP_IP as old_dhcp_ip | table time_of_change, MAC, old_dhcp_ip, new_dhcp_ip
```

有关此示例的更多详细信息，请参阅 Splunk 博客中的帖子。

另请参阅

命令

```
accum  
autoregress  
delta  
fillnull  
eventstats  
makeresults  
trendline
```

博客

从 `stats`、`eventstats` 和 `streamstats` 开始

table

描述

`table` 命令将返回仅由参数中指定的字段所形成的表格。各列按字段的指定顺序进行显示。列标题为字段名称。行为字段值。每行表示一个事件。

`table` 命令与 `fields` 命令的相似点在于，可以指定您想要保留在结果中的字段。想以表格形式保留数据时，请使用 `table` 命令。

除了用于显示数据离散值之间的关系中的趋势散点图外，不应对图表使用 `table` 命令。请参阅“用法”。

语法

```
table <wc-field-list>
```

参数

```
<wc-field-list>
```

语法： `<wc-field> ...`

描述：有效字段名称的列表。该列表可以用空格分隔或用逗号分隔。您可以使用星号（*）作为通配符来指定具有相似名称的字段列表。例如，如果要指定所有以“value”开头的字段，则可以使用通配符，例如 `value*.value*`。

用法

`table` 命令是一个转换命令。请参阅“命令类型”。

可视化

除了散点图之外，您不应将 `table` 命令用于可视化。Splunk Web 需要内部字段（以下划线字符开头的字段）来呈现可视化。默认情况下，`table` 命令从结果中删除这些字段。要构建可视化，应该使用 `fields` 命令。`fields` 命令始终保留所有内部字段。

命令类型

`table` 命令属于非流命令。若您需要类似于 `table` 命令的流命令，请使用 `fields` 命令。

字段重命名

`table` 命令禁止您重命名字段，只能指定您想要在结果表中显示的字段。若要重命名某个字段，请在通过管道符将结果传递给 `table` 之前完成。

截断结果

`table` 命令根据 `limits.conf` 文件中的设置截断返回的结果数。在 `[search]` 段落中，如果 `truncate_report` 参数值为 1，则截断返回的结果数。

结果数由 `[search]` 段落中的 `max_count` 参数控制。如果 `truncate_report` 设为 0，则 `max_count` 参数不适用。

示例

示例 1

此示例使用从 USGS 地震网站下载的近期地震数据。该数据是一个逗号分隔的 ASCII 文本文件，其中包含每次记录的地震的震级（mag）、坐标（经度、纬度）、区域（地点）等。

您可以从 [USGS 地震源](#) 下载当前 CSV 文件并作为输入添加到 Splunk 部署。

搜索最近在 California 市区和周边发生的地震，并仅显示地震发生的时间（Datetime）、地点（Region）以及地震的震级（Magnitude）和深度（Depth）。

```
index=usgs_* source=usgs place=*California | table time, place, mag, depth
```

此示例只是将事件的格式重新设置为表形式，并只显示参数所指定的字段。

time	place	mag	depth
2013-11-19T08:58:06.800Z	7km WNW of Cobb, California	0.5	1.6
2013-11-19T09:17:07.800Z	1km ESE of The Geysers, California	0.9	1.1
2013-11-19T09:19:32.200Z	15km ESE of Mammoth Lakes, California	0.7	9.8
2013-11-19T09:44:59.300Z	5km S of Frazier Park, California	1.4	13.9
2013-11-19T09:58:37.600Z	22km ESE of Yosemite Valley, California	1	9.3
2013-11-19T10:18:48.200Z	8km S of Mammoth Lakes, California	0.3	5.2
2013-11-19T10:24:27.800Z	10km WNW of Cobb, California	0.8	2
2013-11-19T10:32:10.500Z	6km NW of The Geysers, California	1.1	2.1
2013-11-19T11:00:42.500Z	8km NW of The Geysers, California	0.8	2.7
2013-11-19T11:14:28.000Z	4km S of Morongo Valley, California	1.6	12.9
2013-11-19T11:23:18.800Z	9km SW of Idyllwild-Pine Cove, California	1.3	18.3
2013-11-19T11:27:04.800Z	9km W of Mammoth Lakes, California	0.5	16.7

示例 2

此示例使用从 USGS 地震网站下载的近期地震数据。该数据是一个逗号分隔的 ASCII 文本文件，其中包含每次记录的地震的震级（mag）、坐标（经度、纬度）、区域（地点）等。

您可以从 [USGS 地震源](#) 下载当前 CSV 文件并作为输入添加到 Splunk 部署。

显示 Northern California 最近发生的每次地震的日期、时间、坐标和震级。

```
index=usgs_* source=usgs place=*California | rename lat as latitude lon as longitude | table time, place, lat*, lon*, mag
```

此示例首先搜索 Northern California (`Region="Northern California"`) 最近发生的所有地震。

然后再通过管道符把这些事件传递给 `rename` 命令，即可把坐标字段的名称从 `lat` 和 `lon` 更改为 `latitude` 和 `longitude`。
(`table` 命令禁止您重命名字段或重新设置其格式，只能指定您想要在结果表中显示的字段。)

最后再通过管道符把结果传递给 `table` 命令，并用 `lat*`、`lon*` 指定两个坐标字段，用 `mag` 指定震级，日期和时间则由 `time` 指定。

每页 20 个 ▾ 格式 ▾ 预览 ▾

time	place	latitude	longitude	mag
2013-11-19T08:58:06.800Z	7km WNW of Cobb, California	38.8405	-122.8043	0.5
2013-11-19T09:17:07.800Z	1km ESE of The Geysers, California	38.774	-122.742	0.9
2013-11-19T09:19:32.200Z	15km ESE of Mammoth Lakes, California	37.595	-118.8155	0.7
2013-11-19T09:44:59.300Z	5km S of Frazier Park, California	34.7742	-118.947	1.4
2013-11-19T09:58:37.600Z	22km ESE of Yosemite Valley, California	37.6792	-119.3305	1
2013-11-19T10:18:48.200Z	8km S of Mammoth Lakes, California	37.5765	-118.9878	0.3
2013-11-19T10:24:27.800Z	10km WNW of Cobb, California	38.841	-122.8385	0.8
2013-11-19T10:32:10.500Z	6km NW of The Geysers, California	38.8095	-122.8138	1.1
2013-11-19T11:00:42.500Z	8km NW of The Geysers, California	38.829	-122.8325	0.8
2013-11-19T11:14:28.000Z	4km S of Morongo Valley, California	34.0057	-116.5807	1.6
2013-11-19T11:23:18.800Z	9km SW of Idyllwild-Pine Cove, California	33.6833	-116.7932	1.3

此示例仅演示如何在 `table` 命令语法中使用星号通配符指定多个字段。

示例 3

此示例将使用教程中的示例数据集，但应与任意格式的 Apache Web 访问日志配合使用。从[添加数据教程](#)下载数据集，并按照说明将示例数据导入到 Splunk 部署。然后，使用时间范围所有时间运行此搜索。

搜索 IP 地址并对它们所属的网络进行分类。

```
sourcetype=access_* | dedup clientip | eval network=if(cidrmatch("192.0.0.0/16", clientip), "local", "other") | table clientip, network
```

此示例搜索 Web 访问数据，并使用 `dedup` 命令删除访问服务器的 IP 地址 (`clientip`) 的重复值。这些结果通过管道符传递给 `eval` 命令，该命令使用 `cidrmatch()` 函数将 IP 地址和子网范围 (192.0.0.0/16) 进行比较。该搜索还使用 `if()` 函数。该函数的作用是：若 `clientip` 的值落在子网范围内，则为 `network` 分配值 `local`。否则，分配到的将是 `network=other`。

最后再通过管道符把结果传递给 `table` 命令，即可将显示的内容限制为非重复的 IP 地址 (`clientip`) 和网络分类 (`network`)：

叠加: 无 ▾

clientip	network
1 192.0.1.51	local
2 192.168.11.33	other
3 192.168.11.44	other
4 192.168.11.35	other
5 192.1.2.40	other
6 192.1.2.35	other
7 192.0.1.39	local
8 192.1.2.52	other
9 192.1.2.39	other
10 192.0.1.30	local

更多示例

示例 1：新建一个包含 `foo` 和 `bar` 字段及以 'baz' 开头的所有字段的表。

```
... | table foo bar baz*
```

另请参阅

[字段](#)

[tags](#)

[描述](#)

用标记批注搜索结果中的指定字段。如果指定了字段，则仅批注这些字段的标记。否则查找所有字段的标记。

[语法](#)

要求的语法以**粗体**表示。

tags
[outputfield=<field>]
[inclname=<bool>]
[inclvalue=<bool>]
<field-list>

必要参数

无。

可选参数

<field-list>
语法: <field> <field> ...
描述: 指定要从中输出标记的字段。标记被写入 outputfield。
默认值: 所有字段

outputfield
语法: outputfield=<field>
描述: 如果有指定, 则所有字段的标记都将写入此新字段。如果未指定, 则会为每个包含标记的字段创建一个新字段。使用命名约定 tag::<field> 将标记写入这些新字段。此外, 还将创建一个名为 tags 的新字段, 该字段列出所有字段中的所有标记。
默认值: 创建新字段, 并将标记写入新字段。

inclname
语法: inclname=true | false
描述: 如果指定了 outputfield, 则会控制是否将事件字段名称以及标记一起添加到输出字段。指定 true 即会包含字段名称。
默认值: false

inclvalue
语法: inclvalue=true | false
描述: 如果指定了 outputfield, 则会控制是否将事件字段值以及标记一起添加到输出字段。指定 true 即会包含事件字段值。
默认值: false

用法

tags 命令属于可分配的流命令。请参阅“命令类型”。

如果指定了 outputfield, 则字段的标记会写入此字段。默认情况下, 标记会以 <field>::<tag> 的格式写入输出字段。

例如, sourcetype::apache。

如果指定了 outputfield, 则 inclname 和 inclvalue 参数会控制是否将字段名称和字段值添加到 outputfield 中。如果 inclname 和 inclvalue 均设置为 true, 则格式为 <field>::<value>::<tag>。

例如, sourcetype::access_combined_wcookie::apache。

示例

1. 使用默认设置的结果

此示例使用搜索教程中的示例数据, 但应该可以与任意格式的 Apache Web 访问日志配合使用。要在您自己的 Splunk 实例中尝试此示例, 您必须下载示例数据并遵循操作以将教程数据导入 Splunk。当运行搜索时, 请使用时间范围所有时间。

此搜索查找 Web 访问事件, 并按主机对这些事件进行计数。

```
sourcetype=access_* | stats count by host
```

结果形式如下所示:

host	count
www1	13628

www2	12912
www3	12992

如果使用 `tags` 命令时未使用任何参数，则两个新字段将添加到结果 `tag` 和 `tag::host` 中。

```
sourcetype=access_* | stats count by host | tags
```

结果形式如下所示：

host	count	tag	tag:host
www1	13628	tag2	tag2
www2	12912	tag1	tag1
www3	12992		

`host=www3` 没有标记。

将 `sourcetype` 字段添加到 `stats` 命令的 `BY` 子句中。

```
sourcetype=access_* | stats count by host sourcetype | tags
```

结果形式如下所示：

host	sourcetype	count	tag	tag:host	tag::sourcetype
www1	access_combined_wcookie	13628	apache tag2	tag2	apache
www2	access_combined_wcookie	12912	apache tag1	tag1	apache
www3	access_combined_wcookie	12992	apache		apache

`tag` 字段列出了包含主机和源类型组合的事件中使用的所有标记。

`tag::host` 字段列出了包含该主机值的事件中使用的所有标记。

`tag::sourcetype` 字段列出了包含该源类型值的事件中使用的所有标记。

2. 指定字段列表

以 `tag::host` 和 `tag::eventtype` 格式为 `host` 和 `eventtype` 字段写入标记。

```
... | tags host eventtype
```

3. 指定输出字段

将所有字段的标记写入到新的字段 `test` 中。

```
... | tags outputfield=test
```

结果形式如下所示：

host	sourcetype	count	test
			apache

www1	access_combined_wcookie	13628	tag2
www2	access_combined_wcookie	12912	apache tag1
www3	access_combined_wcookie	12992	apache

4. 在搜索结果中包括字段名称

将 host 和 sourcetype 字段的标记以 host::<tag> 或 sourcetype::<tag> 格式写入 test 字段。在输出中包含字段名称。

```
... | tags outputfield=test inclname=t host sourcetype
```

另请参阅

相关信息

- 《知识管理器手册》中的“关于标记和别名”
- 《知识管理器手册》中的“在‘搜索’中为字段-值对设置标记”
- 《知识管理器手册》中的“在‘设置’中定义和管理标记”。

命令

Eval

tail

描述

返回最后的 N 个指定结果。事件以相反顺序返回，从结果集的末尾开始。若未指定整数，将返回最后的 10 个事件。

语法

tail [<N>]

必要参数

无。

可选参数

<N>

- 语法: <int>
 描述: 要返回的结果数。
 默认值: 10

示例

示例 1:

以相反顺序返回后 20 个结果。

```
... | tail 20
```

另请参阅

head, reverse

timechart

描述

使用相应的统计信息表创建时间系柱状图。

Timechart 是应用于字段以产生以时间作为 X 轴的图表的统计聚合。您可以指定拆分依据字段，其中拆分字段的每个不同值变成图表中的序列。若使用 eval 表达式，则需要 split-by 子句。用 limit 和 agg 选项可以指定系列筛选。如果提供了显式的 Where 子句，则忽略这两个选项。如果设置为 limit=0，则不会进行任何系列筛选。

语法

要求的语法以粗体表示。

```
timechart
[sep=<string>]
[format=<string>]
[partial=<bool>]
[cont=<bool>]
[limit=<chart-limit-opt>]
[agg=<stats-agg-term>]
[<bin-options>...]
( <single-agg> [BY <split-by-clause>] ) | (<eval-expression>) BY <split-by-clause>
[dedup_splitvals]
```

必要参数

指定 timechart 命令参数时，必须要有 <single-agg> 或 <eval-expression> BY <split-by-clause> 两者之一。

Eval-expression

语法：<math-exp> | <concat-exp> | <compare-exp> | <bool-exp> | <function-call>

描述：代表目标字段值的文字、字段、运算符以及函数的组合。为使这些计算正常运行，值对于运算类型而言必须有效。例如，除了加法之外，如果值不是数字，则算术运算将不会生成有效的结果。此外，如果两个操作数都是字符串，搜索可以连接这两个操作数。如果使用句点 ‘.’ 对值进行连接，则无论这两个值实际是何种数据类型，此搜索都会将其视为字符串。

single-agg

语法：count | <stats-func>(<field>)

描述：应用到一个单一字段的单一聚合，包括已评估字段。对于 <stats-func>，请参阅 Stats 函数选项。不允许使用通配符。必须指定字段，除非使用整体应用于事件的 count 函数。

split-by-clause

语法：<field> (<tc-options>)... [<where-clause>]

描述：指定拆分结果所依据的字段。如果字段为数字值，将应用默认离散化。使用 tc-options 定义离散化。使用 <where-clause> 指定想要在结果中包含的列数。请参阅本主题中的 tc 选项和 where 子句部分。

可选参数

agg=<stats-agg-term>

语法：agg=(<stats-func> (<evalued-field> | <wc-field>) [AS <wc-field>])

描述：统计聚合函数。请参阅“Stats 函数”选项。该函数可应用于 Eval 表达式、字段或字段组。使用 AS 子句将结果放入您已指定其名称的新字段内。可以在字段名称中使用通配符字符。

bin 选项

语法：bins | minspan | span | <start-end> | aligntime

描述：可用于指定离散数据箱或组的选项，以组织信息。bin-options 设置数据箱数量的最大值，而非数据箱的目标值。请参阅本主题中的“Bin 选项部分”。

默认值：bins=100

cont

语法：cont=<bool>

描述：指定图表是否连续。若设置为 true，此“搜索应用程序”将填充时间间隙。

默认值：true

dedup_splitvals

语法：dedup_splitvals=<boolean>

描述：指定是否删除多值 <split-by-clause> 字段中的重复值。

默认值：false

fixedrange

语法：fixedrange=<bool>

描述：指定是否要强制搜索的最早和最晚时间。设置 fixedrange=false 让 timechart 命令可以收缩或扩展至数据集中所有

事件覆盖的时间范围。

默认值: true

format

语法: format=<string>

描述: 当多个数据系列由 split-by 字段进行分隔时，用于构建输出字段名称。format 优先于 sep 并允许您使用 stats 聚合器和函数 (\$AGG\$) 以及 split-by-field 的值 (\$VAL\$) 指定参数化表达式。

limit

语法: limit=(top | bottom) <int>

描述: 指定要返回的 split-by 字段非重复值的数量限制值。若设置为 limit=0，将使用所有的非重复值。设置 limit=N 或 limit=top N 表示 N 将成为 split-by 字段非重复值的上限。设置 limit=bottom N 表示 N 将成为 split-by 字段非重复值的下限。只要 useother 未设置为 false，其他所有值都将划分为 'OTHER'。N 值通过以下方式确定：

- 若未指定单一聚合，将根据 split-by 值聚合结果中各值的总和计算 N 值。例如，就 timechart avg(foo) BY <field> 而言，将把每个 <field> 值的 avg(foo) 值加总，由此来确定 N 值。
- 若指定了多个聚合，将根据每个 <field> 值的频率计算 N 值。例如，就 timechart avg(foo) max(bar) BY <field> 而言，<field> 的最高分值为 <field> 最常用的值。

根据 split-by 字段的值，按字典顺序划分 N 值等同值。例如，'BAR' 优先于 'bar'，后者则优先于 'foo'。请参阅“用法”。

默认值: top 10

partial

语法: partial=<bool>

描述: 控制是否应保留部分时间数据箱。只有第一个和最后一个数据箱可以是部分的。

默认值: True。将保留部分时间数据箱。

sep

语法: sep=<string>

描述: 当多个数据系列由 split-by 字段进行分隔时，用于构建输出字段名称。这相当于把 format 设置为 \$AGG\$<sep>\$VAL\$。

Stats 函数选项

stats-func

语法: 语法取决于您使用的函数。请参阅“用法”。

描述: 可与 timechart 命令一起使用的统计函数。每次调用 timechart 命令时都可以使用一个或多个函数。但是，只能使用一个 BY 子句。

Bin 选项

bins

语法: bins=<int>

描述: 设置离散为数据箱的最大数量。这不是设置数据箱的目标数量。查找可以生成不超过 N 个非重复数据箱的最小数据箱大小。即使您指定的值为 300，生成的数据箱数也可能少很多。

默认值: 100

minspan

语法: minspan=<span-length>

描述: 指定最小的跨度粒度，以自动使用来自数据时间范围的推断跨度。请参阅“用法”。

span

语法: span=<log-span> | span=<span-length> | span=<snap-to-time>

描述: 使用基于日志的跨度、基于时间的跨度长度或对齐到特定时间的跨度来设置每个数据箱的大小。有关这些选项的描述，请参阅“跨度选项”。

数据箱的开始时间可能与您所在时区不匹配。请参阅“用法”。

<start-end>

语法: end=<num> | start=<num>

描述: 设置数字型数据箱的最小和最大范围。在 [start, end] 范围之外的数据将遭丢弃。

aligntime

语法: aligntime=(earliest | latest | <time-specifier>)

描述: 将数据箱时间和基本 UNIX 时间 (epoch 0) 以外的时间对齐。仅当执行基于时间离散化时 aligntime 选项有效。如果 span 是以日期、月份或年份为单位，请忽略。

跨度选项

<log-span>

语法: [<num>] log [<num>]

描述: 设置为基于对数的跨度。第一个数字为系数, 第二个数字为底。若提供第一个数字, 该数字必须是大于等于 1.0 且小于底的实数。若提供底, 则底必须是大于 1.0 的实数(严格大于 1)。

<span-length>

语法: <int>[<timescale>]

描述: 每个数据箱的跨度(基于时间)。如果提供了 timescale, 则将其用作时间范围。否则, 这是一个绝对的数据箱长度。

<timescale>

语法: <sec> | <min> | <hr> | <day> | <week> | <month> | <subseconds>

描述: 时间刻度单位。

默认值: <sec>

Timescale	有效语法	描述
<sec>	s sec secs second seconds	时间刻度(秒)。
<min>	m min mins minute minutes	时间刻度(分钟)。
<hr>	h hr hrs hour hours	时间刻度(小时)。
<day>	d day days	时间刻度(天)。
<week>	w week weeks	时间刻度(周)。
<month>	mon month months	时间刻度(月)。
<subseconds>	us ms cs ds	单位为微秒(us)、毫秒(ms)、百分之一秒(cs)或十分之一秒(ds)的时间刻度。

<snap-to-time>

语法: [+|-] [<time_integer>] <relative_time_unit>@<snap_to_time_unit>

描述: 各数据箱的跨度, 基于相对时间并对齐到时间单位。<snap-to-time> 必须包含 relative_time_unit、@ 符号和 snap_to_time_unit。加号 (+) 或减号 (-) 表示的偏移是可选的。如果未指定 <time_integer>, 则默认为 1。例如, 如果您指定 w 作为 relative_time_unit, 则假设为 1 周。

该选项只与周时间刻度单位一起使用。不能其他时间刻度单位一起使用, 如分钟或季度。

tc 选项

<tc-option> 是 <split-by-clause> 的一部分。

tc-option

语法: <bin-options> | usenull=<bool> | useother=<bool> | nullstr=<string> | otherstr=<string>

描述: 用于通过字段控制拆分行为的时间图表选项。

bin 选项

请参阅本主题中的“Bin 选项部分”。

nullstr

语法: nullstr=<string>

描述: 如果 usenull=true, 为针对不包含拆分依据字段的事件创建的系列指定标签。

默认值: NULL

otherstr

语法: otherstr=<string>

描述: 如果 useother=true, 为在表和图表中创建的系列指定标签。

默认值: OTHER

usenull

语法: usenull=<bool>

描述: 用于控制是否为不包含 split-by 字段的事件创建一个系列。系列的标签由 nullstr 选项控制。

默认值: true

useother

语法: useother=<bool>

描述: 使用 <agg>、<limit> 和 <where-clause> 选项指定要将哪个系列包含在结果表中。useother 选项指定是否要将所有不包括在结果表中的系列合并成一个新系列。如果 useother=true, 系列的标签由 otherstr 选项控制。

默认值: true

Where 子句

<where-clause> 是 <split-by-clause> 的一部分。<where-clause> 由两部分组成，单一聚合和一些选项。请参阅“where 子句示例”。

Where 子句

语法: <single-agg> <where-comp>

描述: 指定在 <tc-by-clause> 中给出某一字段时包括特定数据系列的条件。此选项最常见的用法是，在系列选择中查找数据中的峰值而不是分布总和。默认值将根据曲线下的区域查找前十个系列。或者，可将总和替换为最大值，以查找具有十个最高峰值的系列。本质上，默认值和指定 where sum in top10 是相同的。<where-clause> 和 where 命令没有相关性。

<where-comp>

语法: <wherein-comp> | <wherethresh-comp>

描述: 指定系列的分组或阈值。

<wherein-comp>

语法: (in |notin) (top | bottom)<int>

描述: 分组条件，要求聚合系列值在或不在某些最大值或最小值分组内。

<wherethresh-comp>

语法: (< | >) [" "] <num>

描述: 阈值条件，要求聚合系列值大于或小于某些数字阈值。您可以指定阈值，无论符号和数字之间是否有空格。

用法

timechart 命令是一个转换命令。请参阅“命令类型”。

bins 和 span 参数

timechart 命令接受 bins 参数或 span 参数。如果您同时指定 bins 和 span，则使用 span。忽略 bins 参数。

若未指定 bins 或 span，timechart 命令将使用默认的 bins=100。

默认时间跨度

若您使用时间范围挑选器中预定义的时间范围，且未指定 span 参数，下表将显示所使用的默认时间跨度。

时间范围	默认跨度
过去 15 分钟	10 秒
过去 60 分钟	1 分钟
过去 4 小时	5 分钟
过去 24 小时	30 分钟
Last 7 days	1 天
Last 30 days	1 天
去年	1 个月

(感谢 Splunk 用户 MuS 和 Martin Mueller 帮忙搜集该默认时间跨度信息。)

指定最小跨度时使用的跨度

当您指定 minspan 值时，用于搜索的跨度必须大于等于下表中其中一个跨度阈值。例如，如果您指定 minspan=15m，相当于 900 秒。可使用的最小跨度是 1800 秒或者 30 分钟。

跨度阈值	等价时间
1 秒	

5 秒	
10 秒	
30 秒	
60 秒	1 分钟
300 秒	5 分钟
600 秒	10 分钟
1,800 秒	30 分钟
3,600 秒	1 小时
86,400 秒	1 天
2,592,000 秒	30 天

数据箱时间跨度和本地时间

span 参数始终四舍五入为第一个数据箱的开始日期。不保证时间 timechart 命令使用的数据箱开始时间与您的本地时区对应。某种程度上，这是由于不同时区的夏令时不同。要使用日期边界，使用 span=1d。请勿使用 span=86400s、span=1440m 或 span=24h。

数据箱时间跨度与 per_* 函数

per_day()、per_hour()、per_minute() 和 per_second() 函数属于聚合器函数，不负责设置结果图表的时间跨度。这些函数用于在未提供显式跨度的情况下获取数据的一致刻度。生成的跨度取决于搜索的时间范围。

例如，per_hour() 将对字段值进行转换，结果为小时率，或 sum() /<hours in the span>。如果图表跨度最终为 30m，则为 sum()*2。

若您希望跨度为 1h，您仍须在搜索中指定参数 span=1h。

您可以对某字段执行 per_hour()，而对相同搜索中的另一个字段执行 per_minute()（或函数的任意组合）。

亚秒级数据箱时间跨度

亚秒级 span 时间刻度（由秒 (ds)、百分之一 (cs)、毫秒 (ms) 或微秒 (us) 组成的时间跨度）应为一秒内可均分的数字。例如，1s = 1000ms。这意味着有效的毫秒 span 值为 1、2、4、5、8、10、20、25、40、50、100、125、200、250 或 500ms。另外，不允许设置 span = 1000ms。改用 span = 1s。

Split-by 字段

若指定 split-by 字段，请确保您在指定该字段前先指定了 bins 和 span 参数。若您在指定 split-by 字段后才指定这两个参数，Splunk 软件将假定您希望在 split-by 字段而非时间轴上控制数据箱。

若使用 chart 或 timechart，则无法将函数中指定的字段用作 split-by 字段。例如，您不能运行以下代码：

```
... | chart sum(A) by A span=log2
```

但是，您可以通过 eval 表达式实现此操作，例如：

```
... | eval A1=A | chart sum(A) by A1 span=log2
```

支持的函数

您可以将 timechart 命令与一系列函数结合使用。有关使用函数的一般信息，请参阅“统计和图表函数”。

- 若需按类别排列的函数列表，请参阅“按类别排列的函数列表”
- 若需按字母顺序排列的函数列表，请参阅“按字母顺序排列的函数列表”

函数和内存用法

就内存而言，某些函数本身就比其他函数占用更多内存。例如，distinct_count 函数需要的内存比 count 函数大得多。values 和 list 函数也会消耗大量的内存。

如果您使用的是未结合 `split-by` 字段或结合按字段的低基数 `split-by` 的 `distinct_count` 函数，考虑用 `estdc` 函数替换 `distinct_count` 函数（估计非重复计数）。`estdc` 函数可降低内存使用和减少运行时间。

按字典顺序

基于用于编码计算机内存中的项目的值按字典顺序对这些项目进行排序。在 Splunk 软件中，几乎都是使用 UTF-8 进行编码，这是 ASCII 的超集。

- 数字排在字母前面。根据第一位数字对数字进行排序。例如数字 10、9、70、100，按照字典顺序排序为 10、100、70、9。
- 大写字母排在小写字母前面。
- 符号的排序标准不固定。有些符号排在数字值前面。有些符号排在字母前面或后面。

您可以指定覆盖字典顺序的自定义排序顺序。请参阅博客“向上排序”！自定义排序顺序。

基本示例

1. 为每个 “host” 绘制平均 “CPU” 与平均 “MEM” 产品的图表

为每个 “host” 计算平均 “CPU” 与平均 “MEM” 之产品。

```
... | timechart span=1m eval(avg(CPU) * avg(MEM)) BY host
```

2. 按处理器绘制 `cpu_seconds` 平均值的图表

此示例使用包含统计函数的 `eval` 表达式，`avg` 用于计算 `cpu_seconds` 字段的平均值，舍入到 2 个小数位。结果按 `processor` 字段中的值组织。当您使用带 `timechart` 命令的 `eval` 表达式时，您也必须使用 `BY` 子句。

```
... | timechart eval(round(avg(cpu_seconds),2)) BY processor
```

3. 为每个 “host” 绘制 “CPU” 平均值的图表

为每个 “host” 计算 “CPU” 的平均值。

```
... | timechart span=1m avg(CPU) BY host
```

4. 按 “host” 绘制 “cpu_seconds” 平均值的图表，并删除离群值

按 “host” 计算 “cpu_seconds” 的平均值。删除可能扭曲时间图表轴的无关值。

```
... | timechart avg(cpu_seconds) BY host | outlier action=tf
```

5. 绘制主机不同时段的 “thrput” 平均值的图表

```
... | timechart span=5m avg(thrput) BY host
```

6. 按数据来源 `ip` 绘制事件类型图表

为每一分钟按计数大于 10 的 `source_ip` 对事件类型计数

```
sshd failed OR failure | timechart span=1m count(eventtype) BY source_ip usenull=f WHERE count>10
```

7. 将图表时间数据箱对齐到本地时间

将时间数据箱对齐到上午 5 点（本地时间）。将跨度设为 12 小时。数据箱将表示上午 5 点到下午 5 点，然后是下午 5 点到第二天上午 5 点，依次类推。

```
... | timechart _time span=12h aligntime=@d+5h
```

8. 在多值 BY 字段中，删除重复的值

对于每个唯一的 `mvfield` 值，返回 `field` 的平均值。删除重复的 `mvfield` 中的值。

```
... | timechart avg(field) BY mvfield dedup_splitval=true
```

延伸示例

1. 为不同产品绘制收入图表

此示例将使用搜索教程中的示例数据，并使用字段查找向事件数据中添加更多信息。要亲自尝试此示例：

- 从搜索教程中的本主题下载 tutorialdata.zip 文件，然后将文件按照说明上传到 Splunk 部署。
- 从搜索教程中的本主题下载 Prices.csv.zip 文件，并按照说明设置您的字段查找。
- 当运行搜索时，请使用时间范围昨天。

Tutorialdata.zip 文件包括 productId 字段，这是 Buttercup Games 网店所出售商品的目录号。字段查找使用 prices.csv 文件以向您的事件添加两个新的字段：productName 和 price，前者为商品的描述性名称，后者为商品的费用。

针对昨天售出的不同产品绘制收入图表。

```
sourcetype=access_* action=purchase | timechart per_hour(price) by productName usenull=f useother=f
```

- 本示例搜索所有购买事件（由 action=purchase 定义）。
- 结果传递给 timechart 命令。
- per_hour() 函数为每个 productName 将 price 字段的值相加，并按时间列出总计。

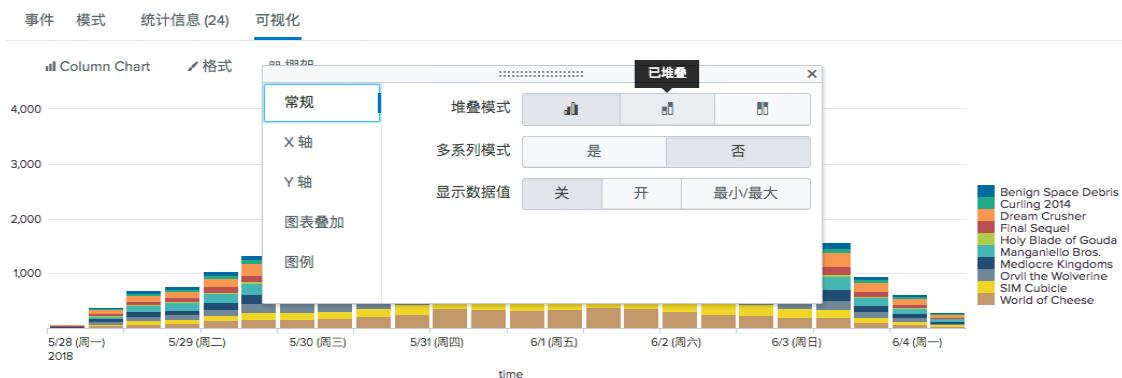
此搜索在统计选项卡中产生以下结果表格。要将数字格式设置为适用于货币的小数位，单击列标题中的格式图标。在数字格式选项卡中，选择精确度。



The screenshot shows a table of purchase data from May 6 to May 14, 2018. A context menu is open over the 'price' column, specifically over the value '0.00' for May 11. The menu is titled '数字格式' (Number Format) and includes options for precision (0, 0.0, 0.00, 0.000, 0.0000), thousands separator (使用千位分隔符), and unit (单位). The '0.00' option is selected.

_time	Benign Space Debris	Curing 2014	Dream Crusher	Final Sequel	Holy Blade of Gouda	Manganelli Bros.	Mediocre Kingdoms	Orville the Wolverine	SIM Cubicle	World of Cheese
2018/05/06	3.12									
2018/05/07	17.70									
2018/05/08	42.69									
2018/05/09	55.19									
2018/05/10	61.43									
2018/05/11	82.26									
2018/05/12	94.75	71.630833	206.615000	135.362500	25.707063	223.277500	148.898750	161.626250	134.099583	156.18
2018/05/13	97.88	64.967500	238.273750	145.775000	23.710417	223.277500	159.311250	154.961250	134.932500	177.01
2018/05/14	99.96	89.955000	291.593750	170.765000	34.442500	254.936250	215.538750	203.282500	155.755417	212.41

单击可视化选项卡。如需要，将图表改为柱状图。在格式菜单中，“一般”选项卡包含“堆叠模式”选项，您可以将图表更改为您希望的堆叠图表。



新建此图表后，可以将鼠标悬停在各个部分，以查看一天中对应小时时段内所售出的产品的更多指标。

请注意，该图表并非以小时跨度显示数据。由于未提供跨度（如 span=1hr），per_hour() 函数会对值进行转换，所以获得的结果为时间范围（本例中为 24 小时）内每小时的总和。

2. 按产品类型绘制每日购买量图表

此示例使用搜索教程中的示例数据。要在您自己的 Splunk 实例中尝试此示例，您必须下载示例数据并遵循操作以将教程数据导入 Splunk。当运行搜索时，请使用时间范围所有时间。

绘制每天每种产品购买数量的图表。

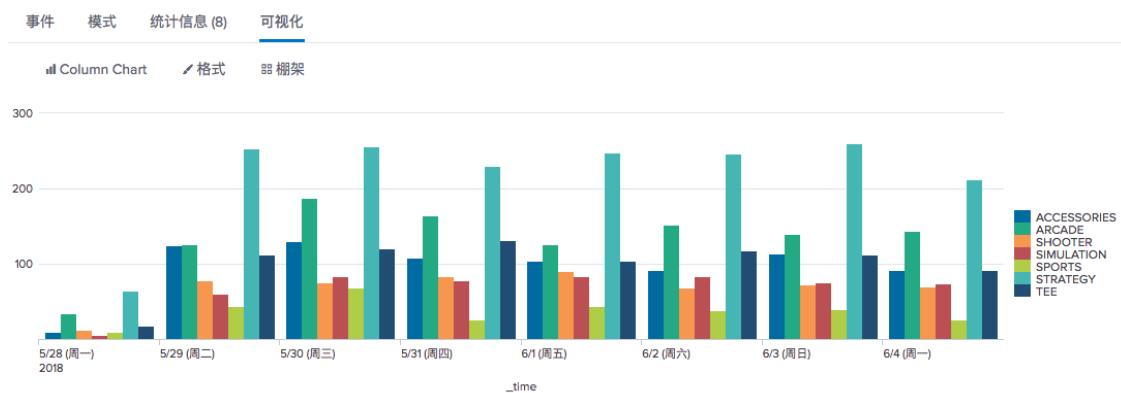
```
sourcetype=access_* action=purchase | timechart span=1d count by categoryId usenull=false
```

- 本示例搜索所有的购物事件，由 `action=purchase` 定义，并通过管道符把结果传递给 `timechart` 命令。
 - `span=1day` 参数用数据桶把一周的购物事件计数存储成以天为单位的数据块。
 - `usenull=f` 参数忽略任何 `categoryId` 为空值的事件。

统计选项卡中显示的结果如下所示：

_time	ACCESSORIES	ARCADE	SHOOTER	SIMULATION	SPORTS	STRATEGY	TEE
2018-03-29	5	17	6	3	5	32	9
2018-03-30	62	63	39	30	22	127	56
2018-03-31	65	94	38	42	34	128	60
2018-04-01	54	82	42	39	13	115	66
2018-04-02	52	63	45	42	22	124	52
2018-04-03	46	76	34	42	19	123	59
2018-04-04	57	70	36	38	20	130	56
2018-04-05	46	72	35	37	13	106	46

单击可视化选项卡。如需要，将图表改为柱状图。



对每天以及一周内售出的各种商品的数量进行对比。

3. 显示 1 周间隔中的结果

此搜索使用从 USGS 地震网站下载的近期地震数据。该数据是一个逗号分隔的 ASCII 文本文件，其中包含每次记录的地震的震级（mag）、坐标（经度、纬度）、区域（地点）等。

您可以从 USGS 地震源下载当前 CSV 文件，然后将文件上传到 Splunk 实例。此示例使用过去 30 天的所有地震数据。

该搜索计算阿拉斯加震级大于或等于 3.5 的地震的数量。结果组织时间跨度为 1 周，从周一开始。

```
source=all_month.csv place=*alaska* mag>=3.5 | timechart span=w@w1 count BY mag
```

- 使用 `<by-clause>` 按震级对地震进行分组。
 - 您只能在 `timechart` 命令中使用具有对齐跨度参数的周跨度。更多信息，请参阅“[指定对齐时间单位](#)”。

统计选项卡中显示的结果如下所示：

<u>_time</u>	3.5	3.6	3.7	3.8	4	4.1	4.1	4.3	4.4	4.5	OTHER
	124	125	126	127	128	129	130	131	132	133	134

2018-03-26	3	3	2	2	3	1	0	2	1	1	1
2018-04-02	5	7	2	0	3	2	1	0	0	1	1
2018-04-09	2	3	1	2	0	2	1	1	0	1	2
2018-04-16	6	5	0	1	2	2	2	0	0	2	1
2018-04-23	2	0	0	0	0	2	1	2	2	0	1

4. 统计一段时间内每个项目的收入

此示例将使用搜索教程中的示例数据，并使用字段查找向事件数据中添加更多信息。运行此示例之前：

- 从搜索教程中的本主题下载数据集，并按照说明将其上传到 Splunk 部署。
- 从搜索教程中的本主题下载 Prices.csv.zip 文件，并按照说明设置您的字段查找。

原始数据集包括 productId 字段，这是 Buttercup Games 网店售出的商品的目录号。字段查找将向事件中添加两个新字段：productName 和 price，前者为商品的描述性名称，后者为商品的费用。

计算在过去 7 天商店售出的每个项目的总计收入。本示例显示生成计算结果的两个不同搜索。

搜索 1

第一个搜索使用 span 参数用数据桶把搜索结果次数存储为以 1 天为基数递增。然后搜索使用 sum() 函数为每个 productName 添加 price。

```
sourcetype=access_* action=purchase | timechart span=1d sum(price) by productName usenull=f
```

搜索 2

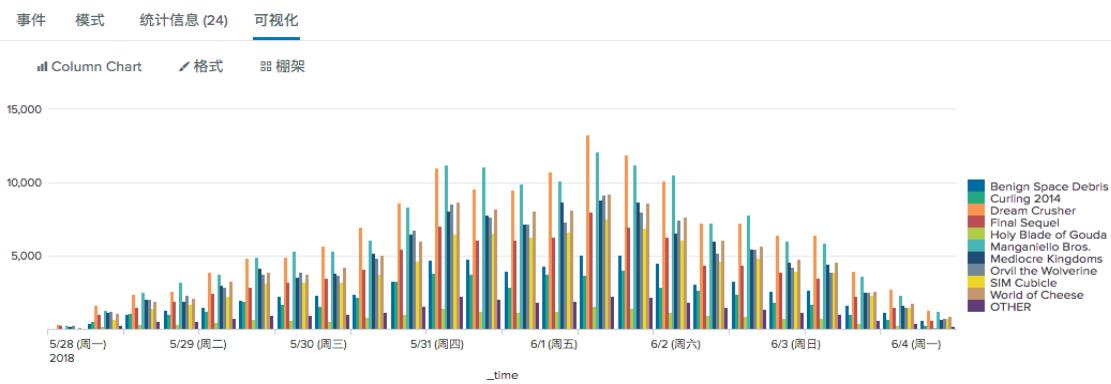
第二个搜索使用 per_day() 函数计算 price 值每天的总和。

```
sourcetype=access_* action=purchase | timechart per_day(price) by productName usenull=f
```

两个搜索将产生相似的结果。搜索 1 产生有两位小数的值。搜索 2 产生有六位小数的值。下图显示搜索 1 的结果。

事件	模式	统计信息 (24)	可视化
每页 20 个	格式	预览	
_time	Benign Space Debris	Curling 2014	Dream Crusher
			Final Sequel
			Holy Blade of Gouda
			Manganiello Bros.
			Mediocre Kingdoms
			Orvil the Wolverine
			SIM Cubicle
			World of Cheese
			OTHER
2018/05/06	74.97	99.95	359.91
	274.89	5.99	279.93
		174.93	279.93
			59.97
			149.94
			40.91
2018/05/07	424.83	539.73	1639.59
	1049.58	191.68	1279.68
		1174.53	1239.69
			659.67
			1124.55
			284.36
2018/05/08	1,024.59	1119.44	2399.40
	1524.39	365.39	2559.36
		2874.17	2879.48
			1439.28
			1924.23
			517.83
2018/05/09	1,324.47	1039.48	2599.35
	1899.24	311.48	3239.19
		1899.24	2359.41
			1679.16
			2149.14
			576.70
2018/05/10	1,474.41	1239.38	3919.02
	2499.00	467.22	3759.06
		3023.79	2879.28
			2278.86
			3298.68
			773.26
2018/05/11	1,974.21	1939.03	4838.79
	2873.85	676.87	4918.77
		4198.32	3759.06
			3118.44
			3898.44
			935.90
2018/05/12	2,274.09	1719.14	4958.76
	3248.70	616.97	5358.66
		3573.57	3879.03
			3218.39
			3748.50
			968.82
2018/05/13	2,349.06	1559.22	5718.57
	3498.60	569.05	5358.66
		3823.47	3719.07
			3238.38
			4248.30
			1008.73
2018/05/14	2,399.04	2158.92	6998.25
	4098.36	826.62	6118.47
		5172.93	4878.78
			3738.13
			5097.96
			1182.35

单击可视化选项卡。如需要，将图表改为柱状图。



现在，您可以对每天以及一周内售出的商品的总收入进行对比。

5. 绘制一天的产品浏览和购买图表

此示例使用搜索教程中的示例数据，但应该可以与任意格式的 Apache Web 访问日志配合使用。要在您自己的 Splunk 实例中尝试此示例，您必须下载示例数据并遵循操作以将教程数据导入 Splunk。当运行搜索时，请使用时间范围昨天。

绘制单日视图的图表，并从 Buttercup Games 线上商店购买。

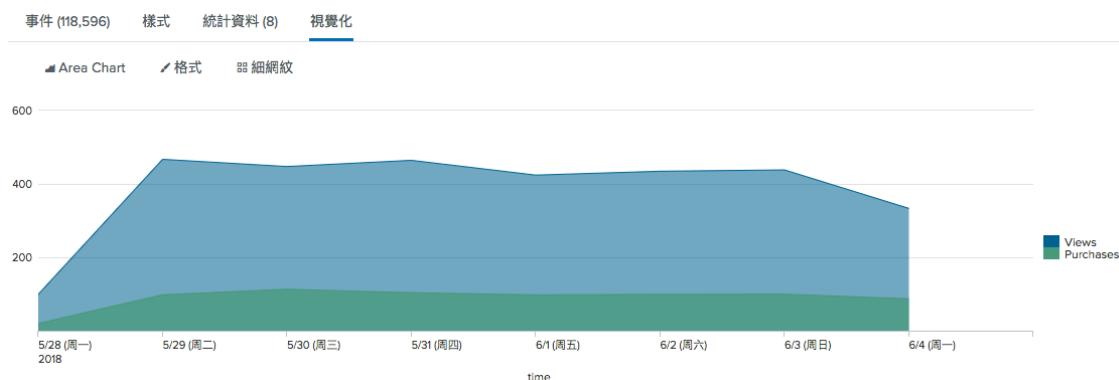
```
sourcetype=access_* | timechart per_hour(eval(method="GET")) AS Views, per_hour(eval(action="purchase")) AS Purchases
```

- 此搜索使用 `per_hour()` 函数和 `eval` 表达式搜索页面浏览和 (`method=GET`) 购物活动 (`action=purchase`)。
- `eval` 表达式的结果将分别重命名为 `Views` 和 `Purchases`。

统计选项卡中显示的结果如下所示：

_time	视图	购买
2018-04-05 00:00:00	150.000000	44.000000
2018-04-05 00:30:00	166.000000	54.000000
2018-04-05 01:00:00	214.000000	72.000000
2018-04-05 01:30:00	242.000000	80.000000
2018-04-05 02:00:00	158.000000	26.000000
2018-04-05 02:30:00	166.000000	20.000000
2018-04-05 03:00:00	220.000000	56.000000

单击可视化选项卡。将结果格式设置为面积图。



两个面积之差即表示许多未达成交易的查看。如果所有查看都变成购买，您将会看到区域完全重叠。两个面积之间没有差别。

where 子句示例

这些示例使用 where 子句控制时间系列图中返回的系列值的数量。

示例 1：基于最小计数值显示 5 个最罕见的系列。所有其他系列值将以 "other" 来标记。

```
index=_internal | timechart span=1h count by source WHERE min in bottom5
```

示例 2：基于最大值显示 5 个最常见的系列。所有其他系列值将以 "other" 来标记。

```
index=_internal | timechart span=1h count by source WHERE max in top5
```

这两个搜索将返回六个数据系列：五个指定的 top 或 bottom 系列，以及标记为 other 的系列。要隐藏 "other" 系列，请指定参数 useother=f。

示例 3：显示 INFO 事件的数据来源系列计数，但只在事件总数大于 100 的时候显示。所有其他系列值将以 "other" 来标记。

```
index=_internal | timechart span=1h sum(eval(if(log_level=="INFO",1,0))) by source WHERE sum > 100
```

示例 4：使用具有 count 函数的 where 子句测量此期间的事件总数。生成的结果与使用 sum 函数的结果类似。

下面的两个搜索返回事件总数大于 100 的来源系列。所有其他系列值将以 "other" 来标记。

```
index=_internal | timechart span=1h count by source WHERE count > 100
```

```
index=_internal | timechart span=1h count by source WHERE sum > 100
```

另请参阅

命令

```
bin  
chart  
sitimechart
```

博客

搜索命令优于 stats、chart 和 timechart

问答

有什么问题吗？请访问 Splunk Answers，查看 Splunk 社区有哪些与使用 timechart 命令相关的问题和解答。

timewrap

描述

显示或将 timechart 命令的输出换行，这样每个时间段都是不同系列。

您可以使用 timewrap 命令比较特定时间段（如每天或每月）的数据。您还可以使用 timewrap 命令比较多个时间段，如某两周和另外两周相比。请参阅“Timescale 选项”。

语法

要求的语法以粗体表示。

```
timewrap  
<timewrap-span>  
[align=now | end]  
[series=relative | exact | short]  
[time_format=<str>]
```

必要参数

timewrap-span

语法: [<int>]<timescale>

描述: 每个数据箱的跨度（基于时间）。timescale 为必填项。int 并非必填项。如果未指定 <int>, 则假定为 1。例如, 如果 timescale 指定为 day, 则假定为 1day。请参阅“Timescale 选项”。

可选参数

align

语法: align=now | end

描述: 指定换行是否应与当前时间或搜索结束时间对齐。

默认: end

series

语法: series=relative | exact | short

描述: 指定数据系列命名方式。如果 series=relative 和 timewrap-span 设为周, 则字段名称为 latest_week、1week_before、2weeks_before 等等。如为 series=exact, 使用 time_format 参数为系列名称指定自定义格式。如为 series=short, 则字段名称为与 series=relative 一起使用的字段名称的缩写。如果使用 series=short, 则字段名称会缩写为 "s", 后跟一个代表时间段的数字。例如, 如果 timewrap-span 设置为 week, 则字段名称为 s0、s1、s2 等等。字段 s0 代表最近的一周。字段 s1 表示最近一周的前 1 周。

默认: relative

time_format

语法: time_format=<str>

描述: 与 series=exact 结合使用, 以为系列指定自定义名称。time_format 设计用于和时间格式变量结合使用。例如, 如果您指定 time_format="week of %d/%m/%y", 则格式显示为 week of 13/2/17 和 week of 20/2/17。如果您指定 time_format=week of %b %d, 则格式显示为 week of Feb 13 和 week of Feb 20。请参阅“用法”一节。

默认值: 无

Timescale 选项

<timescale>

语法: <sec> | <min> | <hr> | <day> | <week> | <month> | <quarter> | <year>

描述: 时间刻度单位。

时间刻度	语法	描述
<sec>	s sec secs second seconds	时间刻度(秒)。
<min>	min mins minute minutes	时间刻度(分钟)。
<hr>	h hr hrs hour hours	时间刻度(小时)。
<day>	d day days	时间刻度(天)。
<week>	w week weeks	时间刻度(周)。
<month>	m mon month months	时间刻度(月)。
<quarter>	qtr quarter quarters	时间刻度(季度)
<year>	y yr year years	时间刻度(年)。

timewrap 命令使用缩写 m 指代月份数。其他命令, 如 timechart 和 bin 使用缩写 m 指代分钟数。

用法

timewrap 命令属于报表命令。

在使用 timewrap 命令之前, 必须在搜索中使用 timechart。

换行是基于搜索的结束时间。如果您指定 All time 的时间范围, 换行则基于今天的日期。您可以在数据系列名称中及 _time 字段的 timestamp 中看到这一点。

带时间图表 BY 子句的字段名称

如果在搜索的 timechart 命令部分使用 BY 子句, 则 timewrap 命令生成的字段名称会附加到 BY 子句生成的字段名称的后面。例如, 假设您有一个搜索, 其中的 timechart 命令包含了 BY categoryId, 那么结果会类似这样:

_time	ACCESSORIES	SPORTS	STRATEGY
2020/5/21	5	17	32
2020/5/22	62	22	127
2020/5/23	65	34	128
2020/5/24	5	17	32
2020/5/25	62	22	127
2020/5/26	65	34	128

在添加 `timewrap` 命令时，例如 `| timewrap w series=short`，系列字段名称会附加到 `timechart BY` 子句的类别 ID 名称后面。

输出形式如下所示：

_time	ACCESSORIES_s1	SPORTS_s1	STRATEGY_s1	ACCESSORIES_s0	SPORTS_s0	STRATEGY_s0
2020/5/21				5	17	32
2020/5/22				62	22	127
2020/5/23				65	34	128
2020/5/24				5	17	32
2020/5/25	62	22	127	17	54	39
2020/5/26	65	34	128			

使用 `time_format` 参数

如果您未将 `time_format` 参数搭配任何时间指标符，则所有数据系列都会显示相同的名称，并相互压缩。

示例

1. 按周比较

在按周比较表中显示一份时间表，其中每个计数时间跨度为 1 天。每个表列是一个系列，为 1 周时间。

```
... | timechart count span=1d | timewrap 1week
```

2. 比较今天、昨天的结果和周平均值

要将几天的结果与周平均值进行比较，您需要计算每日总计、计算每周平均值，并删除您不想使用的天数。例如：

```
... | timechart count span=1h | timewrap d series=short | addtotals s* | eval 7dayavg=Total/7.0 | table _time, _span, s0, s1, 7dayavg | rename s0 as now, s1 as yesterday
```

- 使用 `timewrap` 命令生成过去 7 天的结果。
- 如果使用 `series=short` 参数，输出中生成的字段名称会以 "s" 开头，从而让您可以轻松地通过 `addtotals` 命令创建总计。
- 使用 `addtotals` 和 `eval` 命令来计算这 7 天的平均值。
- `table` 命令用于删除第 3 至第 7 天的结果，以便只返回今天、昨天的结果和周平均值。
- `rename` 命令用于重命名字段。

输出形式如下所示：

_time	now	昨天	7dayavg
2020/2/20 15:00	0	0	0.0
2020/2/20 16:00	0	0	0.29

2020/2/20 17:00	0	0	0. 0
2020/2/20 18:00	0	0	0. 0
2020/2/20 19:00	0	0	0. 57
2020/2/20 20:00	0	0	0. 0
2020/2/20 21:00	0	0	0. 29
2020/2/20 22:00	0	0	1. 1

3. 比较不同周中的同一天

您可以在搜索结束时指定一个过滤器，从而对一周中的某一天与其他周中的同一天进行比较。例如，要比较每周的 Wednesday（星期三），搜索形式如下所示：

```
... | timechart count span=1h | timewrap w | where strftime(_time, "%A") == "Wednesday"
```

输出形式如下所示：

_time	4weeks_before	3weeks_before	2weeks_before	1week_before	latest_week
2020/2/19 0:00	0	1	4	0	1
2020/2/19 1:00	2	0	0	0	1
2020/2/19 2:00	3	5	7	2	0
2020/2/19 3:00	6	4	0	1	2
2020/2/19 4:00	9	0	4	0	0
2020/2/19 5:00	2	8	7	3	1
2020/2/19 6:00	4	2	7	0	1
2020/2/19 7:00	6	9	2	2	0

如果将时间图表跨度更改为 1d 而不是 1h，则输出形式如下所示：

_time	4weeks_before	3weeks_before	2weeks_before	1week_before	latest_week
2020/2/19	32	29	31	8	6

另请参阅

`timechart`

`tojson`

描述

将事件转换为 JSON 对象。可以通过精确匹配或通配符表达式识别字段，从而指定要转换的字段。您还可以使用数据类型函数将特定的 JSON 数据类型应用于字段值。`tojson` 命令可将多值字段转换为 JSON 数组。

当在 `tojson` 搜索中明确命名字段时，该命令生成的 JSON 对象仅限于那些命名字段的值。如果没有为 `tojson` 指定字段，则 `tojson` 将为所有字段生成 JSON 对象，否则搜索将返回这些字段。

语法

要求的语法以**粗体**表示。

```
| tojson
[<tojson-function>]...
```

```
[default_type=<datatype>]  
[fill_null=<boolean>]  
[include_internal=<boolean>]  
[output_field=<string>]
```

可选参数

tojson-function

语法: [auto | bool | json | none | num | str](<wc-field>)...

描述: 将 JSON 数据类型函数应用于命名字段的值。有关 tojson 如何解释这些数据类型函数, 以及 tojson 在将事件转换为 JSON 对象时如何将数据类型应用于字段值的详细信息, 请参阅“用法”。

如果不提供任何字段, tojson 处理器将为包含所有可用字段的每个事件创建 JSON 对象。换句话说, 它将对搜索应用 none(*)。

默认值: none (*)

default_type

语法: default_type=<datatype>

描述: 指定 tojson 处理器应该应用于与数据类型函数没有特别关联的字段的数据类型。

默认值: none

fill_null

语法: fill_null=<boolean>

描述: 如果设置为 true, 则在 tojson 跳过一个值时, tojson 会输出一个文本 null 值。例如, 通常情况下, 当 tojson 尝试将 json 数据类型应用于没有正确设定 JSON 格式的字段时, tojson 会跳过该字段。但是, 如果 fill_null=true, tojson 处理器将输出一个 null 值

默认值: false

include_internal

语法: include_internal=<boolean>

描述: 当设置为 true 时, tojson 在其 JSON 对象输出中包含 `splexicon:internalfield:internal` 字段, 例如 `_time`、`_indexetime` 或 `_raw`。

默认值: false

output_field

语法: output_field=<string>

描述: 指定 tojson 搜索处理器将输出 JSON 对象写入的字段的名称。

默认值: `_raw`

用法

tojson 命令是一个流命令, 这意味着它作用于搜索返回的每个事件。请参阅“命令类型”。

将 JSON 数据类型应用于字段值

tojson 命令根据其数据类型函数中编码的逻辑将 JSON 数据类型应用于字段值。

您可以在编写 tojson 搜索时为字段分配特定的数据类型函数。或者, 您可以命名一组字段而不将它们与数据类型函数相关联, 然后确定 tojson 可以应用于这些非关联字段的 default_type。

如果您没有为 tojson 命令指定任何字段, 则 tojson 将为此时搜索可能返回的每个字段返回 JSON 对象, 并将 none 数据类型函数应用于这些字段的值。none 数据类型函数将数字数据类型应用于纯数字的字段值, 并将字符串数据类型应用于所有其他字段值。

下表解释了各种数据类型函数用于将数据类型应用于与其关联的字段值的逻辑。

数据类型函数	转换逻辑
auto	<p>将指定字段的所有值转换为 JSON 格式的输出。自动确定字段数据类型。</p> <ul style="list-style-type: none">如果值为数字, 则 JSON 输出包含数字输出, 并且包含文本数值。如果值为字符串 true 或 false, 则 JSON 输出为布尔类型。如果值是文字 null, 则 JSON 输出为 null 类型并包含 null 值。如果该值是前面提到的字符串以外的字符串, 则 tojson 将检查该字符串。如果是正确的 JSON, tojson 将输出一个嵌套的 JSON 对象。如果它不是正确的 JSON, tojson 会在输出中包含字符串。
	<p>使用字符串验证将指定字段的有效值转换为布尔数据类型, 并跳过无效值。</p> <ul style="list-style-type: none">如果值为数字, 则 tojson 仅在该值为 0 时才输出 false。否则, tojson 输出 false。

bool	<ul style="list-style-type: none">如果值为字符串，则 <code>tojson</code> 仅在该值为 <code>false</code>、<code>f</code> 或 <code>no</code> 时才输出 <code>false</code>。<code>tojson</code> 处理器仅在该值为代码 <code>true</code>、<code>t</code> 或 <code>yes</code> 时才输出 <code>true</code>。如果该值不适合这两组字符串，则会跳过它。<code>bool</code> 数据类型函数的验证不区分大小写。这意味着它也将 <code>FALSE</code>、<code>False</code>、<code>F</code> 和 <code>NO</code> 解释为 <code>false</code>。
json	<p>使用字符串验证将指定字段的所有值转换为 JSON 类型。跳过具有无效 JSON 的值。</p> <ul style="list-style-type: none">如果值为数字，则 <code>tojson</code> 输出该数字。如果值为字符串，则 <code>tojson</code> 将字符串作为 JSON 块输出。如果值是无效的 JSON，则 <code>tojson</code> 会跳过它。
无	<p>输出 JSON 类型中指定字段的所有值。不应用字符串验证。</p> <ul style="list-style-type: none">如果值为数字，则 <code>tojson</code> 在 JSON 块中输出数字数据类型。如果值为字符串，则 <code>tojson</code> 输出字符串数据类型。
num	<p>使用字符串验证将指定字段的所有值转换为数字类型。</p> <ul style="list-style-type: none">如果值为数字，则 <code>tojson</code> 输出该值并为其指定数字数据类型。如果值为字符串，则 <code>tojson</code> 尝试将字符串解析为数字。如果不能，则跳过该值。
str	<p>使用字符串验证将指定字段的所有值转换为字符串数据类型。 <code>tojson</code> 处理器将字符串类型应用于指定字段的所有值，即使它们是数字、布尔值等也如此。</p>

当字段包含多值时，`tojson` 输出一个 JSON 数组，并将数据类型函数逻辑应用于数组的每个元素。

示例

1. 将搜索返回的所有事件转换为 JSON 对象

对 `index= internal` 的这种搜索会将其时间范围内返回的所有事件转换为 JSON 格式的数据。由于搜索字符串不会将数据类型函数分配给特定字段，默认情况下 `tojson` 将 `none` 数据类型函数应用于搜索返回的所有字段。这意味着它们的所有值都是数字或字符串数据类型。

index= internal | tojson

例如，假设您从如下所示的事件开始：

12-18-2020 18:19:25.601 +0000 INFO Metrics - group=thruput, name=thruput, instantaneous_kbps=5.821, instantaneous_eps=27.194, average_kbps=5.652, total_k_processed=444500.000, kb=180.443, ev=843, load_average=19.780

经过 `tojson` 处理后，此类事件的 JSON 格式如下：

```
{ [-]
  component: Metrics
  date_hour: 18
  date_mday: 18
  date_minute: 22
  date_month: december
  date_second: 9
  date_wday: friday
  date_year: 2020
  date_zone: 0
  event_message: group=thruput, name=thruput, instantaneous_kbps=2.914, instantaneous_eps=13.903, average_kbps=5.062,
total_k_processed=398412.000, kb=90.338, ev=431, load_average=14.690
  group: thruput
  host: sh1
  index: _internal
  linecount: 1
  log_level: INFO
  name: thruput
  punct: --:::+__-=,_=,=_.,=_.,=_.,=_.,=_.
  source: /opt/splunk/var/log/splunk/metrics.log
  sourcetype: splunkd
  splunk_server: idx2
  timeendpos: 29
  timestampstartpos: 0
```

```
}
```

2. 为“日期”字段指定不同的数据类型

internal 索引的以下搜索将结果转换为 JSON 对象，这些对象仅包含来自每个事件的 date* 字段。数字数据类型应用于所有 date_hour 字段值。字符串数据类型应用于所有其他日期字段值。

```
index=_internal | toJSON num(date_hour) str(date_*)
```

此搜索产生的 JSON 对象如下所示：

```
{ [-]
  date_hour: 18
  date_mday: 18
  date_minute: 28
  date_month: december
  date_second: 45
  date_wday: friday
  date_year: 2020
  date_zone: 0
}
```

请注意，所有不以 date_ 开头的字段都已从输出中剔除。

3. 限制 JSON 对象输出并将数据类型应用于字段值

此搜索仅返回 name、age 和 isRegistered 字段的 JSON 对象。它使用 auto 数据类型函数让 toJSON 自动将适当的 JSON 数据类型应用于这些字段的值。

```
... | toJSON auto(name) auto(age) auto(isRegistered)
```

4. 将所有事件转换为 JSON 对象并将适当的数据类型应用于所有字段值

此搜索将搜索返回的每个事件中的所有字段转换为 JSON 对象。它将 auto 数据类型函数与通配符结合使用，将适当的数据类型应用于搜索返回的所有字段的值。

```
... | toJSON auto(*)
```

请注意，此搜索引用了 auto 数据类型函数，该函数确保布尔值、JSON 和 null 字段值与数字和字符串值一起正确键入。

或者，您可以使用 default_type 将 auto 数据类型函数应用于搜索返回的所有字段：

```
... | toJSON default_type=auto
```

5. 将布尔数据类型应用于特定字段

此示例生成包含 isInternal 字段值的 JSON 对象。它使用 bool 数据类型函数将布尔数据类型应用于这些字段值。

```
... | toJSON bool(isInternal)
```

6. 包括内部字段并为跳过的字段分配一个“null”值

此示例演示了 include_internal 和 fill_null 参数的用法。

```
... | toJSON include_internal=true fill_null=true
```

7. 为一组字段指定默认数据类型并将 JSON 对象写入另一个字段

此搜索基于四个字段的值生成 JSON 对象。它使用 default_type 参数将前三个字段转换为 num 数据类型。它将字符串数据类型应用于第四个字段。最后，它将完成的 JSON 对象写入 my_JSON_field 字段。

```
... | toJSON age height weight str(name) default_type=num output_field=my_JSON_field
```

top

描述

查找字段列表中字段的最常见值。计算事件中出现的值的计数和频率百分比。如果包括 <by-clause>, 按照您在 <by-clause> 中指定的字段对结果进行分组。

语法

top [<N> [<top-options>...]]<field-list> [<by-clause>]

必要参数

<field-list>

语法: <field>, <field>, ...

描述: 以逗号分隔的字段名称的列表。

可选参数

<N>

语法: <int>

描述: 要返回的结果数。

默认值: 10

<top-options>

语法: countfield=<string> | limit=<int> | otherstr=<string> | percentfield=<string> | showcount=<bool> | showperc=<bool> | useother=<bool>

描述: top 命令的选项。请参阅 Top 选项。

<by-clause>

语法: BY <field-list>

描述: 分组所依据的一个或多个字段的名称。

Top 选项

countfield

语法: countfield=<string>

描述: 对于 top 命令返回的各值, 结果还会返回具有该值的事件计数。此参数指定包含计数的字段的名称。默认返回计数。如果您不想要返回事件计数, 指定 showcount=false。

默认值: count

limit

语法: limit=<int>

描述: 指定要返回的结果数量。要返回所有值, 指定零 (0)。指定 top limit=<int> 和指定 top N 是相同的。

默认值: 10

otherstr

语法: otherstr=<string>

描述: 如果 useother=true, 将代表所有其他值的行添加到结果中。使用 otherstr=<string> 指定行标签的名称。

默认值: OTHER

percentfield

语法: percentfield=<string>

描述: 对于 top 命令返回的各值, 结果还会返回具有该值的事件百分比。此参数指定包含百分比的字段的名称。默认返回百分比。如果您不想要返回事件百分比, 指定 showperc=false。

默认值: 百分比

showcount

语法: showcount=<bool>

描述: 指定是否使用该元组的计数新建名为 "count" 的字段 (请参阅 "countfield" 选项)。

默认值: true

showperc

语法: showperc=<bool>

描述: 指定是否使用该元组的相对流行度新建名为 "percent" 的字段 (请参阅 "percentfield" 选项)。

默认值: true

useother

语法: useother=<bool>

描述: 指定是否添加一行来表示由于限制截止点的原因而未包括的所有值。

默认值: false

用法

`top` 命令是一个转换命令。请参阅“命令类型”。

默认字段

当您使用 `top` 命令时，会将两个字段添加到结果中： `count` 和 `percent`。

字段	描述
count	搜索结果中包含 <code>top</code> 命令返回的字段值的事件数。请参阅 <code>countfield</code> 和 <code>showcount</code> 参数。
percent	搜索结果中包含 <code>top</code> 命令返回的字段值的事件百分比。请参阅 <code>percentfield</code> 和 <code>showperc</code> 参数。

默认的最大结果数

默认情况下，`top` 命令最多返回 50,000 个结果。此最大值受 `limits.conf` 文件中 `[top]` 段落的 `maxresultrows` 设置控制。提高此限制会导致占用更多内存。

只有有着文件系统访问权限的用户，如系统管理员，才能编辑配置文件。不要更改或复制默认目录中的配置文件。默认目录中的文件必须保持原样并位于其原始位置。在本地目录进行更改。

请参阅“如何编辑配置文件”。

如果您正在使用 Splunk Cloud，想要编辑配置文件，请向 Splunk 支持提交工单。

示例

示例 1：返回字段的 20 个最常用值

该搜索返回 “referer” 字段的 20 个最常用值。结果显示具有 `referer` 计数的事件数（计数），以及每个 `referer` 占事件总数的百分比。

```
sourcetype=access_* | top limit=20 referer
```

referers	count	percent
http://www.buttercupgames.com/category.screen?categoryId=STRATEGY	2284	4.389606
http://www.buttercupgames.com	1980	3.805351
http://www.google.com	1582	3.040437
http://www.buttercupgames.com/category.screen?categoryId=ARCADE	1372	2.636839
http://www.buttercupgames.com/category.screen?categoryId=NULL	1281	2.461946
http://www.buttercupgames.com/product.screen?productId=SF-BVS-G01	1210	2.325492
http://www.buttercupgames.com/category.screen?categoryId=ACCESSORIES	1082	2.079490
http://www.buttercupgames.com/category.screen?categoryId=TEE	993	1.908441
http://www.yahoo.com	766	1.472171
http://www.buttercupgames.com/product.screen?productId=WC-SH-G04	725	1.393373
http://www.buttercupgames.com/category.screen?categoryId=SIMULATION	708	1.360701
http://www.buttercupgames.com/category.screen?categoryId=SHOOTER	703	1.351092
http://www.buttercupgames.com/product.screen?productId=DB-SG-G01	696	1.337638

示例 2：返回由另一个字段组织的一个字段的最高值

此搜索返回每个 “`referer_domain`” 的最高 “`action`” 值。

```
sourcetype=access_* | top action by referer_domain
```

由于未指定限制，因此会返回 “`action`” 和 “`referer_domain`” 所有值组合，以及计数和百分比：

每页 20 个 ▾ 格式 ▾ 预览 ▾

referer_domain	action	count	percent
http://mystore.splunk.com	purchase	51525	95.310766
http://mystore.splunk.com	update	2535	4.689234
http://www.bing.com	view	46	51.111111
http://www.bing.com	addtocart	21	23.333333
http://www.bing.com	remove	12	13.333333
http://www.bing.com	changequantity	11	12.222222
http://www.buttercupgames.com	purchase	5737	30.120229
http://www.buttercupgames.com	addtocart	5572	29.253951
http://www.buttercupgames.com	view	5054	26.534362
http://www.buttercupgames.com	remove	1359	7.134982
http://www.buttercupgames.com	changequantity	1325	6.956476
http://www.google.com	view	201	50.375940
http://www.google.com	addtocart	96	24.060150
http://www.google.com	remove	53	13.283208
http://www.google.com	changequantity	49	12.280702
http://www.yahoo.com	view	90	49.450549

示例 3：返回针对每个类别购买的热卖产品

此示例将使用搜索教程中的示例数据，并使用字段查找向事件数据中添加更多信息。

- 从添加数据教程下载数据集，并按照说明加载教程数据。
- 从使用字段查找教程下载 CSV 文件，并遵照说明设置查找定义，以添加价格和 productName 到事件。

字段查找配置完毕后，可以使用时间范围所有时间运行此搜索。

搜索返回针对每个类别购买的热卖产品。不显示百分比字段。将计数字段重命名为 "total"。

sourcetype=access_* status=200 action=purchase | top 1 productName by categoryId showperc=f countfield=total

每页 20 个 ▾ 格式 ▾ 预览 ▾

categoryId	productName	total
ACCESSORIES	Fire Resistance Suit of Provolone	1496
ARCADE	Manganiello Bros.	1672
SHOOTER	World of Cheese	1960
SIMULATION	SIM Cubicle	1968
SPORTS	Curling 2014	1104
STRATEGY	Mediocre Kingdoms	1904

另请参阅

rare, sitop, stats

transaction

描述

交易命令根据满足各种约束的事件查找交易。交易的构成内容包括：每个成员的原始文本（_raw 字段）、最早期成员的时间和日期字段，以及每个成员所有其他字段的并集。

此外，transaction 命令还把两个字段添加到原始事件、duration 和 eventcount。duration 字段中的值为交易中第一个事件与最后一个事件的时间戳之差。eventcount 字段中的值为交易中事件的数量。

请阅读《搜索手册》中的“关于交易”。

语法

要求的语法以粗体表示。

```
transaction
[<field-list>]
[name=<transaction-name>]
[<txn_definition-options>...]
[<memcontrol-options>...]
[<rendering-options>...]
```

必要参数

无。

可选参数

field-list

语法: <field> ...

描述: 一个或多个字段名称。事件将根据字段的唯一值分组为交易。例如，假设指定了两个字段: client_ip 和 host。对于每个 client_ip 值，系统会为该 client_ip 的每个唯一的 host 值返回一个单独的交易。

memcontrol-options

语法: <maxopentxn> | <maxopenevents> | <keepevicted>

描述: 这些选项为您的交易控制内存使用量。这不是必需的，但您可使用 0 或更多选项定义您的交易。请参阅“内存控制选项”。

name

语法: name=<transaction-name>

描述: 指定在 transactiontypes.conf 文件中配置的交易的段落名称。这将使用配置文件中此段落定义的设置来运行此搜索。如果为此搜索提供了其他的交易定义选项（比如 maxspan），则这些选项会覆盖配置文件中的设置。

rendering-options

语法: <delim> | <mvlist> | <mvraw> | <nullstr>

描述: 这些选项为您的交易控制多值呈现。这不是必需的，但您可使用 0 或更多选项定义您的交易。请参阅“多值呈现选项”。

txn_definition-options

语法: <maxspan> | <maxpause> | <maxevents> | <startswith> | <endswith> | <connected> | <unifyends> | <keeporphans>

描述: 指定交易定义选项，以定义您的交易。可以使用多个选项来定义您的交易。

交易定义选项

connected

语法: connected=<bool>

描述: 仅在指定了 <field-list> 时才有相关性。如果某个事件包含交易所需的字段，但这些字段都没有在此交易中实例化（通过前一个事件添加），则这将打开新交易（connected=true）或添加事件到交易（connected=false）。对于多值字段，如果事件中多值字段的值至少有一个是相同的，则指定 connected=false 将事件合并到一个交易中。请参阅[用法](#)。

默认值: true

endswith

语法: endswith=<filter-string>

描述: 搜索或 Eval 表达式，若某事件满足条件，则标志着交易结束。

keeporphans

语法: keeporphans=true | false

描述: 指定交易命令是否应该输出不属于任何交易的结果。当做 "orphans" 传递的结果有别于含 _txn_orphan 字段的交易事件，后者以值 1 代表孤立结果。

默认值: false

maxspan

语法: maxspan=<int>[s | m | h | d]

描述: 指定事件可以延续的最大时间长度，以秒、分、小时或天为单位。交易中各事件的延续时间不得超过为 maxspan 指定的整数。将超过 maxspan 限制的事件处理为独立交易中的一部分。若该值为负数，将禁用 maxspan 约束，事件延续时间将不再受限。

默认值: -1 (无限制)

maxpause

语法: maxpause=<int>[s | m | h | d]

描述: 指定以秒、分、小时或天数为单位的交易中事件暂停的最大长度时间。若值为负数，将禁用 maxpause 约束，事件延续时间将不再受限。

默认值: -1 (无限制)

maxevents
语法: maxevents=<int>
描述: 一个交易中的最大事件数。如果值为负数，此约束将被禁用。
默认值: 1000

startswith
语法: startswith=<filter-string>
描述: 搜索或 eval 筛选表达式，如果某事件满足条件，则标志新交易的开始。

unifyends
语法: unifyends= true | false
描述: 对于匹配 startswith 和 endswith 约束的事件，是否强制这些事件另外匹配至少一个用于将事件统一为一个交易的字段。
默认值: false

筛选字符串选项

这些选项与 startswith 和 endswith 参数一起使用。

<filter-string>
语法: <search-expression> | (<quoted-search-expression>) | eval(<eval-expression>)
描述: 搜索或 Eval 筛选表达式，如果某事件满足条件，则标志交易的结束。

<search-expression>
描述: 一个不包含引号的有效搜索表达式。

<quoted-search-expression>
描述: 一个包含引号的有效搜索表达式。

<eval-expression>
描述: 一个求值结果为布尔值的有效 Eval 表达式。

内存控制选项

如果您有 Splunk Cloud，则 Splunk 支持将代表您在 limits.conf 文件中管理设置。

keepevicted
语法: keepevicted=<bool>
描述: 是否输出已退出的交易。可以通过检查 'closed_txn' 字段的值来区分已退出的交易与未退出的交易。已退出的交易的 'closed_txn' 字段设为 '0' 或 false，未退出或关闭的交易的 'closed_txn' 设为 '1' 或 true。如果满足以下一个条件，'closed_txn' 字段设为 '1': maxevents、maxpause、maxspan、startswith。对于 startswith，因为 transaction 命令以相反的时间顺序查看事件，所以当它满足开始条件后会关闭交易。如果未指定上述任何条件，则会输出所有交易，尽管所有交易都会将 'closed_txn' 设置为 '0'。在达到内存限制时，交易也会退出。
默认值: false 或 0

maxopenevents
语法: maxopenevents=<int>
描述: 使用 LRU 策略指定在交易开始退出之前属于开放交易一部分的最大事件数。
默认值: 该参数的默认值读取自 limits.conf 文件中的交易段落。

maxopentxn
语法: maxopentxn=<int>
描述: 使用 LRU 策略指定在开始退出交易之前保留在开放池中的尚未关闭交易的最大数量。
默认值: 该参数的默认值读取自 limits.conf 文件中的交易段落。

多值呈现选项

delim
语法: delim=<string>
描述: 指定分隔多个值的字符。当与 mvraw=t 参数结合使用时，代表用于分隔 _raw 字段的值的字符串。
默认值: " " (空格)

mvlist
语法: mvlist= true | false | <field-list>
描述: 控制多值字段处理方式的标记。设置为 mvlist=true 时，交易中的多值字段是以到达顺序排序的原始事件列表。设置为 mvlist=false 时，交易中的多值字段是按字母顺序排列的一组唯一字段值。如果提供了以逗号或空格分隔的字段列表，则只有这些字段以列表形式呈现。
默认值: false

`mvraw`

语法: `mvraw=<bool>`

描述: 用于指定交易搜索结果的 `_raw` 字段是否应为多值字段。

默认值: `false`

`nullstr`

语法: `nullstr=<string>`

描述: 将缺失的字段值呈现为交易中多值字段的一部分时使用的字符串值。此选项仅适用于以列表形式呈现的字段。

默认值: `NULL`

用法

`transaction` 命令属于中央流命令。请参阅“[命令类型](#)”。

在输出中，交易中的事件在 `Events` 字段中被分组为多个值。默认情况下，交易中的每个事件都会另起新的一行。

如果交易中事件超过 5 个，则交易中的剩余事件将折叠。交易结尾显示一条供您选择是否显示交易中所有事件的消息。

指定多值字段

Splunk 软件不一定会把多个字段定义的交易解释为这些字段的结合 (`field1 AND field2 AND field3`) 或分离 (`field1 OR field2 OR field3`)。若字段列表中各字段间存在传递关系，且相关事件以正确的顺序显示，每个事件都有不同的时间戳，则 `transaction` 命令将尝试使用该传递关系。例如，若您搜索

```
... | transaction host cookie
```

您可能会看到以下事件会被分组到一个交易中：

```
event=1 host=a  
event=2 host=a cookie=b  
event=3 cookie=b
```

需要按时间降序排列

`transaction` 命令要求传入事件按时间降序排列。某些命令（如 `eval`）可能会更改事件的顺序或时间标签。如果这些命令其中之一在 `transaction` 命令之前，您的搜索将返回错误，除非在搜索中包括 `sort` 命令。`sort` 命令必须在 `transaction` 命令将搜索结果重新按时间降序排列之前立即执行。

多值字段

如果事件中的其中一个字段是多值字段，则必须指定 `connected=false`，以便在创建交易时合并该字段的值。当多值字段中至少有一个值在事件之间发生重叠时，这些值会进行合并。例如，如果某个事件的多值字段的值包含 `a b c`，而另一事件的包含 `c d e`，则当指定了 `connected=false` 时，这些字段会被合并。

基本示例

1. 主机、时间范围及暂停时间均相同的交易

将具有相同 `host` 和 `cookie` 值，在 30 秒内发生，并且两个事件之间的暂停时间不超过 5 秒的搜索结果归为一组。

```
... | transaction host cookie maxspan=30s maxpause=5s
```

2. "from" 值、时间范围及暂停时间均相同的交易

将具有相同的 `"from"` 值，最大跨度为 30 秒，并且事件之间的暂停时间不超过 5 秒的搜索结果归为一组。

```
... | transaction from maxspan=30s maxpause=5s
```

3. 字段值相同的交易

若事件含 `alert_level`，而您想要新建告警级别相同的交易，使用 `streamstats` 命令可以帮您记住当前事件和上一个事件的告警级别值。您可以使用 `transaction` 命令在告警级别不同时新建交易。将特定的字段输出成表格。

```
... | streamstats window=2 current=t latest(alert_level) AS last earliest(alert_level) AS first | transaction  
endswith=eval(first!=last) | table _time duration first last alert_level eventcount
```

延伸示例

1. 基于 IP 地址的 Web 访问事件交易

此示例使用搜索教程中的示例数据，但应该可以与任意格式的 Apache Web 访问日志配合使用。要在您自己的 Splunk 实例中尝试此示例，您必须下载示例数据并遵循操作以将教程数据导入 Splunk。当运行搜索时，请使用时间范围昨天。

基于共用同一 IP 地址的 Web 访问事件定义一个交易。交易中的第一个和最后一个事件的时间差不得超过三十秒，并且每个事件的时间不能大于五秒。

```
sourcetype=access_* | transaction clientip maxspan=30s maxpause=5s
```

将生成以下事件列表。交易中每个事件的客户端 IP 突出显示。

列表	格式	每页 20 个	< 上一个	1	2	3	4	5	6	7	8	... 下一步 >
< 隐藏字段	≡ 所有字段	i 时间	事件									
选定字段												
# host 3												
# source 3												
# sourcetype 1												
感兴趣的字段												
# action 5												
# bytes 100+												
# categoryid 8												
# clientip 100+												
# closed_bxn 2												
# date_hour 19												
# date_mday 1												
# date_minute 60												
# date_month 1												
# date_second 60												
# date_wday 1												
# date_year 1												
# date_zone 1												
# duration 16												
# eventcount 21												
# field_match_sum 21												
# file 14												
# ident 1												
# index 1												
# itemid 14												
# JSESSIONID 100+												
# linecount 21												
# method 2												
# other 100+												
# productid 16												

此搜索基于访问服务器的 IP 地址和时间约束对事件进行分组。搜索结果的某些字段，如 host 和 source 可能具有多个值。例如，如果有多个顾客来自同一办公室，那么从单个 IP 发出的请求可能来自多个主机。更多信息请参阅《知识管理器手册》中的“关于交易”主题。

2. 基于主机和客户 IP 的 Web 访问事件交易

此示例使用搜索教程中的示例数据，但应该可以与任意格式的 Apache Web 访问日志配合使用。要在您自己的 Splunk 实例中尝试此示例，您必须下载示例数据并遵循操作以将教程数据导入 Splunk。当运行搜索时，请使用时间范围昨天。

基于具有 host 和 clientip 值唯一组合的 Web 访问事件定义一个交易。交易中的第一个和最后一个事件的时间差不得超过三十秒，并且每个事件的时间不能大于五秒。

```
sourcetype=access_* | transaction clientip host maxspan=30s maxpause=5s
```

此搜索产生以下事件列表。

列表			格式	每页 20 个	< 上一个 1 2 3 4 5 6 7 8 ... 下一步 >										
i	时间	事件													
>	18/06/04 18:22:15.000	91.205.189.15 - - [04/Jun/2018:18:22:15] "GET /category.screen?categoryId=SHOOTER&JSESSIONID=SD6SL7FF7ADFF53113 HTTP 1.1" 200 1369 "http://www.google.com" "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/536.5 (KHTML, like Gecko) Chrome/19.0.1084.46 Safari/536.5" 779 91.205.189.15 - - [04/Jun/2018:18:22:15] "GET /category.screen?categoryId=SHOOTER&JSESSIONID=SD6SL7FF7ADFF53113 HTTP 1.1" 200 1369 "http://www.google.com" "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/536.5 (KHTML, like Gecko) Chrome/19.0.1084.46 Safari/536.5" 779 91.205.189.15 - - [04/Jun/2018:18:22:16] "GET /oldlink?itemId=EST-14&JSESSIONID=SD6SL7FF7ADFF53113 HTTP 1.1" 200 1665 "http://www.buttercupgames.com/oldlink?itemId=EST-14" "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/536.5 (KHTML, like Gecko) Chrome/19.0.1084.46 Safari/536.5" 159 91.205.189.15 - - [04/Jun/2018:18:22:16] "GET /oldlink?itemId=EST-14&JSESSIONID=SD6SL7FF7ADFF53113 HTTP 1.1" 200 1665 "http://www.buttercupgames.com/oldlink?itemId=EST-14" "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/536.5 (KHTML, like Gecko) Chrome/19.0.1084.46 Safari/536.5" 159 host = www2 source = tutorialdata.zip:/www2/access.log sourcetype = access_combined_wcookie													
>	18/06/04 18:20:50.000	182.236.164.11 - - [04/Jun/2018:18:20:50] "GET /category.screen?categoryId=STRATEGY&JSESSIONID=SD6SL8FF10ADFF53101 HTTP 1.1" 200 1200 "http://www.google.com" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_7_4) AppleWebKit/536.5 (KHTML, like Gecko) Chrome/19.0.1084.46 Safari/536.5" 490 182.236.164.11 - - [04/Jun/2018:18:20:50] "GET /category.screen?categoryId=STRATEGY&JSESSIONID=SD6SL8FF10ADFF53101 HTTP 1.1" 200 1200 "http://www.google.com" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_7_4) AppleWebKit/536.5 (KHTML, like Gecko) Chrome/19.0.1084.46 Safari/536.5" 490 182.236.164.11 - - [04/Jun/2018:18:20:52] "GET /product.screen?productId=MB-AG-G07&JSESSIONID=SD6SL8FF10ADFF53101 HTTP 1.1" 200 1035 "http://www.buttercupgames.com/category.screen?categoryId=ARCADE" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_7_4) AppleWebKit/536.5 (KHTML, like Gecko) Chrome/19.0.1084.46 Safari/536.5" 461 182.236.164.11 - - [04/Jun/2018:18:20:52] "GET /oldlink?itemId=EST-19&JSESSIONID=SD6SL8FF10ADFF53101 HTTP 1.1" 200 1653 "http://www.buttercupgames.com/oldlink?itemId=EST-19" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_7_4) AppleWebKit/536.5 (KHTML, like Gecko) Chrome/19.0.1084.46 Safari/536.5" 915 182.236.164.11 - - [04/Jun/2018:18:20:52] "GET /product.screen?productId=MB-AG-G07&JSESSIONID=SD6SL8FF10ADFF53101 HTTP 1.1" 200 1035 "http://www.buttercupgames.com/category.screen?categoryId=ARCADE" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_7_4) AppleWebKit/536.5 (KHTML, like Gecko) Chrome/19.0.1084.46 Safari/536.5" 461 显示所有 22 行 host = www1 source = tutorialdata.zip:/www1/access.log sourcetype = access_combined_wcookie													

在搜索指定的时间约束限制内，这些事件中的每个事件的 IP 地址 (clientip) 值和 host 值组合都不同。

3. 基于 IP 地址和时间范围的购买交易

此示例使用搜索教程中的示例数据，但应该可以与任意格式的 Apache Web 访问日志配合使用。要在您自己的 Splunk 实例中尝试此示例，您必须下载示例数据并遵循操作以将教程数据导入 Splunk。当运行搜索时，请使用时间范围昨天。

此搜索定义一个购买交易，该交易包含在 10 分钟时间跨度内发生的来自同一 IP 地址的 3 个事件。

```
sourcetype=access_* action=purchase | transaction clientip maxspan=10m maxevents=3
```

此搜索根据含 action=purchase 值的 Web 访问事件定义购物事件。然后通过管道符把结果传递给 transaction 命令。此搜索通过共用同一 clientip 的事件识别购买交易，其中每个会话的时间不超过 10 分钟，且包含的事件数量不超过 3 个。

此搜索产生以下事件列表：

列表			格式	每页 20 个	< 上一个 1 2 3 4 5 6 7 8 ... 下一步 >										
i	时间	事件													
a host 3 a source 3 a sourcetype 1 感兴趣的字段 a action 1 a bytes 100+ a categoryd 8 a clientip 100+ # closed_txn 2 # date_hour 19 # date_mday 1 # date_minute 60 a date_month 1 # date_second 60 a date_wday 1 # date_year 1 a date_zone 1 # duration 13 # eventcount 3 # field_match_sum 3 a file 8 a ident 1 a index 1 a itemid 14 a JSESSIONID 100+ # linecount 3 a method 2 a other 100+ a productid 15 a print 10 a referer 100+ a referer_domain 1		182.236.164.11 - - [04/Jun/2018:18:20:54] "POST /cart.do?action=purchase&itemId=EST-6&JSESSIONID=SD6SL8FF10ADFF53101 HTTP 1.1" 200 1803 "http://www.buttercupgames.com/cart.do?action=addtocart&itemId=EST-6&categoryId=ARCADE&productId=MB-AG-G07" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_7_4) AppleWebKit/536.5 (KHTML, like Gecko) Chrome/19.0.1084.46 Safari/536.5" 524 182.236.164.11 - - [04/Jun/2018:18:20:54] "POST /cart.success?doJSESSIONID=SD6SL8FF10ADFF53101 HTTP 1.1" 200 356 "http://www.buttercupgames.com/cart.do?action=purchase&itemId=EST-6" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_7_4) AppleWebKit/536.5 (KHTML, like Gecko) Chrome/19.0.1084.46 Safari/536.5" 220 host = www1 source = tutorialdata.zip:/www1/access.log sourcetype = access_combined_wcookie													
>	18/06/04 18:20:54.000	182.236.164.11 - - [04/Jun/2018:18:20:54] "POST /cart.do?action=purchase&itemId=EST-6&JSESSIONID=SD6SL8FF10ADFF53101 HTTP 1.1" 200 1803 "http://www.buttercupgames.com/cart.do?action=addtocart&itemId=EST-6&categoryId=ARCADE&productId=MB-AG-G07" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_7_4) AppleWebKit/536.5 (KHTML, like Gecko) Chrome/19.0.1084.46 Safari/536.5" 524 host = www1 source = tutorialdata.zip:/www1/access.log sourcetype = access_combined_wcookie													
>	18/06/04 18:18:58.000	182.236.164.11 - - [04/Jun/2018:18:18:58] "POST /cart.do?action=purchase&itemId=EST-16&JSESSIONID=SD10SL2FF4ADFF53099 HTTP 1.1" 200 821 "http://www.buttercupgames.com/cart.do?action=addtocart&itemId=EST-16&categoryId=SIMULATION&productId=SC-MG-G18" "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/536.5 (KHTML, like Gecko) Chrome/19.0.1084.46 Safari/536.5" 178 198.35.1.75 - - [04/Jun/2018:18:18:59] "POST /cart.success?doJSESSIONID=SD10SL2FF4ADFF53099 HTTP 1.1" 200 2568 "http://www.buttercupgames.com/cart.do?action=purchase&itemId=EST-16" "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/536.5 (KHTML, like Gecko) Chrome/19.0.1084.46 Safari/536.5" 386 host = www1 source = tutorialdata.zip:/www1/access.log sourcetype = access_combined_wcookie													
>	18/06/04 18:18:57.000	198.35.1.75 - - [04/Jun/2018:18:18:57] "POST /cart.do?action=purchase&itemId=EST-27&JSESSIONID=SD10SL2FF4ADFF53099 HTTP 1.1" 200 3577 "http://www.buttercupgames.com/cart.do?action=addtocart&itemId=EST-27&categoryId=TEE&productId=MB-AG-T01" "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/536.5 (KHTML, like Gecko) Chrome/19.0.1084.46 Safari/536.5" 827 198.35.1.75 - - [04/Jun/2018:18:18:57] "POST /cart.success?doJSESSIONID=SD10SL2FF4ADFF53099 HTTP 1.1" 200 613 "http://www.buttercupgames.com/cart.do?action=purchase&itemId=EST-27" "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/536.5 (KHTML, like Gecko) Chrome/19.0.1084.46 Safari/536.5" 328 198.35.1.75 - - [04/Jun/2018:18:18:58] "POST /cart.do?action=purchase&itemId=EST-16&JSESSIONID=SD10SL2FF4ADFF53099 HTTP 1.1" 200 821 "http://www.buttercupgames.com/cart.do?action=addtocart&itemId=EST-16&categoryId=SIMULATION&productId=SC-MG-G18" "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/536.5 (KHTML, like Gecko) Chrome/19.0.1084.46 Safari/536.5" 178 host = www1 source = tutorialdata.zip:/www1/access.log sourcetype = access_combined_wcookie													

有 2 个事件的交易

有 3 个事件的交易

4. 基于 `maxevents` 和 `endswith` 的电子邮件交易

此示例使用示例电子邮件数据。您可以将 `sourcetype=cisco:esa` 替换为 `sourcetype` 值并将 `mailfrom` 字段替换为您数据的电子邮件地址字段名，从而实现对任何电子邮件数据运行此搜索。例如：电子邮件可能为 To、From 或 Cc）。

此示例将电子邮件交易定义为最多 10 个事件的组。每个事件的 `mid`（消息 ID）、`icid`（进来连接 ID）和 `dcid`（交付连接 ID）值都相同。交易中的最后事件包含 `消息已完成` 字符串。

```
sourcetype="cisco:esa" | transaction mid dcid icid maxevents=10 endswith="Message done"
```

此搜索产生以下事件列表：

The screenshot shows a Splunk search interface titled "新搜索". The search bar contains the query: `source="cisco_esa.txt" | transaction mid dcid icid maxevents=10 endswith="Message done"`. Below the search bar, it says "4,999 个事件 (18/06/05 14:48:04.000 之前) 无事件采样" and "所有时间". The main pane displays a table of events with columns: 时间 (Time), 事件 (Event). The table shows several log entries from March 19, 2018, at 18:55:36, 18:55:34, 18:55:33, 18:55:26, 18:55:25, and 18:55:24. The last event in each group is highlighted with a red box around the text "Message done". The table includes pagination controls at the bottom: "每页 20 个" (20 per page), "1 2 3 4 5 6 7 8 ... 下一步 >".

时间	事件
18/03/19 18:55:36.000	Mon Mar 19 18:55:36 2018 Info: Message finished MID 19990744 done duration = 0 host = 127.0.0.1 source = cisco_esa.txt sourcetype = cisco_esa
18/03/19 18:55:34.000	Mon Mar 19 18:55:34 2018 Info: Message done DCID 8751025 MID 19990414 to RID [0, 1] duration = 0 host = 127.0.0.1 source = cisco_esa.txt sourcetype = cisco_esa
18/03/19 18:55:33.000	Mon Mar 19 18:55:33 2018 Info: Message done DCID 8778227 MID 20038555 to RID [0] duration = 0 host = 127.0.0.1 source = cisco_esa.txt sourcetype = cisco_esa
18/03/19 18:55:26.000	Mon Mar 19 18:55:26 2018 Info: Message finished MID 19990744 done duration = 0 host = 127.0.0.1 source = cisco_esa.txt sourcetype = cisco_esa
18/03/19 18:55:25.000	Mon Mar 19 18:55:25 2018 Info: Message finished MID 19991600 done duration = 0 host = 127.0.0.1 source = cisco_esa.txt sourcetype = cisco_esa
18/03/19 18:55:24.000	Mon Mar 19 18:55:24 2018 Info: Message finished MID 20038555 done duration = 0 host = 127.0.0.1 source = cisco_esa.txt sourcetype = cisco_esa

默认情况下，只显示交易中的前 5 个事件。第一个交易包含 7 个事件，最后事件被隐藏。第二个和第三个交易在交易中最后事件中显示 `消息已完成` 字符串。

5. 基于 `maxevents`、`maxspan` 和 `mvlist` 的电子邮件交易

此示例使用示例电子邮件数据。您可以将 `sourcetype=cisco:esa` 替换为 `sourcetype` 值并将 `mailfrom` 字段替换为您数据的电子邮件地址字段名，从而实现对任何电子邮件数据运行此搜索。例如：电子邮件可能为 To、From 或 Cc）。

此示例将电子邮件交易定义为最多 10 个事件的组。每个事件的 `mid`（消息 ID）、`icid`（进来连接 ID）和 `dcid`（交付连接 ID）值都相同。交易中的第一个和最后一个事件相隔应不超过三十秒。

```
sourcetype="cisco:esa" | transaction mid dcid icid maxevents=10 maxspan=30s mvlist=true
```

默认情况下，`mvlist` 默认为 `false` 的情况下，搜索结果中将抑制多值字段的值。在本搜索中指定 `mvlist=true` 将显示选定字段的所有值。将生成以下事件列表：

<隐藏字段	所有字段	i 时间	事件
选定字段 # duration 1 a host 1 a source 1 a sourcetype 1		> 3/19/18 8:18:04.000	Mon Mar 19 20:18:28 2018 Info: MID 19990410 RID [8] Response '2.6.0 <7436c832-4f04-4385-a6a1-980358db5a51> Queued mail for delivery' duration = 0 host = buttercup-mbpr15.sv.splunk.com source = cisco_esa.txt sourcetype = cisco:esa
感兴趣的字段 # closed_txn 1 # date_hour 3 # date_mday 1 # date_minute 60 a date_month 1 # date_second 60 a date_wday 1 # date_year 1 a date_zone 1 # eventcount 1 # field_match_sum 1 a index 1 # linecount 1 a punct 4 a splunk_server 1 # timeendpos 1 # timestamppos 1		> 3/19/18 8:18:04.000	Mon Mar 19 20:18:12 2018 Info: MID 19991599 Subject "Path for the Ranbaxy cases" duration = 0 host = buttercup-mbpr15.sv.splunk.com source = cisco_esa.txt sourcetype = cisco:esa
		> 3/19/18 8:17:56.000	Mon Mar 19 20:17:56 2018 Info: Start MID 19990413 ICID 26006572 duration = 0 host = buttercup-mbpr15.sv.splunk.com source = cisco_esa.txt sourcetype = cisco:esa
		> 3/19/18 8:17:56.000	Mon Mar 19 20:17:56 2018 Info: MID 20038555 queued for delivery duration = 0 host = buttercup-mbpr15.sv.splunk.com source = cisco_esa.txt sourcetype = cisco:esa
		> 3/19/18 8:17:46.000	Mon Mar 19 20:17:46 2018 Info: MID 19990410 using engine: CASE span negative duration = 0 host = buttercup-mbpr15.sv.splunk.com source = cisco_esa.txt sourcetype = cisco:esa
		> 3/19/18 8:17:31.000	Mon Mar 19 20:17:31 2018 Info: MID 19990744 ready 15204 bytes from <notify@example.com> Mon Mar 19 20:17:33 2018 Info: Message finished MID 19990744 done Mon Mar 19 20:17:57 2018 Info: RPC Message done RCID 8413024 MID 19990744 duration = 26 host = buttercup-mbpr15.sv.splunk.com host = buttercup-mbpr15.sv.splunk.com host = buttercup-mbpr15.sv.splunk.com source = cisco_esa.txt source = cisco_esa.txt sourcetype = cisco:esa sourcetype = cisco:esa sourcetype = cisco:esa sourcetype = cisco:esa

从上图中，您可以看到每个交易的持续时间都小于三十秒。同样，如果某个字段具有多个值，也会列出每一个值。

6. 会话 ID 及 IP 地址相同的交易

此示例使用搜索教程中的示例数据，但应该可以与任意格式的 Apache Web 访问日志配合使用。要在您自己的 Splunk 实例中尝试此示例，您必须下载示例数据并遵循操作以将教程数据导入 Splunk。当运行搜索时，请使用时间范围所有时间。

将会话 ID (JSESSIONID)，且来自相同的 IP 地址 (clientip) 的一组事件定义为一个交易，其中第一个事件包含字符串 "view"，最后一个事件包含字符串 "purchase"。

```
sourcetype=access_* | transaction JSESSIONID clientip startswith="view" endswith="purchase" | where duration>0
```

此搜索使用 startswith="view" 参数将交易中的第一个事件定义为包含字符串 "view" 的事件。endswith="purchase" 参数对交易中的最后一个事件执行相同的操作。

之后，此示例通过管道符把交易传递给 where 命令和 duration 字段，由此过滤掉完成时间不到一秒的所有交易。where 过滤器无法在 transaction 命令之前应用，因为 duration 字段由 transaction 命令添加。

<隐藏字段	所有字段	i 时间	事件
选定字段 # duration 13 a host 3 a source 3 a sourcetype 1		> 18/06/04 18:18:58.000	198.35.1.75 - - [04/Jun/2018:18:18:58] "GET /cart.do?action=view&itemId=EST-12&JSESSIONID=SD10SL2FF4ADFF53099 HTTP/1.1" 406 3907 "http://www.buttercupgames.com/product.screen?productId=Sf-BVS-G01" "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/536.5 (KHTML, like Gecko) Chrome/19.0.1084.46 Safari/536.5" 959 198.35.1.75 - - [04/Jun/2018:18:18:59] "POST /cart/success.do?JSESSIONID=SD10SL2FF4ADFF53099 HTTP/1.1" 200 2568 "http://www.buttercupgames.com/cart.do?action=purchase&itemId=EST-16" "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/536.5 (KHTML, like Gecko) Chrome/19.0.1084.46 Safari/536.5" 386
感兴趣的字段 a action 5 a bytes 100+ a categoryid 8 a clientip 100+ # closed_tnx 1 # date_hour 24 # date_mday 6 # date_minute 60 a date_month 20 a date_second 60 a date_wday 6 # date_year 1 a date_zone 1 # eventcount 19 # field_match_sum 19 a file 14 a ident 1 a index 1			duration = 1 host = www1 source = tutorialdata.zip:/www1/access.log sourcetype = access_combined_wcookie
		> 18/06/04 18:18:55.000	198.35.1.75 - - [04/Jun/2018:18:18:55] "GET /product.screen?productId=Sf-BVS-G01&JSESSIONID=SD10SL2FF4ADFF53099 HTTP/1.1" 500 2809 "http://www.buttercupgames.com/cart.do?action=view&itemId=EST-14" "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/536.5 (KHTML, like Gecko) Chrome/19.0.1084.46 Safari/536.5" 370 198.35.1.75 - - [04/Jun/2018:18:18:56] "POST /cart/do?action=addtocart&itemId=EST-27&productId=MB-AG-T01&JSESSIONID=SD10SL2FF4ADFF53099 HTTP/1.1" 200 2615 "http://www.buttercupgames.com/product.screen?productId=MB-AG-T01" "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/536.5 (KHTML, like Gecko) Chrome/19.0.1084.46 Safari/536.5" 838 198.35.1.75 - - [04/Jun/2018:18:18:56] "GET /product.screen?productId=SC-MG-G10&JSESSIONID=SD10SL2FF4ADFF53099 HTTP/1.1" 200 3675 "http://www.buttercupgames.com/category.screen?categoryId=SIMULATION" "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/536.5 (KHTML, like Gecko) Chrome/19.0.1084.46 Safari/536.5" 328 198.35.1.75 - - [04/Jun/2018:18:18:56] "GET /cart.do?action=adddtocart&itemId=EST-6&productId=FS-SG-G03&JSESSIONID=SD10SL2FF4ADFF53099 HTTP/1.1" 200 1159 "http://www.buttercupgames.com/product.screen?productId=FS-SG-G03" "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/536.5 (KHTML, like Gecko) Chrome/19.0.1084.46 Safari/536.5" 778 198.35.1.75 - - [04/Jun/2018:18:18:56] "POST /cart.do?action=adddtocart&itemId=EST-27&productId=MB-AG-T01&JSESSIONID=SD10SL2FF4ADFF53099 HTTP/1.1" 200 2615 "http://www.buttercupgames.com/product.screen?productId=MB-AG-T01" "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/536.5 (KHTML, like Gecko) Chrome/19.0.1084.46 Safari/536.5" 838 显示所有 8 行
			duration = 2 host = www1 source = tutorialdata.zip:/www1/access.log sourcetype = access_combined_wcookie

您可能会关心交易为何会花费很长的时间，因此，查看这些事件可能会帮助您解决问题。

此数据中并未显示，但某些交易花费很长时间的原因可能是，用户在完成购买之前正在更新和删除购物车中的商品。另外，此搜索在所有事件上运行。transaction 命令之前没有筛选。在您可以筛选首个管道前的搜索时，搜索运行更快。

另请参阅

引用

《搜索手册》中的“关于交易”。

命令

```
stats  
concurrency
```

transpose

描述

将指定的行（搜索结果）数以列（字段值列表）形式返回，以便每个搜索行都变成一列。

语法

要求的语法以**粗体**表示。

```
transpose  
[int]  
[column_name=<string>]  
[header_field=<field>]  
[include_empty=<bool>]
```

必要参数

无。

可选参数

column_name

语法: `column_name=<string>`
描述: 您想要用于转置行的第一列的名称。该列包含字段的名称。
默认值: column

header_field

语法: `header_field=<field>`
描述: 您的结果中要用于转置数据中各列（第一列除外）的名称的字段。
默认值: 行 1、行 2、行 3 等等。

include_empty

语法: `include_empty=<bool>`
描述: 指定是否包含 (true) 或排除 (false) 含空值的字段。
默认值: true

整型

语法: `<int>`
描述: 限制要转置的行数。要转置所有行，请指定 | transpose 0，表示需转置行数不受限。
默认值: 5

用法

当您使用 `transpose` 命令，输出中使用的字段名称取决于您与命令一起使用的参数。默认情况下，字段名称是: column、row1、row2 等。

示例

1. 转置图表命令的结果

使用 `transpose` 命令的默认设置转置 `chart` 命令的结果。

```
... | chart count BY host_error_code | transpose
```

2. 按 sourcetype 计入事件数量，并转置结果以显示 3 个最高计数

按 `sourcetype` 统计事件数量，并按计数显示各 `sourcetype`（计数最高的排第一）。

```
index=_internal | stats count by sourcetype | sort -count
```

事件	模式	统计信息 (11)	可视化
每页 20 个	✓ 格式	预览	
sourcetype			count
splunkd			2846239
splunkd_ui_access			55726
splunkd_access			10285
mongod			2905
splunk_web_service			2703
splunk_web_access			2110
scheduler			318
splunkd_stderr			5
first_install-too_small			4
splunkd_conf			3
splunk_version			1

使用 `transpose` 命令把行转换成列，并显示计数最高的 3 个来源类型。

```
index= internal | stats count by sourcetype | sort -count | transpose 3
```

事件	模式	统计信息 (2)	可视化	
每页 20 个	格式	预览		
column	row 1		row 2	row 3
sourcetype	splunkd	splunkd_ui_access	splunkd_access	
count	2845900	55676	10283	

3. 将一组数据转置为一个系列以生成一个图表

此示例将使用搜索教程中的示例数据。

- 从添加数据教程下载数据集，并按照说明将教程数据导入到 Splunk 部署。

搜索所有成功的事件，并计入浏览次数、将物品添加到购物车的次数和购买次数。

```
sourcetype=access_* status=200 | stats count AS views count(eval(action="addtocart")) AS addtocart count(eval(action="purchase")) AS purchases
```

此搜索生成单行数据。

count AS views 的值是和条件 sourcetype=access_* status=200 匹配的事件总数量或者所有操作的总数。addtocart 和 purchases 的值显示这些特定操作的事件数量。

当您切换到“可视化”选项卡时，数据显示一个图表（其中“34282 浏览”作为 X 轴标签）和两列，一列用于“添加到购物车”，另一列用于“购买”。因为有关浏览的信息显示在 X 轴，所以这个图表很混乱。



如果更改为饼图，则只显示“浏览”。



使用 transpose 命令将单行的列转换为多行。

```
sourcetype=access_* status=200 | stats count AS views count(eval(action="addtocart")) AS addtocart count(eval(action="purchase")) AS purchases | transpose
```

Q 新搜索

```
sourcetype=access_* status=200 | stats count AS views count(eval(action="addtocart")) AS addtocart count(eval(action="purchase")) AS purchases | transpose
```

✓ 34,282 个事件 (16/12/06 6:44:15.000 之前) 无事件采样

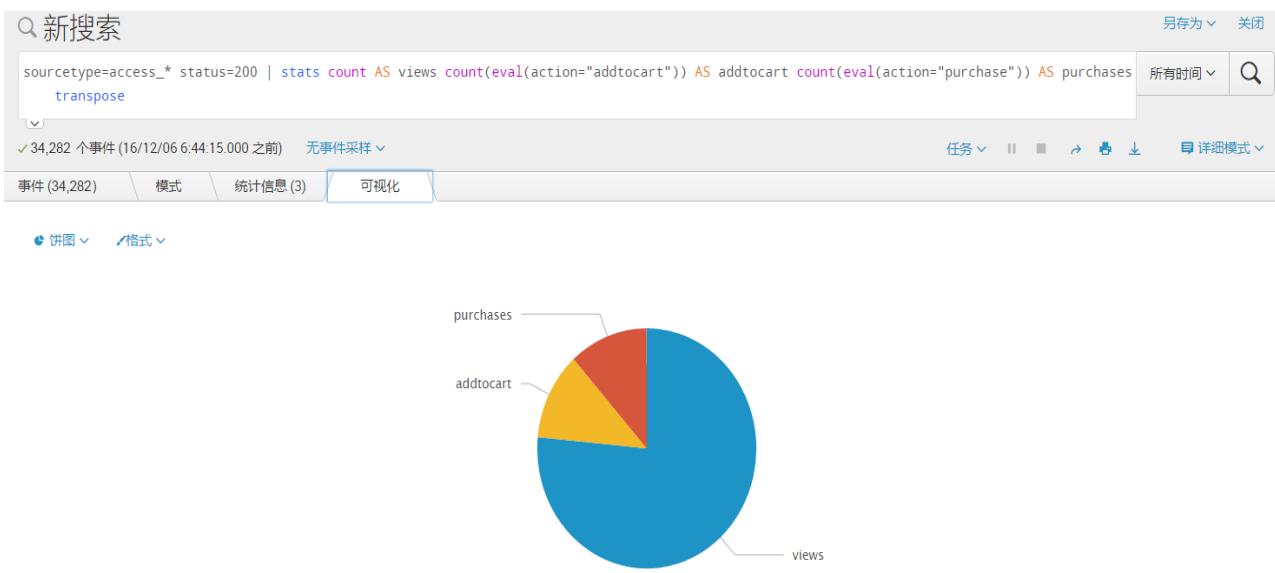
事件 (34,282) 模式 统计信息 (3) 可视化

任务 所有时间 详细模式

每页 20 个 格式 预览

column	views	addtocart	purchases
views			
addtocart			
purchases			

现在，这些行可以显示在列或饼图中，您可以在其中比较值。



在此特定示例中，使用饼图是具有误导性的。views 是所有操作的总计数，不仅仅是 addtocart 和 purchases 操作。使用饼图意味着 views 是一个类似 addtocart 和 purchases 的操作。饼图意味着事实上，当 views 是总数时，views 的值是总数的一部分。

有几种方法可以解决这个问题：

- 使用柱形图
- 您可将 count AS views 条件从搜索中删除
- 您可以将 table 命令添加到搜索中的 transpose 命令前面，例如：

```
sourcetype=access_* status=200 | stats count AS views count(eval(action="addtocart")) AS addtocart
count(eval(action="purchase")) AS purchases | table addtocart purchases | transpose
```

另请参阅

命令

字段
stats
untable
xyseries

trendline

描述

计算字段的移动平均线：简单移动平均线（sma）、指数移动平均线（ema）和加权移动平均线（wma）。输出将写入到新字段中，您可以指定该新字段。

SMA 和 WMA 均计算 period 内各最新值的总和。相较于过去值，WMA 更重视较新值。EMA 使用以下公式计算。

$$EMA(t) = alpha * EMA(t-1) + (1 - alpha) * field(t)$$

其中 $alpha = 2/(period + 1)$ 和 $field(t)$ 为字段的当前值。

语法

```
trendline ( <trendtype><period>"(<field>)" [AS <newfield>] )...
```

必要参数

trendtype

语法：sma | ema | wma

描述：要计算的趋势的类型。当前支持的趋势类型包括简单移动平均线（sma）、指数移动平均线（ema）和加权移动平均线（wma）。

period

语法: <num>

描述: 计算趋势所用的周期，一个介于 2 和 10000 之间的整数。

<field>

语法: "("<field>")"

描述: 用于计算趋势的字段名称。

可选参数

<newfield>

语法: <field>

描述: 指定要将输出写入到的新字段。

默认值: <trendtype><period>(<field>)

用法

示例

示例 1: 计算 'foo' 字段的五个事件简单移动平均线，并将结果写入名为 'smoothed_foo' 的新字段。另外，在同一平均线内计算 'bar' 字段的十个事件指数移动平均线。由于未指定 AS 子句，结果将写入 'ema10(bar)' 字段。

```
... | trendline sma5(foo) AS smoothed_foo ema10(bar)
```

示例 2: 在按月显示的事件图表中叠加一条趋势线。

```
index="bar" | stats count BY date_month | trendline sma2(count) AS trend | fields * trend
```

另请参阅

accum, autoregress, delta, streamstats

tscollect

已弃用此功能。

从版本 7.3.0 开始，Splunk 平台中已弃用 tscollect 命令。尽管此命令将继续运行，但未来版本中可能会删除。此命令已被数据模型取代。请参阅《知识管理器手册》中的“加速数据模型”。

在 7.3.0 版本说明中，请参见“已弃用的功能”。

描述

tscollect 命令使用索引字在您定义的命名空间中段新建时间系列索引 (tsidx) 文件。这些文件中的结果表是您已索引的数据的子集。因此，您之后可以使用 tstats 命令对这些 tsidx 文件执行搜索并提交报表，而不是搜索原始数据。由于您搜索的是完整索引的子集，因此搜索的完成速度应比之前的方式要快。

tscollect 命令在相同的命名空间内新建多个 tsidx 文件。当命令确定现在正在新建的 tsidx 文件变得足够大时，命令将开始新的 tsidx 文件。

只有有 indexes_edit 功能的用户可以运行此命令。请参阅“用法”。

语法

```
... | tscollect [namespace=<string>] [squashcase=<bool>] [keepresults=<bool>]
```

可选参数

keepresults

语法: keepresults = true | false

描述: 如果为 true，tscollect 将输出它作为输入收到的相同结果。如果为 false，tscollect 将返回处理的结果数（这样效率更高，因为不需要存储很多结果）。

默认值: false

namespace

语法: namespace=<string>

描述: 定义 tsidx 文件的位置。若提供了命名空间，tsidx 文件将写入主 tsidxstats 目录（亦即 \$SPLUNK_DB/tsidxstats 中）下该名称的目录中。这些命名空间可以写入多次以添加新数据。

默认值: 如果未提供命名空间，文件将写入该搜索的任务目录中的目录，并且存在时间将与任务的时间一样长。如果您有 Splunk Enterprise，您可以通过编辑 indexes.conf 和设置属性 tsidxStatsHomePath 来配置命名空间位置。

squashcase

语法: squashcase = true | false

描述: 指定整个 field::value 标记在放入字典时是否区分大小写。要使新建的索引字段 tsidx 文件与 Splunk Enterprise 新建的相似，请设置 squashcase=true，以使结果转换为全小写形式。

默认值: false

用法

您必须有 indexes_edit 功能才能运行 tscollect 命令。默认情况下，管理员角色有此功能，用户和超级用户角色没有此功能。

示例

示例 1: 将结果表写入命名空间 foo 中的 tsidx 文件。

```
... | tscollect namespace=foo
```

示例 2: 将检索主索引的值，并将字段 foo 的值写入任务目录中的 tsidx 文件。

```
index=main | fields foo | tscollect
```

另请参阅

collect, stats, tstats

tstats

描述

使用 tstats 命令对 tsidx 文件中的索引字段执行统计查询。索引字段可以来自普通索引数据、tscollect 数据或加速数据模型。

语法

要求的语法以粗体表示。

```
| tstats
[prestats=<bool>]
[local=<bool>]
[append=<bool>]
[summariesonly=<bool>]
[include_reduced_buckets=<bool>]
[allow_old_summaries=<bool>]
[chunk_size=<unsigned int>]
[fillnull_value=<string>]
<stats-func>...
[FROM ( <namespace> | sid=<tscollect-job-id> | datamodel=<data_model_name>.<root_dataset_name> [where
nodename = <root_dataset_name>.<...>.<target_dataset_name>]) ]
[WHERE <search-query> | <field> IN (<value-list>)]
[BY (<field-list> | (PREFIX(<field>))) [span=<timespan>] ]
```

必要参数

<stats-func>

语法: (count [<field>] | <function>(PREFIX(<string>) | <field>))... [AS <string>]

描述: 执行字段的基本计数或对字段使用函数。要查看 tstats 命令支持的函数列表，请参阅下方表格。您必须指定一个或多个函数。您可以将函数应用于某个字段，或者如果想汇总索引事件中的原始段，好像它是一个提取的字段-值对一样，则可以将此函数应用于 PREFIX() 指令。您还可以使用 AS 关键字重命名结果，除非您正处于 prestats 模式 (prestats=true)。

如果没有函数应用于字段或解析为字段的 eval 表达式，则无法指定这些函数。您无法使用通配符指定字段名称。

请参阅“用法”，了解有关使用 `PREFIX()` 的更多信息，以及为在数据中查找原始段而可运行的搜索。

以下表格按照函数类型列出支持的函数。使用表格中的链接查看每个函数的介绍和示例。有关使用带有命令的函数的概述，请参见统计和图表函数。

函数类型	支持的函数和语法			
聚合函数	<code>avg()</code> <code>count()</code> <code>distinct_count()</code> <code>estdc()</code>	<code>exactperc<int>()</code> <code>max()</code> <code>median()</code> <code>min()</code> <code>mode()</code>	<code>perc<int>()</code> <code>range()</code> <code>stdev()</code> <code>stdevp()</code>	<code>sum()</code> <code>sumsq()</code> <code>upperperc<int>()</code> <code>var()</code> <code>varp()</code>
事件顺序函数	<code>first()</code>	<code>last()</code>		
多值统计和 <code>chart</code> 函数	<code>values()</code>			
时间函数	<code>earliest()</code> <code>earliest_time()</code>	<code>latest()</code> <code>latest_time()</code>	<code>rate()</code>	

可选参数

append

语法: `append=<bool>`

描述: 若在 `prestats` 模式 (`prestats=true`) 下启用 `append=true`, `prestats` 结果将附加到现有结果，而非生成 `prestats` 结果。

默认值: `false`

allow_old_summaries

语法: `allow_old_summaries=true | false`

描述: 仅当从加速数据模型中选择时才适用。要仅当这些目录是最新时才从摘要目录返回结果，请将此参数设置为 `false`。若数据模型定义已更改，则当从 `tstats` 生成输出时，将不再使用比新定义旧的摘要目录。此默认设置确保来自 `tstats` 的输出将始终反映您的当前配置。如果设置为 `true`, `tstats` 将使用当前摘要数据和在定义更改之前生成的摘要数据。在您知道旧摘要“足够好”的情况下，实际上这是一个高级性能功能。

默认值: `false`

chunk_size

语法: `chunk_size=<unsigned_int>`

描述: 高级选项。当 Splunk 软件处理搜索时，此参数控制一次从单个 `tsidx` 文件中检索多少个事件。只有当您发现特定的 `tstats` 搜索使用了过多的内存或不频繁地返回事件时，才可以将其设置从默认值降至更低一点。当搜索按过高的基数字段（具有大量不同值的字段）进行分组时，可能就会发生这种情况。发生这种情况时，较低的 `chunk_size` 值可使 `tstats` 搜索的响应速度更快，但完成的速度可能会变慢。另一方面，较高的 `chunk_size` 可以帮助长时间运行的搜索完成得更快，但可能也会因此导致搜索响应速度降低。对于 `tstats`, `chunk_size` 不能设置为小于 10000 的值。

默认值: 10000000 (1 千万)

`chunk_size` 参数的默认值由 `limits.conf` 的 `[tstats]` 段落中的 `chunk_size` 设置进行设置。

fillnull_value

描述: 此参数设置用户指定的值，`tstats` 命令用该值替换其 `group-by` 字段列表中任何字段的空值。空值包括返回的事件的子集中缺少的字段值，以及所有返回的事件中的字段值。如果未提供 `fillnull_value` 参数，则 `tstats` 会省略结果中的特定行，这些行包括带一个或多个空字段值的事件。

默认值: 空字符串

include_reduced_buckets

语法: `include_reduced_buckets=true | false`

描述: 仅当 `indexes.conf` 包含 `enableTSIDXReduction=true` 时，此设置才适用。设置为 `false` 时，`tstats` 命令会仅通过未减少的索引数据桶生成结果。如果希望 `tstats` 通过减少的存储桶中生成结果，则设置为 `true`。

默认值: `false`

local

语法: `local=true | false`

描述: 如果 `true`，仅强制在搜索头上运行处理器。

默认值: `false`

prestats

语法: `prestats=true | false`

描述: 指定是否使用 `prestats` 格式。`prestats` 格式是 Splunk 内部格式，设计用于可以生成聚合计算的命令使用。使

用 `prestats` 格式时，您可以将数据导成图表、数据或时间表命令，这些可以接受 `prestats` 格式。`prestats=true` 时，AS 说明不重要。聚合的字段名称由使用 `prestats` 格式的命令确定，产生聚合输出。

默认值：`false`

`summariesonly`

语法：`summariesonly=<bool>`

描述：仅当从加速数据模型中选择时才适用。如果设置为 `false`，则从汇总数据和未汇总的数据中生成结果。对于未汇总为 `TSIDX` 数据的数据，针对原始索引数据使用完整搜索行为。如果设置为 `true`，则 '`tstats`' 将仅从已由加速自动生成的 `TSIDX` 数据中生成结果，且将不会提供未汇总的数据。

默认值：`false`

FROM 子句参数

`FROM` 子句是可选的。您可以指定 `namespace`、`sid` 或 `datamodel`。请参阅“选择数据”查看有关此子句的更多信息。

`namespace`

语法：`<string>`

描述：使用 `$SPLUNK_DB/tsidxstats` 定义 `tsidx` 文件的位置。如果您有 Splunk Enterprise，您可以通过编辑 `indexes.conf` 的本地版本和设置属性 `tsidxStatsHomePath` 来配置此位置。请参阅《管理员手册》中的“如何编辑配置文件”。

`sid`

语法：`sid=<tscollect-job-id>`

描述：`tscollect` 搜索的任务 ID 字符串（生成的 `tsidx` 文件）。

`datamodel`

语法：`datamodel=<data_model_name>.<root_dataset_name> [where nodename = <root_dataset_name>.<...>.<target_dataset_name>]`

描述：数据模型的名称，与要搜索的根数据集的名称连接在一起。如果要过滤子数据集，则需要使用 `where` 子句，并且该子句使用 `nodename` 来引用数据模型中数据集层次结构中的特定子数据集。有关更多信息，请参阅“选择数据”。

WHERE 子句参数

`WHERE` 子句是可选的。此子句用作筛选器。您可以指定搜索或字段，以及带 `IN` 运算符的一组值。

`<search-query>`

指定筛选搜索条件。

`<field> IN (<value-list>)`

对于 field，请指定要包含在搜索结果中的值列表。

BY 子句参数

`BY` 子句是可选的。您无法在 `BY` 子句中与 `tstats` 命令结合使用通配符。请参阅“用法”。如果您使用 `BY` 子句，必须指定 `field-list`。您还可以指定 `span`。

`<field-list>`

语法：`<field>, ...`

描述：指定用于对结果进行分组的一个或多个字段。

`PREFIX()`

语法：`PREFIX(<string>)`

描述：在索引事件中指定要拆分的原始段，好像它是一个提取的字段-值对一样。请参阅“用法”，了解有关 `PREFIX()` 指令的更多信息，以及为在索引数据中查找原始段而可运行的搜索。

`span`

语法：`span=<timespan>`

描述：每个时间数据箱的跨度。如果您使用 `BY` 子句按照 `_time` 进行分类，使用 `span` 参数对时间数据桶进行分组。您可以指定时间范围，如 `BY _time span=1h` 或 `BY _time span=5d`。如果不指定 `<timespan>`，默认为 `auto`，这表示时间数据桶的数字适应于生产一个合理数量的结果。例如，如果最初的秒数用于 `<timespan>`，返回了结果过多，`<timespan>` 变为更长的值（如分钟）以返回更少的时间数据桶。

默认：自动

`<timespan>`

语法：`auto | <int><timescale>`

`<timescale>`

语法：`<sec> | <min> | <hr> | <day> | <month>`

描述：时间刻度单位。对于 `tstats` 命令，`<timescale>` 不支持亚秒。

默认值：`sec`

时间刻度	语法	描述
<sec>	s sec secs second seconds	时间刻度（秒）。
<min>	m min mins minute minutes	时间刻度（分钟）。
<hr>	h hr hrs hour hours	时间刻度（小时）。
<day>	d day days	时间刻度（天）。
<month>	mon month months	时间刻度（月）。

用法

tstats 命令是报表生成命令（prestats=true 时除外）。如果 prestats=true，tstats 命令是事件生成命令。请参阅“命令类型”。

生成命令使用前导管道符且应是搜索中的第一个命令，prestats=true 时除外。

通配符字符

tstats 命令不支持在聚合函数或 BY 子句中字段值的通配符字符。

例如，您不可以指定 | tstats avg(foo*) 或 | tstats count WHERE host=x BY source*。

聚合函数包括 avg()、count()、max()、min() 和 sum()。有关更多信息，请参阅“聚合函数”。

如果聚合函数或 BY 子句包含通配符字符，则所返回的任何结果都只是最新几分钟的数据，而且没有汇总。包括 summariesonly=t 参数（使用 tstats 命令）返回汇总数据。

统计函数必须有命名字段

除 count 外，tstats 命令只支持应用于字段或解析为字段的 eval 表达式的统计函数。例如，您不可以指定 | tstats sum 或 | tstats sum()。相反，tstats 语法要求为该函数至少提供一个字段参数：| tstats sum(<field>)。

不支持嵌套 eval 表达式

您不能将聚合函数中的 eval 表达式和 tstats 命令结合使用。

例如，不支持 | tstats count(eval(...))。

如果 stats 命令支持嵌套 eval 表达式，那么 tstats 命令不支持。

函数和内存用法

就内存而言，某些函数本身就比其他函数占用更多内存。例如，distinct_count 函数需要的内存比 count 函数大得多。values 和 list 函数也会消耗大量的内存。

如果您使用的是未结合 split-by 字段或结合按字段的低基数 split-by 的 distinct_count 函数，考虑用 estdc 函数替换 distinct_count 函数（估计非重复计数）。estdc 函数可降低内存使用和减少运行时间。

使用 PREFIX() 在索引数据中进行汇总或按原始标记分组

PREFIX() 指令让您可以在索引数据中搜索原始段，好像它是一个提取的字段-值对一样。这将使搜索运行于索引器中的 tsidx 文件而非日志行。这种做法可以大大减少索引器上的 CPU 负载。

PREFIX() 指令与 CASE() 和 TERM() 指令相似，都是与原始数据中的字符串进行匹配。您可以使用 PREFIX() 在原始事件数据中定位一个循环段，该循环段实际上是一个由分隔符分隔的键值对，该分隔符也是一个次要分隔符，例如 = 或 :。您给 PREFIX() 要添加在值之前的文本（即“前缀”），然后它会返回带前缀的值。这使您可以按这些值进行分组，并使用 tstats 函数对它们进行聚合。这些值可以是字符串或纯数字。

例如，假设您在事件数据中建立了索引段，诸如 kbps=10 或 kbps=333。您可以使用 PREFIX() 指令将 kbps= 标识为公共前缀字符串，从而隔离这些段中的数值并对其进行汇总或分组操作。使用 PREFIX(kbps=) 对事件数据运行 tstats 搜索，它将返回 10 和 333。这些值非常适合需要纯数字输入的 tstats 聚合函数。

注意，在此示例中，您需要包含一个= 分隔符。如果您运行 PREFIX(kbps)，搜索会返回 =10 和 =333。对此类结果进行聚合可能会返回意外的结果，特别是在通过需要纯数字值的聚合函数运行这些结果时。

您为 `PREFIX()` 指令提供的文本必须为小写。例如，`tstats` 搜索处理器将无法处理 `PREFIX(connectionType=)`。改用 `PREFIX(connectiontype=)`。它仍然会匹配事件中的 `connectionType=` 字符串。

Splunk 软件在使用 `segmenters.conf` 中指定的规则为数据建立索引时会将事件分为原始段。您可以运行以下搜索来识别索引事件中的原始段：

```
| walklex index=<target-index> type=term | stats sum(count) by term
```

您不能将 `PREFIX()` 指令应用于包含主要分隔符（如空格、方括号或大括号、括号、分号或感叹号）的段前缀和值。

有关 `CASE()` 和 `TERM()` 指令的更多信息，请参阅《搜索手册》中的“使用 `CASE()` 和 `TERM()` 匹配短语”。

有关索引事件分段的更多信息，请参阅《数据导入》中的“关于事件分段”。

有关分段中主要和次要分隔符的更多信息，请参阅《搜索手册》中的“事件分段和搜索”。

内存和 `tstats` 搜索性能

一对 `limits.conf` 设置在 `tstats` 搜索的性能及其在搜索过程中所占用的 RAM 和磁盘内存量之间取得平衡。如果 `tstats` 一直完成得很慢，则可以调整这些设置以提高其性能，但同时会增加搜索时的内存使用率，并可能因此导致搜索失败。

如果您有 Splunk Cloud，则需要提交支持工单，申请更改这些设置。

有关更多信息，请参阅《搜索手册》中的“内存和 `stats` 搜索性能”。

复杂的聚合函数

`tstats` 命令不支持复杂的聚合函数，如 `...count(eval('Authentication.action'=="failure"))`。

考虑以下查询。本查询不会返回准确的结果，因为 `tstats` 命令不支持复杂的聚合函数。

```
| tstats summariesonly=false values(Authentication.tag) as tag, values(Authentication.app) as app, count(eval('Authentication.action'=="failure")) as failure, count(eval('Authentication.action'=="success")) as success from datamodel=Authentication by Authentication.src | search success>0 | where failure > 5 | `settags("access")` | `drop_dm_object_name("Authentication")`
```

但是会将 `eval` 函数从聚合函数中独立出来，如以下搜索所示。

```
| tstats `summariesonly` values(Authentication.app) as app, count from datamodel=Authentication.Authentication by Authentication.action, Authentication.src | `drop_dm_object_name("Authentication")` | eval success=if(action="success",count,0), failure=if(action="failure",count,0) | stats values(app) as app, sum(failure) as failure, sum(success) as success by src
```

迷你图

只有在您指定 `BY` 子句中的 `_time` 字段并使用 `stats` 命令生成实际迷你图时，才可以通过 `tstats` 命令生成迷你图。例如：

```
| tstats count from datamodel=Authentication.Authentication BY _time, Authentication.src span=1h | stats sparkline(sum(count),1h) AS sparkline, sum(count) AS count BY Authentication.src
```

选择数据

使用 `tstats` 命令执行 `.tsidx` 文件中的索引字段上的统计查询。您可以通过多种方式选择索引字段的数据。

普通索引数据

使用 `FROM` 子句指定命名空间、搜索任务 ID 或数据模型。如果未指定 `FROM` 子句，Splunk 软件将以与 `search` 命令相同的方式从索引数据中进行选择。您只能通过用户角色从允许的索引中选择数据。通过使用 `WHERE` 子句，可精确控制从中选择数据的索引。如果 `WHERE` 子句搜索未提到任何索引，Splunk 软件将使用默认索引。默认情况下，应用基于角色的搜索过滤器，但是可在 `limits.conf` 文件中关闭。

使用 `tscollect` 命令手动收集的数据

您可以通过指定 `FROM <namespace>` 来从命名空间中选择数据。如果未使用 `tscollect` 命令指定命名空间，则会将数据收集到该任务的 `dispatch` 目录中。如果数据在 `dispatch` 目录中，则通过指定 `FROM sid=<tscollect-job-id>` 来选择数据。

加速数据模型

您可以从用于加速数据模型的高性能分析存储中选择数据，这是一个 `.tsidx` 数据摘要集合。您可以使用 `FROM datamodel=<data_model_name>.<root_dataset_name>` 从此加速数据模型中选择数据。

在为 `tstats` 搜索选择数据模型时，还必须在该数据模型中选择要搜索的根数据集。不可以一次选择一个数据模型中的所

有根数据集。

搜索过滤器不适用于加速数据模型。基于角色的搜索过滤器和基于用户的搜索过滤器均包括在内。

加速数据模型中的子数据集

您可以从加速数据模型的子数据集中选择数据。使用 WHERE 子句指定子数据集的 nodename。nodename 参数指示目标数据集在数据模型层次结构中的位置。其语法如下所示：

```
... | tstats <stats-func> FROM datamodel=<data_model_name>.〈root_dataset_name〉 where  
nodename=<root_dataset_name>.〈...〉.〈target_dataset_name〉
```

例如，假设您有一个包含三个根数据集的数据模型，每个根数据集都有自己的数据集层次结构。

```
ButtercupGamesPromos  
- NYC (BaseEvent)  
  - TShirtStore (NYC)  
    - FashionShows (TShirtStore)  
    - Giveaways (TShirtStore)  
- Chicago (BaseEvent)  
  - BeerAndBrautsPopup (Chicago)  
    - BeerSales (BeerAndBrautsPopup)  
    - BrautSales (BeerAndBrautsPopup)  
- Tokyo (BaseSearch)  
  - GiantRobotBattles (Tokyo)  
    - UFORobotGrendizer (GiantRobotBattles)  
    - MechaGodzilla (GiantRobotBattles)
```

在此层次结构中，如果要运行 tstats 搜索，而且该搜索要从包含由东京办公室上演的 MechaGodzilla 巨型机器人战役的记录的数据集中选择数据，则可以使用以下搜索：

```
... | tstats count FROM datamodel=ButtercupGamesPromos.Tokyo where nodename=Tokyo.GiantRobotBattles.MechaGodzilla
```

搜索过滤器不适用于加速数据模型数据集。基于角色的搜索过滤器和基于用户的搜索过滤器均包括在内。

当搜索 tsidx 文件时，您可能在检索的事件中看到计数不匹配。无法区分索引字段标记和 tsidx 文件中的原始标记。但若在加速数据模型上运行 tstats 命令或从 tscollect 命令上运行，效果将更显著，因为结果仅存储字段和值，不存储原始标记。

使用 WHERE 进行筛选

您可以提供任意数量要执行的聚合 (aggregate-opt)，也可以选择使用 WHERE 关键字提供筛选查询。此查询和您在搜索处理器中使用的普通查询近似。支持所有与搜索一样的时间参数，例如 earliest=-1y。

按 _time 分组

您可以提供任意数量的 BY 字段。若按 _time 分组，提供一个具有 span 的时间跨度，以对时间数据桶进行分组，例如 ...BY _time span=1h 或 ...BY _time span=3d。

Tstats 和 tsidx 数据桶减少

tstats 搜索如果针对经过 tsidx 数据桶减少的索引进行，则会返回错误结果。

有关更多信息，请参阅《管理索引器和索引器群集》中的“减少 tsidx 磁盘使用量”。

示例

示例 1：获取 mydata 命名空间中所有事件的计数。

```
| tstats count FROM mydata
```

示例 2：返回 foo 字段（位于 mydata 中）的平均值，尤其当 bar 为 value2 且 baz 的值大于 5 时。

```
| tstats avg(foo) FROM mydata WHERE bar=value2 baz>5
```

示例 3：为 host=x 的事件按数据来源提供计数。

```
| tstats count WHERE host=x BY source
```

示例 4：使用天粒度提供默认索引中所有数据的时间图表。

```
| tstats prestats=t count BY _time span=1d | timechart span=1d count
```

示例 5：将 `prestats` 模式与 `append` 结合使用，以计算 `foo` 和 `bar`（两者位于不同的命名空间中）的中值。

```
| tstats prestats=t median(foo) FROM mydata | tstats prestats=t append=t median(bar) FROM otherdata | stats median(foo) median(bar)
```

示例 6：使用 `summariesonly` 参数获取名为 `mydm` 的加速数据模型的摘要时间范围。

```
| tstats summariesonly=t min(_time) AS min, max(_time) AS max FROM datamodel=mydm | eval prettymin=strftime(min, "%c") | eval prettymax=strftime(max, "%c")
```

示例 7：将 `summariesonly` 与 `timechart` 结合使用，即可显示在过去一小时内为标题是 `mydm` 的加速数据模型汇总了哪些数据。

```
| tstats summariesonly=t prestats=t count FROM datamodel=mydm BY _time span=1h | timechart span=1h count
```

示例 8：使用 `values` 统计函数以提供“Splunk 内部服务器日志”数据模型返回的 `source` 的所有非重复值的列表。该列表以多值条目的形式返回。

```
| tstats values(source) FROM datamodel=internal_server
```

结果形式如下所示：

values (source)
/Applications/Splunk/var/log/splunk/license_usage.log
/Applications/Splunk/var/log/splunk/metrics.log
/Applications/Splunk/var/log/splunk/metrics.log.1
/Applications/Splunk/var/log/splunk/scheduler.log
/Applications/Splunk/var/log/splunk/splunkd.log
/Applications/Splunk/var/log/splunk/splunkd_access.log

如果未定义 `internal_server` 数据模型，请在设置->数据模型下方查看可访问的数据模型的列表。

示例 9：使用 `values` 统计函数以提供“Splunk 内部服务器日志”数据模型中的警告数据集返回的 `source` 的所有非重复值的列表。

```
| tstats values(source) FROM datamodel=internal_server where nodename=server.scheduler.alerts
```

示例 10：使用 `PREFIX kbps=` 获取未索引的原始术语的计数和平均值，然后使用 `PREFIX group=` 将其按索引源和另一个未索引术语进行拆分。

```
| tstats count avg(PREFIX(kbps=)) where index=_internal by source PREFIX(group=)
```

另请参阅

命令

```
datamodel  
stats  
tscollect  
walklex
```

typeahead

描述

返回指定前缀的键盘缓冲信息。返回结果的数量上限取决于您为 `count` 参数指定的值。可将索引作为 `typeahead` 命令的目标，并通过时间进行限制。

语法

```
| typeahead prefix=<string> count=<int> [max_time=<int>] [<index=<string>>] [<starttimeu=<int>>  
[<endtimeu=<int>>] [collapse=<bool>]
```

必要参数

prefix
语法: prefix=<string>
描述: 要返回 typeahead 信息的完整搜索字符串。

count
语法: count=<int>
描述: 要返回的最大结果数。

可选参数

index-specifier
语法: index=<string>
描述: 搜索指定索引而非默认索引。

max_time
语法: max_time=<int>
描述: typeahead 可以运行的最长时间, 以秒为单位。若 max_time=0, 则无限制。

starttimeu
语法: starttimeu=<int>
描述: 将开始时间设置为 N 秒, 以 UNIX 时间计。
默认值: 0

endtimeu
语法: endtimeu=<int>
描述: 将结束时间设置为 N 秒, 以 UNIX 时间计。
默认值: 现在

collapse
语法: collapse=<bool>
描述: 指定在事件计数相同时是否折叠充当另一个术语前缀的术语。
默认值: true

用法

typeahead 命令属于生成命令, 应该是搜索中的第一个命令。生成命令使用前导管道字符。

键盘缓冲和来源类型重命名

重命名 sourcetype (位于 props.conf 文件中) 后, 需要约 5 分钟 (具体时间可能略受服务器性能的影响) 清除缓存数据。在清理缓存时运行的 typeahead 搜索返回缓存来源类型数据。这属于预期行为。

要删除缓存数据, 请在终端窗口中运行以下命令:

```
rm $SPLUNK_HOME/var/run/splunk/typeahead/*, then re-run the typeahead search.
```

重新运行 typeahead 搜索后, 应该可以看到重命名的来源类型。

更多信息, 请参阅《数据导入》手册中的“重命名来源类型”。

Typeahead 和 tsidx 数据桶减少

typeahead 搜索如果针对经过 tsidx 数据桶减少的索引进行, 则会返回错误结果。

有关更多信息, 请参阅《管理索引器和索引器群集》中的“减少 tsidx 磁盘使用量”。

示例

示例 1:

返回 “_internal” 索引中来源的键盘缓冲信息。

```
| typeahead prefix=source count=10 index=_internal
```

typelearner

Splunk Enterprise 从版本 5.0 开始弃用 typelearner 命令。这意味着，尽管这个命令还可以运行，但可能在未来版本中删除。

改用 findtypes 命令。

描述

通过使用以前的搜索结果并新建一个可用作事件类型的潜在搜索列表，来生成建议的事件类型。根据默认设置，typelearner 命令最初按分组字段的值对事件进行分组。然后根据它们所含的关键字进一步统一并合并这些组。

语法

```
typelearner [<grouping-field>] [<grouping-maxlen>]
```

可选参数

grouping-field

语法: <field>

描述: 包含 typelearner 命令最初分组事件时所使用值的字段。

默认值: punct, 即所看到的标点符号, 存在于 _raw

grouping-maxlen

语法: maxlen=<int>

描述: 决定要查看的分组字段值中的字符数。如果设置为负数，则使用分组字段值总值进行事件分组。

默认值: 15

示例

示例 1:

让搜索自动发现并将事件类型应用到搜索结果。

```
... | typelearner
```

另请参阅

typer

typer

描述

计算与已知事件类型匹配的搜索结果的 'eventtype' 字段。

语法

```
typer [eventtypes=<string>] [maxlen=<unsigned_integer>]
```

可选参数

eventtypes

语法: eventtypes=<string>

描述: 提供用逗号分隔的事件类型列表，以过滤 typer 可以在 eventtype 字段中返回的一组事件类型。eventtypes 参数会过滤掉列表中除有效事件类型以外的所有事件类型。如果针对 eventtypes 列出的所有事件类型均无效，或者未列出任何事件类型，则会禁用 typer，并且不会返回任何事件类型。eventtypes 参数接受通配符。

默认值: 无默认值（默认情况下，typer 会返回所有可用的事件类型）

maxlen

语法: maxlen=<unsigned_integer>

描述: 默认情况下，typer 命令会查看事件的前 10000 个字符以确认其事件类型。使用 maxlen 覆盖此默认值。例如，maxlen=300 会将 typer 限制为根据事件的前 300 个字符确定事件类型。

用法

typer 命令属于可分配的流命令。请参阅“命令类型”。

更改 maxlen 的默认设置

具有文件系统访问权限的用户，如系统管理员可以更改 maxlen 的默认设置。

前提条件

- 只有具有文件系统访问权限的用户，如系统管理员，可以使用配件文件更改 maxlen 的默认设置。
- 请参阅 Splunk Enterprise 《管理员手册》中的“如何编辑配置文件”了解具体步骤。
- 您可以有几个具有相同名称的配置文件，分散在默认目录、本地目录和应用目录中。请参阅 Splunk Enterprise 《管理员手册》中“在何处放置（或查找）已修改的配置文件”。

不要更改或复制默认目录中的配置文件。默认目录中的文件必须保持原样并位于其原始位置。更改本地目录中的文件。

步骤

1. 在 \$SPLUNK_HOME/etc/apps/search/local 为“搜索”应用打开或新建本地 limits.conf 文件。
2. 在 [typer] 段落中，为 maxlen 设置指定默认设置。

如果您使用的是 Splunk Cloud 并想更改默认设置，打开支持工单。

示例

示例 1:

强制搜索应用程序已配置的事件类型（当您查看 "eventtype" 字段时，Splunk Web 会自动执行此操作）。

```
... | typer
```

另请参阅

命令

typelearner

union

描述

将两个或两个以上数据集中的结果合并到一个数据集中。其中一个数据集可以是一个结果集，然后通过管道符将结果传递给 union 命令并和第二个数据集合并。

union 命令附加或合并指定数据集中的事件，基于数据集是流数据集还是非流数据集，以及命令在哪里运行。union 命令并行在索引器上运行（如可行），并在处理事件时自动交错 _time 结果。请参阅“用法”。

如果您对 SQL 很熟悉，但对 SPL 较为陌生，请参阅“面向 SQL 用户的 Splunk SPL”。

语法

要求的语法以粗体表示。

```
union
[<subsearch-options>]
<dataset>
[<dataset>...]
```

必要参数

数据集

语法：<dataset-type>:<dataset-name> | <subsearch>

描述：您想要执行 union 的数据集。数据集可以是已命名的数据集，也可以是未命名的数据集。

- 已命名的数据集由 <dataset-type>:<dataset-name> 组成。对于 <dataset-type>，您可以指定数据模型、保存的搜索或 inputlookup。例如，datamodel:"internal_server.splunkdaccess"。
- 子搜索是一个已命名的数据集。

指定多个数据集之后，可在数据集名称之间使用空格或逗号分隔符。

可选参数

subsearch-options

语法：maxtime=<int> maxout=<int> timeout=<int>

描述：您可以指定一组 subsearch-options，可用于所有子搜索。您可以指定一个或多个 subsearch-options。仅当子搜索被视为非流搜索时适用这些选项。

- maxtime 参数指定了完成之前运行子搜索的最大秒数。默认值为 60 秒。
- maxout 参数指定了返回子搜索的最大结果数。默认值为 50000 个结果。此值是 maxresultrows 设置（位于 limits.conf 文件内的 [searchresults] 段落中）。
- timeout 参数指定了缓存子搜索结果的最大时间数（以秒为单位）。默认值为 300 秒。

用法

union 命令属于生成命令。

union 命令如何处理数据集取决于数据集是流数据集还是非流数据集。数据集类型由用于新建数据集的命令决定。请参阅“命令类型”。

有两种类型的流命令，可分配流命令和中央流命令。在此次 union 命令的讨论中，流数据集指的是可分配流命令。

运行命令的地方

数据集是流数据集还是非流数据集决定了在索引器还是搜索头重运行 union 命令。下表指定了运行命令的地方。

数据集类型	数据集 1 是流数据集	数据集 1 是非流数据集
数据集 2 是流数据集	索引器	搜索头
数据集 2 是非流数据集	搜索头	搜索头

如何处理命令

这类数据集还可确定如何处理 union 命令。

数据集类型	处理影响
中央流或非流命令	处理为 append 命令。
可分配流	处理为 multisearch 命令。 将 <streaming_dataset1> 放在 union 命令后面更有效。

流式发送数据集的优化语法

流式发送数据集，与此语法相比：

```
<streaming_dataset1> | union <streaming_dataset2>
```

用此语法可提高搜索效率：

```
... | union <streaming_dataset1>, <streaming_dataset2>
```

为什么联合结果可能会被截断

考虑以下搜索，使用 union 命令合并三个索引中的事件。每个索引包含 60,000 个事件，共计 180,000 个事件。

```
| union maxout=10000000 [ search index=union_1 ] [ search index=union_2 ] [ search index=union_3 ] | stats count by index
```

此搜索产生以下联合事件：

index	count
union_1	60000
union_2	60000
union_3	60000

在此示例中，所有子搜索都是可分配的流命令，因此可使用和 `multisearch` 命令相同的方式处理，将这些命令联合起来。每个索引都有 60,000 个事件，联合起来共计 180,000 个合并事件。

但是，如果您在其中一个子搜索中指定中央流命令，如 `head` 命令，那么结果会更改。

```
| union maxout=10000000 [ search index=union_1 | head 60000 ] [ search index=union_2 ] [ search index=union_3 ] | stats count by index
```

此搜索将产生以下联合结果，共计 160,000 个合并事件。

index	count
union_1	60000
union_2	50000
union_3	50000

因为 `head` 命令是中央流命令而不是可分配的流命令，因此使用 `append` 命令处理任何遵循 `head` 命令的子搜索。换句话说，如果命令强制处理搜索头，则必须在搜索头上处理所有子搜索命令。

在内部，搜索转换为：

```
| search index=union_1 | head 60000 | append [ search index=union_2 ] | append [ search index=union_3 ] | stats count by index
```

将 `union` 命令和非流命令结合使用时，默认强制执行 `maxout` 参数。`maxout` 参数的默认值为 50,000 个事件。在此示例中，`maxout` 参数默认值为强制从使用非流命令的子搜索开始。对任何后续的子搜索强制执行默认值。

如果非流命令在最后的子搜索上，按照流命令的处理方式处理前两个子搜索。使用 `multisearch` 命令处理将这些子搜索联合起来。最后的子搜索包括非流命令 `head`。使用 `append` 命令处理将该子搜索联合起来。

在内部，搜索转换为：

```
| multisearch [ search index=union_1 ] [ search index=union_2 ] | append [ search index=union_3 | head 60000 ] | stats count by index
```

在此示例中，`maxout` 参数的默认值只适用于最后的子搜索。该子搜索仅返回 50,000 个事件，而不是所有 60,000 个事件。合并的事件总数是 170,000。第一个和第二个子搜索中的 60,000 个事件和最后一个搜索中的 50,000 个事件。

交错结果

如果是按照时间降序（此为默认排序顺序）从磁盘检索两个数据集，`union` 命令会交错结果。基于 `_time` 字段进行交错。例如，您有以下数据集：

dataset_A

_time	host	bytes
4	mailsrv1	2412
1	dns15	231

dataset_B

_time	host	bytes
3	router1	23

2	dns12	22o
---	-------	-----

两个数据集是按照 `_time` 降序排列的。以下数据集为运行 `| union dataset_A, dataset_B` 时的结果。

<code>_time</code>	<code>host</code>	<code>bytes</code>
4	mailsrv1	2412
3	router1	23
2	dns12	22o
1	dns15	231

示例

1. 联合两个子搜索的事件

以下示例将来自 `index a` 和 `index b` 的事件合并。使用 `eval` 命令将新字段 `type` 和 `mytype` 添加到各子搜索。

```
| union [search index=a | eval type = "foo"] [search index=b | eval mytype = "bar"]
```

2. 将子搜索的结果联合到主搜索的结果

以下示例将子搜索的错误结果表格附加到当前主要搜索的结果。

```
... | chart count by category1 | union [search error | chart count by category2]
```

3. 联合数据模型的事件和索引的事件

以下示例联合 REST API 调用内部服务器日志的内置数据模型和来自 `index a` 的事件。

```
... | union datamodel:"internal_server.splunkdaccess" [search index=a]
```

4. 指定子搜索选项

以下示例将返回子搜索的最大结果数设为 20,000。示例指定子搜索期限限制为 120 秒。示例还将子搜索结果缓存最大时间设为 600 秒（5 分钟）。

```
... | chart count by category1 | union maxout=20000 maxtime=120 timeout=600 [search error | chart count by category2]
```

另请参阅

相关信息

《[搜索手册](#)》中的“关于子搜索”。

请参阅[知识管理器手册](#)中的“数据模型”

命令

```
search  
inputlookup
```

uniq

描述

`uniq` 命令将对您传递给该命令的搜索结果进行过滤。此命令将与先前结果完全重复的所有搜索结果删除。此命令不含任何参数。

我们不建议对大型数据集运行此命令。

语法

```
uniq
```

示例

示例 1:

仅保留过去一小时内所有网络流量的唯一结果。

```
eventtype=webtraffic earliest=-1h@s | uniq
```

另请参阅

[dedup](#)

untab le

描述

将结果从表格格式转换为与 `stats` 输出类似的格式。此命令是 `xseries` 命令的反向命令。

语法

```
untab le <x-field> <y-name-field> <y-data-field>
```

必要参数

<x-field>

语法: <field>

描述: 要用作 X 轴的字段。

<y-name-field>

语法: <field>

描述: 包含要用作数据系列标签的值的字段。

<y-data-field>

语法: <field>

描述: 包含要制作作为图表的数据的字段。

用法

`untab le` 命令属于可分配的流命令。请参阅“[命令类型](#)”。

字段值重复的结果

当您使用 `untab le` 命令处理一组结果，然后使用 `xseries` 命令合并结果时，包含重复值的结果会被移除。

您可以使用 `streamstats` 命令创建唯一的记录编号，并使用这些编号保留所有结果。请参阅[延伸示例](#)。

基本示例

此示例使用搜索教程中的示例数据。要在您自己的 Splunk 实例中尝试此示例，您必须下载示例数据并遵循操作以将教程数据导入 Splunk。当运行搜索时，请使用时间范围所有时间。

要显示如何使用 `untab le` 命令，我们需要以表格形式显示的结果。运行此搜索。

```
sourcetype=access_* status=200 action=purchase | top categoryId
```

统计选项卡中显示的结果如下所示：

categoryId	计数	百分比
STRATEGY	806	30. 495649
ARCADE	493	18. 653046

TEE	367	13. 885736
ACCESSORIES	348	13. 166856
SIMULATION	246	9. 307605
SHOOTER	245	9. 269769
SPORTS	138	5. 221339

top 命令会自动将计数和百分比字段添加到结果中。

每个 categoryId 都有两个值，计数和百分比。当您使用 untable 命令处理这些结果时，输出中将有三列：

- 第一个列将列出类别 ID
- 第二列列出计算类型：计数或百分比
- 第三列列出各计算值

当您使用 untable 命令转换表格结果时，您必须先指定 categoryId 字段。您可以将任何您想要的字段名用于计算类型和值。例如：

```
sourcetype=access_* status=200 action=purchase | top categoryId | untable categoryId calculation value
```

统计选项卡中显示的结果如下所示：

categoryId	计算	值
STRATEGY	计数	806
STRATEGY	百分比	30. 495649
ARCADE	计数	493
ARCADE	百分比	18. 653046
TEE	计数	367
TEE	百分比	13. 885736
ACCESSORIES	计数	348
ACCESSORIES	百分比	13. 166856
SIMULATION	计数	246
SIMULATION	百分比	9. 307605

延伸示例

untable 命令正如名字那样，会将表格信息转换为单独的结果行。假设您有此搜索：

```
... | table _time EventCode Message
```

搜索产生这些结果：

_time	EventCode	消息
date-time1	4136	现在太晚了
date_time2	1234	我不知道
date_time3	3456	太晚了，稍后再问
date_time4	1256	所有事情都突然发生了
date_time4	1257	以及现在

注意：这组事件在 _time 字段中有 date_time4 的重复值。稍后将返回此窗口。

使用 `untab` 命令删除表格格式。

```
...| untab _time FieldNameFieldValue
```

以下是 `untab` 命令的结果：

_time	FieldName	FieldValue
date-time1	EventCode	4136
date-time1	消息	现在太晚了
date_time2	EventCode	1234
date-time2	消息	我不知道
date_time3	EventCode	3456
date-time3	消息	太晚了，稍后再问
date_time4	EventCode	1256
date-time4	消息	所有事情都突然发生了
date_time4	EventCode	1257
date-time4	消息	以及现在

时间戳重复的事件

记住：本示例中的原始事件集有重复的 `date_time4`。如果您想要以某种方式处理事件，然后将这些事件放在一起，您可以使用 `streamstats` 命令避免消除重复的事件。

使用 `streamstats` 命令为各事件指定唯一的记录号，然后将该唯一的编号用作 `untab` 和 `xseries` 命令的关键字段。

例如，您可以将 `streamstats` 命令添加到原始搜索中。

```
...| table _time EventCode Message | streamstats count as recno
```

搜索产生这些结果：

_time	EventCode	消息	recno
date-time1	4136	现在太晚了	1
date_time2	1234	我不知道	2
date_time3	3456	太晚了，稍后再问	3
date_time4	1256	所有事情都突然发生了	4
date_time4	1257	以及现在	5

然后可以使用 `recno` 作为 `<x-field>` 将 `untab` 命令添加到搜索中：

```
...| table _time EventCode Message | streamstats count as recno | untab recno FieldNameFieldValue
```

搜索产生这些结果：

recno	FieldName	FieldValue
1	EventCode	4136
1	消息	现在太晚了
2	EventCode	1234
2	消息	我不知道

3	EventCode	3456
3	消息	太晚了，稍后再问
4	EventCode	1256
4	消息	所有事情都突然发生了
4	EventCode	1257
4	消息	以及现在

可以使用 `xyseries` 命令重新将这些事件放在一起，当然也要再次使用 `recno` 字段作为 `<x-field>`。例如：

```
...| xyseries recno FieldName FieldValue
```

搜索产生这些结果：

recno	EventCode	消息
1	4136	现在太晚了
2	1234	我不知道
3	3456	太晚了，稍后再问
4	1256	所有事情都突然发生了
5	1257	以及现在

恢复时间戳

除了使用 `streamstats` 命令生成记录编号外，还可以使用 `rename` 命令在 `xyseries` 命令之后还原时间戳信息。例如：

```
...| table _time EventCode Message | streamstats count as recno | rename _time as time | untable recno FieldName FieldValue | xyseries recno FieldName FieldValue | rename time as _time
```

（感谢 Splunk 用户 DalJeanis 和 BigCosta 在此示例中提供的帮助。）

另请参阅

`xyseries`

`walklex`

描述

从每个事件索引数据桶中生成术语或索引字段的列表。仅适用于合并了 `lexicon` 文件或单个 `.tsidx` 文件的温数据桶。

由于 `merged_lexicon.lex` 和 `.tsidx` 文件的可变性，`walklex` 并不会一直返回一致的结果。

语法

要求的语法以粗体表示。

```
| walklex
[ type=<walklex-type> ]
[ prefix=<string> | pattern=<wc-string> ]
<index-list>
[ splunk_server=<wc-string> ]
[ splunk_server_group=<wc-string> ]...
```

必要参数

`<index-list>`

语法： `index=<index-name> index=<index-name> ...`

描述： 把搜索限制到一个或多个索引。例如，`index=_internal`。

可选参数

前缀 | 模式

语法: prefix=<string> | pattern=<wc-string>

描述: 将结果限制为和特定模式或前缀匹配的术语。前缀或模式可以指定其中一个，但不能同时指定。只包含带有 merged_lexicon 文件或单个 tsidx 文件的数据桶。这表示通常不包括热数据桶。

默认值: pattern=*

splunk_server

语法: splunk_server=<wc-string>

描述: 指定返回结果的分布式搜索节点。

- 如果您使用 Splunk Cloud，请忽略此参数。
- 如果您使用 Splunk Enterprise，您仅可指定一个 splunk_server 参数。但在指定服务器名称时可以使用通配符，以指示多个服务器。例如，您可以指定 splunk_server=peer01 或 splunk_server=peer*. 用 local 表示搜索头。

默认值: 所有配置的搜索节点都会返回信息

splunk_server_group

语法: splunk_server_group=<wc-string>

描述: 把结果限制到一个或多个服务器组。可以在字符串中指定一个通配符字符，来指示名称类似的多个服务器组。如果您使用 Splunk Cloud，请忽略此参数。

默认值: 无

type

语法: type = (all | field | fieldvalue | term)

描述: 指定要在字典中返回的术语类型。有关使用 type 参数选项的更多信息，请参阅[用法](#)了解更多信息。

- 使用 field 只返回每个索引数据桶中唯一的字段名。
- 使用 fieldvalue 只包含索引字段术语。
- 使用 term 排除所有格式为 <field>::<value> 的索引字段术语。

默认: 所有

用法

walklex 命令属于生成命令，使用前导管道字符。walklex 命令必须是搜索管道中的首个命令。请参阅“[命令类型](#)”。

当 Splunk 软件索引事件数据时，它使用在 segmenters.conf 文件中指定的规则将每个事件分割为原始标记。您可能会得到原始标记，这些标记实际上是由任意分隔符（如等号 (=)）分隔的键-值对。

以下搜索使用 walklex 和 where 命令在索引中查找原始标记。它使用 stats 命令计数原始标记。

```
| walklex index=<target-index> | where NOT like(term, "%::%") | stats sum(count) by term
```

仅返回已建立索引的字段名称

指定 type=field 参数以使 walklex 只返回索引字段的字段名称。

walklex 返回的索引字段可以包括默认字段，例如 host、source、sourcetype、date_* 字段、punct 等。它也可以包括其他索引字段，这些字段在 props.conf 和 transforms.conf 中配置，并使用 INDEXED_EXTRACTIONS 设置或其他 WRITE_META 方法创建。最后一组额外索引字段的发现可能会帮助您加快搜索速度。

返回表示带索引值的索引字段的一组术语

指定 type=fieldvalue 参数，以使 walklex 从索引返回一组术语，这些术语即为带索引值的索引字段。

type=fieldvalue 参数从索引中返回一组术语，这些术语即为带索引值的索引字段。与 type=field 参数不同（其返回的值只是字段名称本身），type=fieldvalue 参数会返回带任意字段值的索引字段名称。

例如，如果索引字段术语是 runtime::0.04，则 type=fieldvalue 参数返回的值是 runtime::0.04。而 type=field 参数返回的值是 runtime。

返回不属于索引字段结构的所有 TSIDX 关键字

指定 type=term 参数以使 walklex 从 TSIDX 文件返回不属于任何索引字段结构的关键字。换句话说，它会排除所有格式为 <field>::<value> 的所有索引字段术语。

返回所有三种类型的术语

如果没有指定类型或指定 `type=all`，则 `walklex` 会使用默认的 `type=all` 参数。这将导致 `walklex` 返回所有三种类型的索引中的术语：`field`、`fieldvalue` 和 `term`。

当使用 `type=all` 时，索引字段不会像使用 `type=field` 参数时那样被显式调出。您需要在 `::` 处拆分术语字段，以便从索引术语中获取字段值。

限制

`walklex` 命令只适用于事件索引。它不能与指标索引一起使用。

将搜索过滤器应用于一个或多个角色的人不能使用 `walklex`，除非他们也具有一个角色，该角色拥有 `run_walklex capability` 或 `admin_all_objects capability` 功能。有关基于角色的搜索过滤器的更多信息，请参阅《确保 Splunk 平台安全》中的“通过 Splunk Web 创建和管理角色”。有关基于角色的功能的更多信息，请参阅《确保 Splunk 平台安全》中的“使用功能定义 Splunk 平台上的角色”。

基本示例

1. 返回特定数据桶中每个术语的总计数

以下示例将返回 `_internal` 索引的每个数据桶中的所有术语并查找各术语的总计数。

```
| walklex index=_internal | stats sum(count) BY term
```

2. 指定多个索引

以下示例将返回 `_internal` 和 `_audit` 索引的各数据桶以 `foo` 开头的所有术语。

```
| walklex prefix=foo index=_internal index=_audit
```

3. 使用模式查找索引的字段术语

以下示例将返回 `_internal` 索引中以 `bar` 结尾的各数据桶的所有已索引字段术语。

```
| walklex pattern=*bar type=fieldvalue index=_internal
```

4. 返回索引字段的所有字段名称

以下示例将返回 `_audit` 索引的各数据桶中已索引字段的所有字段名称。

```
| walklex type=field index=_audit
```

另请参阅

命令

```
metadata  
tstats
```

where

描述

`where` 命令使用 `eval` 表达式筛选搜索结果。这些 `eval` 表达式必须是布尔值表达式，返回 `true` 或 `false`。`where` 命令仅返回 `eval` 表达式返回 `true` 的结果。

语法

```
where <eval-expression>
```

必要参数

Eval-expression

语法：`<eval-mathematical-expression> | <eval-concatenate-expression> | <eval-comparison-expression> | <eval-boolean-expression> | <eval-function-call>`

描述：代表目标字段值的值、变量、运算符以及函数的组合。请参阅“用法”。

运行搜索前会先检查 eval 表达式的语法；若表达式无效，会引发异常。

下表描述了需要特殊处理的 eval 表达式的特性。

表达式特性	描述	示例
以数字字符开头的字段名称	若表达式引用了以数字字符开头的字段名称，需要用单引号将字段名称括起。	'5minutes'="late" 此表达式是一个字段名称，等于字符串值。因为字段是以一个数字开始的，因此必须用单引号括起来。因为值是一个字符串，因此必须用双引号括起来。
含非字母数字字符的字段名称	若表达式引用了包含非字母数字字符的字段名称，必须用单引号将字段名称括起。	new=count+'server-1' 此表达式可能会被解释成一个数学方程，其中破折号被解释为减号。为避免出现这种情况，您必须用单引号将字段名称 server-1 括起来。
文字字符串	如果表达式引用文字字符串，那么必须用双引号将文字字符串括起来。	new="server- "+count 本示例存在两个问题。首先，server- 可能被解释为字段名称或使用加减号的数学方程的一部分。要确保 server- 被解释为文字字符串，用双引号将字符串括起来。

用法

where 命令属于可分配的流命令。请参阅“命令类型”。

where 命令使用的表达式语法与 eval 命令相同。同样，两个命令都将带引号的字符串解释为文字。如果字符串不带引号，则将其视为字段名称。有鉴于此，您可以使用 where 命令来比较两个不同的字段，该操作无法用 search 命令完成。

命令	示例	描述
Where	... where foo=bar	此搜索查找字段 foo 等于字段 bar 的事件。
搜索	search foo=bar	此搜索查找字段 foo 包含字符串值 bar 的事件。
Where	... where foo="bar"	此搜索查找字段 foo 包含字符串值 bar 的事件。

布尔表达式

使用 where 命令评估的布尔表达式的顺序为：

1. 括号中的表达式
2. NOT 子句
3. AND 子句
4. OR 子句

此评估顺序和通过 search 命令使用的顺序不同。search 命令会先评估 OR 子句再评估 AND 子句。

在 where 命令中使用通配符

在 where 命令中只能使用 like 函数来指定通配符。百分号（%）是搭配 like 函数使用通配符。请参阅 like() 评估函数。

支持的函数

您可以将 `where` 命令与一系列评估函数结合使用。有关使用函数的一般信息，请参阅“评估函数”。

- 若需按类别排列的函数列表，请参阅“按类别排列的函数列表”。
- 若需按字母顺序排列的函数列表，请参阅“按字母顺序排列的函数列表”。

示例

1. 在 `where` 命令中指定通配符

只能使用 `like` 函数来为 `where` 命令指定通配符。百分号（%）是搭配 `like` 函数必须使用通配符。如果 `ipaddress` 字段以值 198. 开头，则 `where` 命令返回 `like=TRUE`。

```
... | where like(ipaddress, "198.%")
```

2. 使用 `where` 命令匹配 IP 地址或子网

返回与 IP 匹配或在指定子网中的 “CheckPoint” 事件。

```
host="CheckPoint" | where like(src, "10.9.165.%") OR cidrmatch("10.9.165.0/25", dst)
```

3. 在 `where` 命令表达式中指定计算

返回速度大于 100 的 “physicjobs” 事件。

```
sourcetype=physicsjobs | where distance/time > 100
```

另请参阅

`eval`, `search`, `regex`

x11

描述

`x11` 命令移除基于时间的数据系列的季节性模式，以便您可以看到数据中真实的趋势。此命令与 `trendline` 命令具有相同的用途，但它使用业内更流行、更成熟的 X11 方法。

时间系列数据中的季节性组件要么是加法，要么是乘法，定义为季节性的两种类别，可以用 `x11` 计算：`add()`（适用于加法）和 `mult()`（适用于乘法）。请参阅搜索手册中的“有关时间系列预测”。

语法

```
x11 [<type>] [<period>] (<fieldname>) [AS <newfield>]
```

必要参数

<fieldname>

语法：<field>

描述：用于计算季节趋势的字段名称。

可选参数

<type>

语法：`add()` | `mult()`

描述：指定要计算的 `x11` 的类型，加法还是乘法。

默认：`mult()`

<period>

语法：<int>

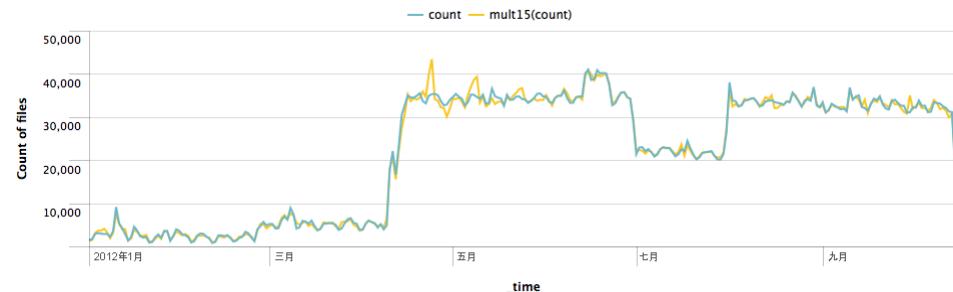
描述：相对于数据点的数据周期，以介于 5 到 1000 之间的整数表示。如果周期为 7，则该命令希望数据的周期为每 7 个数据点。如果省略此参数，Splunk 软件将自动计算周期。如果周期小于 5，该算法将不起作用；如果周期大于 10,000，则速度会非常慢。

```
<newfield>
    语法: <string>
    描述: 为 x11 命令的输出指定字段名称。
    默认值: 无
```

示例

示例 1: 本示例中, 类型是默认的 mult, 周期是 15。指定的字段名称是 count。

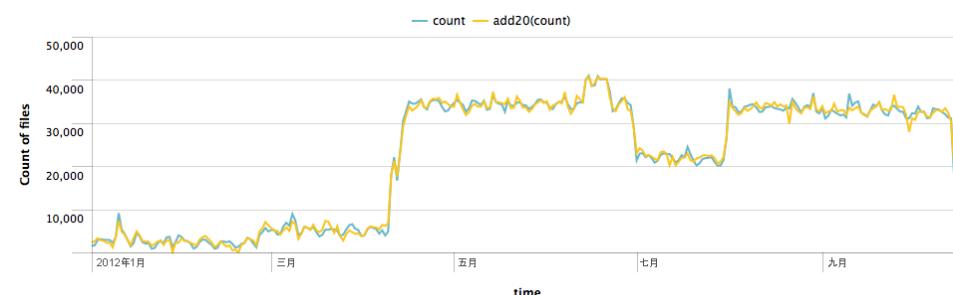
```
index=download | timechart span=1d count(file) as count | x11 mult15(count)
```



由于 span=1d, 每个数据点都以 1 天为基础。因此, 本示例中的周期为 15 天。您可以将本示例中的语法改为 ... | x11 15(count), 因此 mult 类别是默认类别。

示例 2: 在本示例中, 类别是 add, 周期是 20。指定的字段名称是 count。

```
index=download | timechart span=1d count(file) as count | x11 add20(count)
```



另请参阅

`predict`, `trendline`

xmlkv

描述

`xmlkv` 命令自动从 XML 格式的数据中提取键值对。

JSON 格式的数据请使用 `spath` 命令。

语法

要求的语法以**粗体**表示。

```
xmlkv
[<field>]
maxinputs=<int>
```

必要参数

无。

可选参数

字段

语法: <field>
描述: 要从中提取键和值对的字段。
默认值: _raw 字段。

maxinputs

语法: maxinputs=<int>
描述: 用作 xmlkv 命令输入的最大事件或搜索结果数。
默认值: 50000

用法

xmlkv 命令属于可分配的流命令。请参阅“命令类型”。

XML 元素中的键和值

在以下 XML 中，name 是键，而 Settlers of Catan 是第一个元素中的值。

```
<game>
  <name>Settlers of Catan</name>
  <category>competitive</category>
</game>
<game>
  <name>Ticket to Ride</name>
  <category>competitive</category>
</game>
```

示例

1. 自动提取键值对

从 _raw 字段中的 XML 标记中提取键值对。最多处理 50000 个事件。

```
... | xmlkv
```

2. 提取特定数量的键值对

从前一万个事件中提取键值对。

```
... | xmlkv maxinputs=10000
```

另请参阅

命令

```
extract
kvform
multikv
rex
spath
xpath
```

xmlunescape

描述

取消转义 xml 字符，包括 &、<、and > 等实体引用，使它们回归到对应的字符。例如，& 变为 &。

语法

```
xmlunescape maxinputs=<int>
```

必要参数

maxinputs
语法: maxinputs=<int>
描述: 输入的最大值。

示例

示例 1: 取消转义所有 XML 字符。

```
... | xmlunescape
```

xpath

描述

从 field 提取 xpath 值，并设置 outfield 属性。

语法

```
xpath [outfield=<field>] <xpath-string> [field=<field>] [default=<string>]
```

必要参数

xpath-string
语法: <string>
描述: 指定 XPath 引用。

可选参数

字段
语法: field=<field>
描述: 用于从中查找和提取引用的 xpath 值的字段。
默认值: _raw

outfield
语法: outfield=<field>
描述: 写入或输出 xpath 值的字段。
默认值: xpath

默认
语法: default=<string>
描述: 若在 xpath 中引用的属性不存在，该语法将指定要写入 outfield 的内容。如果未定义此参数，则没有默认值。

用法

xpath 命令属于可分配的流命令。请参阅“命令类型”。

xpath 命令支持 Python Standard Library 19.7.2.2 中介绍的语法。支持的 Xpath 语法。

示例

1. 从 _raw XML 事件中的单个元素中提取值

您想要从 _raw XML 事件中的单个元素中提取值并将这些值写入特定字段。

_raw XML 事件如下所示：

```
<foo>
  <bar nickname="spock">
    </bar>
  </foo>
  <foo>
    <bar nickname="scotty">
      </bar>
    </foo>
    <foo>
```

```
<bar nickname="bones">  
</bar>  
</foo>
```

从 `_raw` XML 事件中提取 `nickname` 值。将这些值输出到 `name` 字段。

```
sourcetype="xml" | xpath outfield=name "//bar/@nickname"
```

2. 从 `_raw` XML 事件中提取多个值

从 `_raw` XML 事件中提取多个值

`_raw` XML 事件如下所示：

```
<DataSet xmlns="">  
    <identity_id>3017669</identity_id>  
    <instrument_id>912383KM1</instrument_id>  
    <transaction_code>SEL</transaction_code>  
    <sname>BARC</sname>  
    <currency_code>USA</currency_code>  
</DataSet>  
  
<DataSet xmlns="">  
    <identity_id>1037669</identity_id>  
    <instrument_id>219383KM1</instrument_id>  
    <transaction_code>SEL</transaction_code>  
    <sname>TARC</sname>  
    <currency_code>USA</currency_code>  
</DataSet>
```

从 `_raw` XML 事件的 `identity_id` 元素中提取值：

```
... | xpath outfield=identity_id "//DataSet/identity_id"
```

此搜索将返回两个结果：`identity_id=3017669` 和 `identity_id=1037669`。

要提取两个元素的组合，带特定值和 `instrument_id` 的 `sname`，使用此搜索：

```
... | xpath outfield=instrument_id "//DataSet[sname='BARC']/instrument_id"
```

因为您指定 `sname='BARC'`，所以此搜索返回一个结果：`instrument_id=912383KM1`。

3. 测试 XML 事件的提取值

您可以使用 `makeresults` 命令测试 `xpath` 提取。

您必须将 `field=xml` 添加到搜索末端。例如：

```
| makeresults  
| eval xml=<DataSet xmlns="">  
    <identity_id>1037669</identity_id>  
    <instrument_id>219383KM1</instrument_id>  
    <transaction_code>SEL</transaction_code>  
    <sname>TARC</sname>  
    <currency_code>USA</currency_code>  
</DataSet>"  
| xpath outfield=identity_id "//DataSet/identity_id" field=xml
```

另请参阅

`extract`, `kvform`, `multikv`, `rex`, `spath`, `xmlkv`

xsDisplayConcept

`xsDisplayConcept` 命令是与 Splunk Enterprise Security 结合使用的极端搜索命令。

如需此命令的相关信息，请参阅管理 *Splunk Enterprise Security* 中的“极端搜索命令”。

xsDisplayContext

`xsDisplayContext` 命令是与 *Splunk Enterprise Security* 结合使用的极端搜索命令。

如需此命令的相关信息，请参阅管理 *Splunk Enterprise Security* 中的“极端搜索命令”。

xsFindBestConcept

`xsFindBestConcept` 命令是与 *Splunk Enterprise Security* 结合使用的极端搜索命令。

如需此命令的相关信息，请参阅管理 *Splunk Enterprise Security* 中的“极端搜索命令”。

xsListConcepts

`xsListConcepts` 命令是与 *Splunk Enterprise Security* 结合使用的极端搜索命令。

如需此命令的相关信息，请参阅管理 *Splunk Enterprise Security* 中的“极端搜索命令”。

xsListContexts

`xsListContexts` 命令是与 *Splunk Enterprise Security* 结合使用的极端搜索命令。

如需此命令的相关信息，请参阅管理 *Splunk Enterprise Security* 中的“极端搜索命令”。

xsUpdateDDContext

`xsUpdateDDContext` 命令是与 *Splunk Enterprise Security* 结合使用的极端搜索命令。

如需此命令的相关信息，请参阅管理 *Splunk Enterprise Security* 中的“极端搜索命令”。

xsWhere

`xsWhere` 命令是与 *Splunk Enterprise Security* 结合使用的极端搜索命令。

如需此命令的相关信息，请参阅管理 *Splunk Enterprise Security* 中的“极端搜索命令”。

xyseries

本主题介绍了如何使用 `xyseries` 命令。

描述

将结果转换为适用于绘图的表格格式。此命令是 `untabular` 命令的反向命令。

语法

```
xyseries [grouped=<bool>] <x-field> <y-name-field> <y-data-field>... [sep=<string>] [format=<string>]
```

必要参数

<x-field>

语法：<field>

描述：用于 x 轴标签的字段名称。该字段的值显示为在 x 轴上绘制的数据系列的标签。

<y-name-field>

语法：<field>

描述：包含要用作数据系列标签的值的字段。

<y-data-field>

语法：<field> [<field>] ...

描述：包含要绘制的数据的一个或多个字段。指定多个字段时，用逗号分隔各字段名称。

可选参数

format

语法: format=<string>

描述: 当多个数据系列与 split-by 字段结合使用, 并且要分隔 <y-name-field> 和 <y-data-field> 时, 用于构建输出字段名称。format 优先于 sep 并允许您使用 stats 聚合器、函数 (\$AGG\$) 及 split-by-field 的值 (\$VALUE\$) 指定参数化表达式。

grouped

语法: grouped= true | false

描述: 若为 true, 则指示将按 <x-field> 的值对输入进行排序, 且允许多文件输入。

默认值: false

sep

语法: sep=<string>

描述: 当多个数据系列由 split-by 字段进行分隔时, 用于构建输出字段名称。这相当于把 format 设置为 \$AGG\$<sep>\$VALUE\$。

用法

xyseries 命令是可分配的流命令, 除非指定 grouped=true, 然后 xyseries 命令是转换命令。请参阅“命令类型”。

别名

xyseries 命令的别名为 maketable。

字段值重复的结果

当您使用 xyseries 命令将结果转换为表格格式时, 包含重复值的结果会被删除。

您可以使用 streamstats 命令创建唯一的记录编号, 并使用这些编号保留所有结果。有关示例, 请参阅延伸示例了解 untable 命令。

示例

让我们来看一个示例, 通过这个示例了解如何使用 xyseries 命令重新格式化搜索结果。

编写搜索

此示例使用搜索教程中的示例数据, 但应该可以与任意格式的 Apache Web 访问日志配合使用。要在您自己的 Splunk 实例中尝试此示例, 您必须下载示例数据并遵循操作以将教程数据导入 Splunk。当运行搜索时, 请使用时间范围所有时间。

在搜索和报表应用中运行以下搜索:

```
sourcetype=access_* status=200 action=purchase | top categoryId
```

top 命令会自动将计数和百分比字段添加到结果中。每个 categoryId 都有两个值, 计数和百分比。

搜索结果如下所示:

categoryId	计数	百分比
STRATEGY	806	30. 495649
ARCADE	493	18. 653046
TEE	367	13. 885736
ACCESSORIES	348	13. 166856
SIMULATION	246	9. 307605
SHOOTER	245	9. 269769
SPORTS	138	5. 221339

在 `xyseries` 命令语法中标识您的字段

在本例中：

- <x-field> = categoryId
- <y-name-field> = count
- <y-data-field> = percent

使用 `xyseries` 重新格式化搜索结果

当应用 `xyseries` 命令时，`categoryId` 在搜索结果中用作 <x-field>。`count` 的计算结果成为搜索结果中的列 (<y-name-field>)。<y-data-field> (`percent`) 对应于搜索结果中的值。

在搜索和报表应用中运行以下搜索：

```
sourcetype=access_* status=200 action=purchase | top categoryId | xyseries categoryId count percent
```

搜索结果如下所示：

categoryId	138	245	246	348	367	493	806
SPORTS	5. 221339						
ACCESSORIES				13. 166856			
ARCADE						18. 653046	
SHOOTER		9. 269769					
SIMULATION				9. 307605			
STRATEGY							30. 495649
TEE						13. 885736	

延伸示例

让我们来看一个示例，通过这个示例了解如何为 `xyseries` 命令添加可选参数。

编写搜索

要为 `xyseries` 命令添加可选参数，您需要编写一个搜索，其中包括针对多个聚合的 `split-by-field` 命令。使用 `sep` 和 `format` 参数修改搜索结果中的输出字段名称。

在搜索和报表应用中运行以下搜索：

```
sourcetype=access_combined_wcookie | stats count(host) count(productId) by clientip, referer_domain
```

该搜索按 `clientip` 对 `referer_domain`、`count(host)` 和 `count(productId)` 进行排序。

clientip	referer_domain	count(host)	count(productId)
107.3.146.207	http://www.bing.com	3	1
107.3.146.207	http://www.buttermcupgames.com	358	259
107.3.146.207	http://www.google.com	10	4
107.3.146.207	http://www.yahoo.com	13	6
108.65.113.83	http://www.buttermcupgames.com	227	163
108.65.113.83	http://www.google.com	13	6

在搜索和报表应用中运行以下搜索：

```
sourcetype=access_combined_wcookie | stats count(host) count(productId) by clientip, referer_domain | xyseries clientip referer_domain count(host), count(productId)
```

在本例中：

- <x-field> = clientip
- <y-name-field> = referer domain
- <y-data-field> = host, productId

`xyseries` 命令需要两个聚合，在此示例中为：`count(host)` 和 `count(productId)`。前几个搜索结果如下所示：

clientip	count(host): http://www.bing.com	count(host): http://www.buttercupgames.com	count(host): http://www.google.com	count(host): http://www.yahoo.com	count(productId): http://www.bing.com
107.3.146.207	3	358	10	13	1
108.65.113.83		227	13	3	
109.169.32.135	2	400	13	7	1
110.138.30.229	1	159	10	5	1
110.159.208.78	2	241	9	4	0
111.161.27.20	2	192	9	4	2

添加可选参数：sep

在 `sep` 参数中添加一个字符串，以更改用于分隔 `<y-name-field> host` 和 `<y-data-field> productId` 的默认字符。`format` 参数添加 `<y-name-field>` 并用默认的 ":" 分隔字段名称和字段值

在搜索和报表应用中运行以下搜索：

```
sourcetype=access_combined_wcookie | stats count(host) count(productId) by clientip, referer_domain | xyseries clientip
referer_domain count(host), count(productId) sep="-"
```

前几个搜索结果如下所示：

clientip	count(host)- http://www.bing.com	count(host)- http://www.buttercupgames.com	count(host)- http://www.google.com	count(host)- http://www.yahoo.com	count(productId)- http://www.bing.com
107.3.146.207	3	358	10	13	1
108.65.113.83		227	13	3	
109.169.32.135	2	400	13	7	1
110.138.30.229	1	159	10	5	1
110.159.208.78	2	241	9	4	0
111.161.27.20	2	192	9	4	2

添加可选参数：format

`format` 参数添加 `<y-name-field>` 并用默认的 ":" 分隔字段名称和字段值例如，此示例的默认值为：
`count(host):referer_domain`

如果指定了一个字符串在使用 `format` 参数时分隔 `<y-name-field>` 和 `<y-data-field>`，它会覆盖 `sep` 参数的所有分配条件。在下面的示例中，`sep` 参数分配 "-" 字符用于分隔 `<y-name-field>` 和 `<y-data-field>` 字段。`format` 参数分配了一个 "+"，并且此分配的优先级高于 `sep`。在这种情况下，`VAL` 和 `AGG` 既代表 `<y-name-field>` 也代表 `<y-data-field>`。从搜索结果中可以看出，`<y-name-field>`, `host` 和 `<y-data-field>`, `productId` 可以对应于 `VAL` 或 `AGG`。

在搜索和报表应用中运行以下搜索：

```
sourcetype=access_combined_wcookie | stats count(host) count(productId) by clientip, referer_domain | xyseries clientip
referer_domain count(host), count(productId) sep="-" format="$AGG$ + $VAL$ TEST"
```

前几个搜索结果如下所示：

clientip ↴ ↵	count(host) + ↴ http://www.bing.com TEST ↴	count(host) + ↴ http://www.buttercupgames.com TEST ↴	count(host) + ↴ http://www.google.com TEST ↴	count(host) + ↴ http://www.yahoo.com TEST ↴	count(productId) ↴ + http://www.bing.com TEST ↴
107.3.146.207	3	358	10	13	1
108.65.113.83		227	13	3	
109.169.32.135	2	400	13	7	1
110.138.30.229	1	159	10	5	1
110.159.208.78	2	241	9	4	0
111.161.27.20	2	192	9	4	2

添加可选参数: *grouped*

`grouped` 参数决定 `xyseries` 命令是作为可分配的流命令还是作为转换命令运行。`xyseries` 命令在默认状态 `grouped=False` 下作为流命令运行。

另请参阅

命令
`untable`

第三方自定义命令

欢迎使用搜索参考。有关内置搜索命令的链接，请参见左侧导航面板。如果在列表中没有找到某一命令，该命令可能是第三方应用程序或加载项的一部分。有关应用程序和加载项提供的命令的信息，请参阅 `Splunkbase` 文档。

内部命令

关于内部命令

内部搜索命令是指一组旨在用于特定情况的命令，通常是在指令和 Splunk 支持的指导下使用。在将来的版本中可能会对删除这些命令或者更新并重新实现。

使用其中任何命令之前，咨询您的 Splunk 管理员或 Splunk 支持。

- collapse
- dump
- findkeywords
- mcatalog
- noop
- runshellscript
- sendalert

collapse

collapse 命令是不受支持的试验性内部命令。请参阅“关于内部命令”。

描述

collapse 命令将多文件结果精简为 chunkszie 选项所允许的文件数量。此命令在您使用 outputlookup 和 outputcsv 命令时自动运行。

语法

```
... | collapse [chunkszie=<num>] [force=<bool>]
```

可选参数

chunkszie

语法: chunkszie=<num>

描述: 限制生成的文件数。

默认值: 50000

force

语法: force=<bool>

描述: 如果 force=true 并且结果完全在内存中，则将结果重新划分到适合的组块文件中。

默认值: false

示例

示例 1：折叠结果。

```
... | collapse
```

dump

dump 命令是不受支持的试验性内部命令。请参阅“关于内部命令”。

描述

对于 Splunk Enterprise 部署，将搜索结果导出到本地磁盘上的一组文件块。有关其他导出方法的信息，请参阅《[搜索手册](#)》中的“导出搜索结果”。

语法

```
dump basefilename=<string> [rollsize=<number>] [compress=<number>] [format=<string>] [fields=<comma-delimited-string>]
```

必要参数

basefilename
语法: `basefilename=<string>`
描述: 导出文件名的前缀。

可选参数

compress
语法: `compress=<number>`
描述: gzip 压缩级别。指定 0 至 9 的数字，其中 0 意味着无压缩，数字越大意味着压缩比越高，写入速度越慢。
默认值: 2

字段

语法: `fields=<comma-delimited-string>`
描述: 要导出的字段列表。整个列表必须用引号引起来。忽略无效字段名称。

format
语法: `format= raw | csv | tsv | json | xml`
描述: 输出数据格式。
默认: raw

rollsize
语法: `rollsize=<number>`
描述: 最小文件大小 (MB)，此时无法将更多事件写入文件，且它将成为 HDFS 传输的候选。
默认值: 63 MB

用法

此命令将事件导出到本地磁盘上的一组数据块文件中，路径为 "\$SPLUNK_HOME/var/run/splunk/dispatch/<sid>/dump"。该命令可以识别输入事件中的特殊字段 _dstpath，若设置该字段，它会作为路径附加到 dst 目录，用于计算最终目标路径。

dump 命令将在接收事件的同时保留事件的排序。

示例

示例 1: 将所有事件从索引 "bigdata" 导出到本地磁盘上 "\$SPLUNK_HOME/var/run/splunk/dispatch/<sid>/dump/" 目录中的 "YYYYmmdd/HH/host" 位置，并将 "MyExport" 作为导出文件名的前缀。通过 eval 继续执行转储命令，可实现导出数据分区。

```
index=bigdata | eval _dstpath=strftime(_time, "%Y%m%d%H") + "/" + host | dump basefilename=MyExport
```

示例 2: 将所有事件从索引 "bigdata" 导出到本地磁盘上，并将 "MyExport" 作为导出文件名的前缀。

```
index=bigdata | dump basefilename=MyExport
```

findkeywords

findkeywords 命令是不受支持的试验性内部命令。请参阅“关于内部命令”。

描述

给定标记事件分组的某个整数，找到搜索来生成这些分组。

语法

```
findkeywords labelfield=<field>
```

必要参数

labelfield
语法: `labelfield=<field>`
描述: 字段名称。

用法

使用 findkeywords 命令（用于 cluster 命令之后），或一个类似的事件分组命令。findkeyword 命令获取含某个字段 (labelfield) 的结果集，该字段为分组集提供结果分类。此命令派生一个搜索来生成每个分组。此搜索可以保存为事件类型。

示例

返回特定 `log_level` 值的日志并对结果进行分组

返回所有日志，其中 `log_level` 是 DEBUG、WARN、ERROR、FATAL，并按群集计数对结果进行分组。

```
index=_internal source=*splunkd.log* log_level!=info | cluster showcount=t | findkeywords labelfield=cluster_count
```

结果是一张统计表：

The screenshot shows the Splunk Search & Reporting interface. At the top, there's a search bar with the query: `index=_internal source=*splunkd.log* log_level!=info | cluster showcount=t | findkeywords labelfield=cluster_count`. Below the search bar, it says "83 个事件 (15/02/03 7:49:33.000 之前)". The interface includes tabs for "事件" (Events), "模式" (Mode), "统计信息 (21)" (Statistics), and "可视化" (Visualize). The "统计信息 (21)" tab is selected. The table below has the following columns: confidence, eventTypeable, excludeKeywords, groupID, includeKeywords, numInInputGroup, numMatched, percentInInputGroup, percentMatched, sampleEvent. The data rows are as follows:

confidence	eventTypeable	excludeKeywords	groupID	includeKeywords	numInInputGroup	numMatched	percentInInputGroup	percentMatched	sampleEvent
0.000000	1		1		36	83	0.433735	1.000000	02-03-2015 07:47:46.328 +0000 ERRC snippet=true&snippetEmbedJS=false sent a 0 byte response after earlier cli
0.000000	1	ja		2	11	73	0.132530	0.879518	02-03-2015 07:44:14.230 +0000 WAR
0.000000	1		6		6	0	0.072289	0.000000	02-03-2015 07:44:52.810 +0000 ERRC snippet=true&snippetEmbedJS=false Length of 5188!
0.000000	1		11		4	0	0.048193	0.000000	02-03-2015 07:08:06.527 +0000 ERRC
0.000000	1		4		4	0	0.048193	0.000000	02-01-2015 04:03:25.980 +0000 WAR
0.000000	1		10		3	0	0.036145	0.000000	02-02-2015 09:26:03.151 +0000 ERRC response after earlier claiming a Cont
0.000000	1		5		3	0	0.036145	0.000000	01-27-2015 09:18:58.204 +0000 ERRC snippet=true&snippetEmbedJS=false Length of 1759!

groupID 的值是 `cluster_count` 的值（由 `cluster` 命令返回）。

另请参阅

`cluster`、`findtypes`

makejson

`makejson` 命令是不受支持的试验性内部命令。请参阅“关于内部命令”。

描述

根据搜索结果中的指定字段集创建 JSON 对象，并将 JSON 对象放入新字段中。

语法

```
makejson <wc-field-list> output=<string>
```

必要参数

`output`

语法： `output=<string>`

描述：用于放置 JSON 对象的输出字段的名称。

可选参数

`wc-field-list`

语法： `<field>(<field>) ...`

描述：以逗号分隔的字段列表，用于生成 JSON 对象。可以在字段名称中使用通配符字符。

默认值：如果未指定列表，则 JSON 对象会包含所有字段。

用法

您不能使用 `table` 或 `fields` 命令在创建的 JSON 对象中指定字段顺序。

示例

1. 使用所有可用字段创建 JSON 对象

以下搜索会在名为 "data" 的字段中创建一个 JSON 对象，并从所有可用字段中获取值。

```
| makeresults count=5 | eval owner="vladimir", error=random()%3 | makejson output=data
```

- `makeresults` 命令会创建五个包含时间戳的搜索结果。
- `eval` 命令会在每个搜索结果中创建两个字段。一个字段名为 `owner`，并包含值 `vladimir`。另一个字段名为 `error`。该字段取一个随机数，然后使用模数学运算符（%）用此随机数除以 3。
- `makejson` 命令根据每个搜索结果中的字段的值创建一个 JSON 对象。

结果形式如下所示：

_time	owner	错误	data
2020/3/10 21:45:14	vladimir	1	{"owner": "vladimir", "error": 1, "_time": 1583901914}
2020/3/10 21:45:14	vladimir	0	{"owner": "vladimir", "error": 0, "_time": 1583901914}
2020/3/10 21:45:14	vladimir	0	{"owner": "vladimir", "error": 0, "_time": 1583901914}
2020/3/10 21:45:14	vladimir	2	{"owner": "vladimir", "error": 2, "_time": 1583901914}
2020/3/10 21:45:14	vladimir	1	{"owner": "vladimir", "error": 1, "_time": 1583901914}

2. 通过一组特定字段创建 JSON 对象

考虑以下数据：

_time	owner	error_code
2020-03-10 21:45:14	claudia	1
2020-03-10 20:45:17	alex	4
2020-03-10 06:48:11	wei	2
2020-03-09 21:15:35	david	3
2020-03-09 16:22:10	maria	4
2020-03-08 23:32:56	vanya	1
2020-03-07 14:05:14	claudia	2

`makejson` 命令用于仅使用 `_time` 和 `owner` 字段中的值在名为 "data" 的字段中创建 JSON 对象。`error` 字段不包含在 JSON 对象中。

```
| makeresults count=7 | eval owner="claudia", error=random()%5 | makejson _time, owner output=data
```

结果形式如下所示：

data
{"owner": "claudia", "_time": 1583876714}
{"owner": "alex", "_time": 1583873117}

{"owner": "wei", "_time": 1583822891}
{"owner": "david", "_time": 1583788535}
{"owner": "maria", "_time": 1583770930}
{"owner": "vanya", "_time": 1583710376}
{"owner": "claudia", "_time": 1583589914}

3. 使用字段通配符列表创建 JSON 对象

使用 _time 字段和以 _owner 结尾的字段中的值，在名为 "json-object" 的字段中创建 JSON 对象。

```
| makeresults count=5 | eval product_owner="wei", system_owner="vanya", error=random()%5 | makejson _time, *_owner output="json-object"
```

结果形式如下所示：

_time	product_owner	system_owner	错误	json-object
2020/3/10 22:23:24	wei	vanya	3	{"product_owner": "wei", "system_owner": "vanya", "_time": 1583904204}
2020/3/10 22:23:24	wei	vanya	2	{"product_owner": "wei", "system_owner": "vanya", "_time": 1583904204}
2020/3/10 22:23:24	wei	vanya	1	{"product_owner": "wei", "system_owner": "vanya", "_time": 1583904204}
2020/3/10 22:23:24	wei	vanya	3	{"product_owner": "wei", "system_owner": "vanya", "_time": 1583904204}
2020/3/10 22:23:24	wei	vanya	2	{"product_owner": "wei", "system_owner": "vanya", "_time": 1583904204}

4. 与绑定架构的查找一起使用

您可以将 makejson 命令与绑定架构的查找一起使用，以便将 JSON 对象存储在 description 字段中，以供以后处理。

假设 Splunk 应用程序随附一个名为 example_ioc_indicators 的 KVStore 集合，其中包含字段 key 和 description。为确保长期的可支持性，您不想修改该集合，而只是想在框架（例如 Splunk Enterprise Security (ES) 威胁框架）中使用自定义查找。

让我们先来看搜索的第一部分：

```
| makeresults count=1 | eval threat="maliciousdomain.example", threat_expiry="2020-01-01 21:13:37 UTC", threat_name="Sample threat", threat_campaign="Sample threat", threat_confidence="100" | makejson threat_expiry, threat_name, threat_campaign, threat_confidence output=description | table threat, description
```

该搜索产生的结果如下所示：

threat	description
maliciousdomain.example	{"threat_name": "Sample threat", "threat_confidence": 100, "threat_expiry": "2020-01-01 21:13:37 UTC", "threat_campaign": "Sample threat"}

然后，您添加 outputlookup 命令以将搜索结果发送到查找：

```
... | outputlookup append=t example_ioc_indicators
```

要在框架中使用这个自定义查找，您要在搜索中指定以下内容：

```
... | lookup example_ioc_indicators OUTPUT description AS match_context | spath input=match_context
```

另请参阅

相关命令

spath

mcatalog

mcatalog 命令是不受支持的试验性内部命令。请参阅“关于内部命令”。

描述

mcatalog 命令实施 metric_name 中的值和指标索引中的 dimension 字段的聚合。

语法

```
| mcatalog [prestats=<bool>] [append=<bool>] ( <values("<field> ")> [AS <field>] )  
[WHERE <logical-expression>] [ (BY|GROUPBY) <field-list> ]
```

必要参数

值 (<field>)

语句: values(<field>) [AS <field>]

描述: 以多值条目的形式返回指定字段所有唯一值的列表。各值按字典顺序排列。请参阅“用法”。

可选参数

append

语法: append=<bool>

描述: 只有当 prestats=true 时有效。此参数运行 mcatalog 命令并添加结果到一组现有的结果而不是生成新的结果。

默认值: false

<field-list>

语法: <field>, ...

描述: 指定用于对结果进行分组的一个或多个字段。

<logical-expression>

语法: <time-opts>|<search-modifier>|((NOT)? <logical-expression>)|<index-expression>|<comparison-expression>|(<logical-expression> (OR)? <logical-expression>)

描述: 包括时间和搜索调节器, 比较和索引表达式。不支持 CASE 或 TERM 指令。也无法使用 WHERE 子句搜索术语或短语。

prestats

语法: prestats=true | false

描述: 指定是否使用 prestats 格式。prestats 格式是 Splunk 内部格式, 设计用于可以生成聚合计算的命令使用。使用 prestats 格式时, 您可以将数据导成图表、数据或时间表命令, 这些可以接受 prestats 格式。prestats=true 时, AS 说明不重要。聚合的字段名称由使用 prestats 格式的命令确定, 产生聚合输出。

默认值: false

逻辑表达式选项

<comparison-expression>

语法: <field><comparison-operator><value> | <field> IN (<value-list>)

描述: 将字段和文本值比较或提供可以出现在字段中的值的列表。

<index-expression>

语法: "<string>" | <term> | <search-modifier>

描述: 使用文本字符串和搜索修饰符描述要从索引中检索的事件。

<time-opts>

语法: [<timeformat>] (<time-modifier>)*

描述: 描述搜索的起始时间和结束时间条件的格式

比较表达式选项

<comparison-operator>

语法: = | != | < | <= | > | >=

描述: 您可以在搜索字段/值对时使用比较运算符。含 equal (=) 或 not equal (!=) 运算符的比较表达式比较字符串值。例如, "1" 与 "1.0" 不匹配。含大于或小于运算符 <><= >= 的比较表达式按数字顺序比较两个数字, 并按字典顺序比较其他值。请参阅“用法”。

```
<field>
    语法: <string>
    描述: 字段的名称。

<value>
    语法: <literal-value>
    描述: 在比较表达式中, 字段的文本数字或字符串值。

<value-list>
    语法: (<literal-value>, <literal-value>, ... )
    描述: 和 IN 运算符结合使用, 以指定两个或两个以上值。例如, 用 error IN (400, 402, 404, 406), 而不用 error=400 OR
error=402 OR error=404 OR error=406
```

索引表达式选项

```
<string>
    语法: "<string>"
    描述: 指定要匹配的关键字或带引号的短语。搜索字符串和带引号的字符串(任何非搜索修饰符的字符串)时, Splunk
软件会搜索 _raw 字段查找匹配事件或结果。

<search-modifier>
    语法: <sourcetype-specifier> | <host-specifier> | <source-specifier> | <splunk_server-specifier>
    描述: 从指定字段中搜索事件。例如, 搜索一个主机或主机组合、来源和来源类型。请参阅知识管理器手册中的用默认字
段搜索。

<sourcetype-specifier>
    语法: sourcetype=<string>
    描述: 从指定的来源类型字段搜索事件。

<host-specifier>
    语法: host=<string>
    描述: 从指定的 host 字段搜索事件。

<source-specifier>
    语法: source=<string>
    描述: 从指定的 source 字段搜索事件。

<splunk_server-specifier>
    语法: splunk_server=<string>
    描述: 从特定的服务器搜索事件。使用 "local" 代表搜索头。
```

时间选项

有关时间调节器的列表, 请参阅“用于搜索的时间调节器”。

```
<timeformat>
    语法: timeformat=<string>
    描述: 设置用于 starttime 和 endtime 条件的时间格式。
    默认值: timeformat=%m/%d/%Y:%H:%M:%S。

<time-modifier>
    语法: starttime=<string> | endtime=<string> | earliest=<time_modifier> | latest=<time_modifier>
    描述: 指定开始时间和结束时间(使用相对或绝对时间)。
```

注意: 您还可以使用 earliest 和 latest 属性来指定搜索的绝对和相对时间范围。有关该时间调节器语法的更多信息, 请参阅《搜索手册》中的“关于搜索时间范围”。

```
starttime
    语法: starttime=<string>
    描述: 事件必须晚于或等于该时间。必须匹配 timeformat。

endtime
    语法: endtime=<string>
    描述: 所有事件都必须早于或等于该时间。
```

用法

您可以使用 mcatalog 命令来搜索指标数据。指标数据为指标字段使用特定的格式。请参阅指标中的“指标数据格式”。使用此

命令时，不允许使用 `_values` 字段。

`mcatalog` 命令属于报表的生成命令。生成命令使用前导管道字符。`mcatalog` 命令必须为搜索管道中的首个命令，除非 `append=true`。

所有指标搜索命令均区分大小写。举例来说，`mcatalog` 会把以下值视为 `metric_name` 的三个不同值：`cap.gear`、`CAP.GEAR` 和 `Cap.Gear`。

如果您的角色没有 `list_metrics_catalog` 功能，您无法使用 `mcatalog`。

请参阅《确保 Splunk Enterprise 安全》中的“关于定义带功能的角色”。

WHERE

使用 `WHERE` 子句接受支持的维度筛选。

如果您未在 `WHERE` 子句中指定索引名称，`mcatalog` 命令返回与您的角色相关的默认指标索引结果。如果您不指定索引名称，则没有默认指标索引与您的角色相关，`mcatalog` 不返回结果。要搜索所有指标索引，请使用 `WHERE index=*`。

有关为角色定义默认指标索引的更多信息，请参阅确保 *Splunk Enterprise* 安全中的“使用 Splunk Web 添加和编辑角色”。

分组方式

您可以按照 `dimension` 和 `metric_name` 字段进行分组。

`mcatalog` 命令不允许按时间范围分组。该参数不包含在其语法中。

时间维度

`mcatalog` 命令无法识别以下时间相关的维度。

不受支持的维度		
<code>date_hour</code> <code>date_mday</code> <code>date_minute</code> <code>date_month</code> <code>date_second</code>	<code>date_wday</code> <code>date_year</code> <code>date_zone</code> <code>metric_timestamp</code> <code>time</code>	<code>timeendpos</code> <code>timestamp</code> <code>timestartpos</code>

按字典顺序

基于用于编码计算机内存中的项目的值按字典顺序对这些项目进行排序。在 Splunk 软件中，几乎都是使用 UTF-8 进行编码，这是 ASCII 的超集。

- 数字排在字母前面。根据第一位数字对数字进行排序。例如数字 10、9、70、100，按照字典顺序排序为 10、100、70、9。
- 大写字母排在小写字母前面。
- 符号的排序标准不固定。有些符号排在数字值前面。有些符号排在字母前面或后面。

您可以指定覆盖字典顺序的自定义排序顺序。请参阅博客“向上排序”！自定义排序顺序

示例

1. 返回特定指标索引中的所有指标名称

返回 `new-metric-idx` 中的所有指标名称。

```
| mcatalog values(metric_name) WHERE index=new-metric-idx
```

2. 返回与用户角色相关的默认指标索引中的所有指标名称

如果用户角色没有分配得到默认指标索引，搜索不返回事件。

```
| mcatalog values(metric_name)
```

3. 返回所有指标索引中特定指标名称的所有 IP 地址

返回 `login.failure` 指标名称的 IP 地址。

```
| mcatalog values(ip) WHERE index=* AND metric_name=login.failure
```

4. 返回与用户角色相关的默认指标索引中的所有可用维度列表

```
| mcatalog values(_dims)
```

noop

`noop` 命令是不受支持的试验性内部命令。请参阅“关于内部命令”。

描述

`noop` 命令是可以用于调试搜索的内部命令。它包括可用于解决搜索优化问题的几个参数。

您可以使用 `noop` 命令向搜索添加注释。若想将注释添加到您的搜索中，请参阅《搜索手册》中“添加注释到搜索”。

语法

```
noop [<log-level-expression>] [<appender-expression>] [set_ttl = <timespan>] [search_optimization = <boolean>]  
[search_optimization.<optimization_type> = <boolean>] [sample_ratio = <int>]...
```

必要参数

无。

可选参数

appender-expression

语法: `log_appender = "<appender_name> [<attribute_name> = <attribute_value>], ..."`

描述 识别来自 `log-searchprocess.log` 的附加器，并指定属于该附加器的一个或多个属性的更改值。这些值更改适用于任务生存期内的搜索任务。搜索完成后不会重新使用。属性值更改列表应该用引号括起来。请参见“附加器表达式选项”。

log-level-expression

语法: `log_<level> = "<channel>, ..."`

描述 在搜索启动时设置或更改一个或多个日志通道的日志级别。日志通道列表应用双引号括起来。请参见“日志级别表达式选项”。

optimization_type

语法: `search_optimization.<optimization_type> = <boolean>`

描述 为搜索启用或禁用特定类型的搜索优化。要禁用多个优化类型，请新建一个以逗号隔开的 `search_optimization.<optimization_type>` 参数列表。请参阅“优化类型参数”。

默认值: `true`

sample_ratio

语法: `sample_ratio = <int>`

描述 设置随机采样的结果子集以从给定搜索返回。它会从每 `<sample_ratio>` 个事件中返回 1 个。例如，如果您提供 `noop sample_ratio=25`，则 Splunk 软件会从搜索结果集的每 25 个事件中返回 1 个随机样本。`sample_ratio` 参数要求 `search` 是您要对其应用 `noop` 的搜索的生成命令。

`sample_ratio` 与您可通过 Splunk Web 管理的事件采样功能提供相同的作用。区别在于，它可以将事件采样应用于子搜索，而事件采样的 Splunk Web 版本只能应用于主搜索。请参阅《搜索手册》中的“事件采样”。

默认值: 1

search_optimization

语法: `search_optimization = <boolean>`

描述 启用或禁用搜索的所有优化。

默认值: `true`

set_ttl

语法: `set_ttl = <timespan>`

描述 使用时间调节器（如表示一天的 `1d` 或者表示十二小时的 `12h`）指定搜索任务的生存期。搜索任务生存期是该任务在被删除之前存在于系统中的时间。即席搜索的默认生存期是 10 分钟。您可以使用此设置使即席搜索任务保留在系统中 24 小时或 7 天。

优化类型参数

以下是可以与 `noop` 结合使用的 `search_optimization.<optimization_type>` 参数。

search_optimization 参数	控制
<code>search_optimization.eval_merge</code>	Eval 合并优化
<code>search_optimization.merge_union</code>	合并联合优化
<code>search_optimization.predicate_merge</code>	断言合并优化
<code>search_optimization.predicate_push</code>	断言下推优化
<code>search_optimization.predicate_split</code>	断言拆分优化
<code>search_optimization.projection_elimination</code>	投影清除优化
<code>search_optimization.required_field_values</code>	必填字段值优化
<code>search_optimization.replace_append_with_union</code>	将 union 命令优化替换 append 命令
<code>search_optimization.replace_stats_cmds_with_tstats</code>	用 tstats 命令优化替换 stats 命令 默认情况下会禁用这种优化类型。
<code>search_optimization.search_flip_normalization</code>	断言翻转标准化
<code>search_optimization.search_sort_normalization</code>	断言排序标准化

有关特定搜索优化类型的更多信息，请参阅“[内置优化](#)”。

日志级别表达式选项

级别

语法: `log_<level>`

描述: 有效值为 Splunk 平台内部日志级别: `debug`、`info`、`warn`、`error` 和 `fatal`。您可以将不同的日志级别应用于不同的通道集。

通道

语法: `<channel>, ...`

描述: 指定一个或多个要应用日志级别的通道。使用通配符捕获名称中具有匹配字符串的所有通道。

附加器表达式选项

appender_name

语法: `<string>`

描述: 附加器名称来自 `log-searchprocess.cfg` 文件。使用通配符 * 识别 `log-searchprocess.cfg` 文件中的所有附加器。`noop` 分析器是区分大小写的。如果提交的附加器名称大小写格式不正确，会发送错误消息。

attribute_name

语法: `maxFileSize | maxBackupIndex | ConversionPattern | maxMessageSize`

描述: 可以为指定的附加器更改的属性。`noop` 分析器是区分大小写的，因此，请勿更改这些属性的大小写。如果提交的附加器名称大小写格式不正确，会发送错误消息。

属性名称	描述	示例值
<code>maxFileSize</code>	在文件汇总之前设置 <code>search.log</code> 文件的最大大小（单位是字节）。您必须提供一个比目前为 <code>log-searchprocess.cfg</code> 文件中所选的附加器设置的值更高的值。	250000000
<code>maxBackupIndex</code>	设置汇总的 <code>search.log</code> 文件的最大数量。您必须提供一个比目前为 <code>log-searchprocess.cfg</code> 文件中所选的附加器设置的值更高的值。	5
<code>ConversionPattern</code>	指定日志条目格式。可能的变量有: <code>%c</code> (类别)、 <code>%d</code> (日期, 后面是大括号中的日期变量)、 <code>%m</code> (日志消息)、 <code>%n</code> (newline)、 <code>%p</code> (优先级 - 日志级别)、 <code>%r</code> (相对时间, 毫秒)、 <code>%R</code> (相对时间, 秒)、 <code>%t</code> (线程时间) 和 <code>%T</code> (线程 ID)。	<code>%d {%-m-%d-%Y %H:%M:%S.%I} %-5p %c - %m%n</code>

maxMessageSize	设置日志发送的消息的最大大小（单位是字节）。默认为 16384。您必须提供一个比目前为 log-searchprocess.cfg 文件中所选的附加器设置的值更高的值。	16384
----------------	--	-------

attribute_value

语法: <string>

描述: 为所选的附加器属性提供更新值。您为 maxFileSize、maxBackupIndex 和 maxMessageSize 属性提供的值必须高于当前为 log-searchprocess.cfg 文件中的属性设置的值。换句话说，如果 searchprocessAppender 的设置目前设为 10000000，您只能提交比 10000000 大的新 maxFileSize 值。

用法

您可以在运行搜索时使用 `noop` 命令启用或禁用搜索优化。启用或禁用搜索优化可以帮助您解决特定类型的搜索问题。例如：您可以尝试禁用和启用搜索优化，以确定它们是否会导致搜索完成较为缓慢。

关于为 Splunk 平台部署中所有用户通过 `limits.conf` 管理搜索优化的信息，请参阅《搜索手册》中的“内置优化”。

使用 `noop` 命令管理所有搜索优化

`noop` 命令可以为单个搜索运行启用或禁用所有搜索优化。

如果在 `limits.conf` 中为 Splunk 部署启用了所有搜索优化，您可以向搜索字符串的末尾添加下列参数以在您运行该搜索时禁用所有优化：

```
.... | noop search_optimization=false
```

如果在 `limits.conf` 中禁用了 Splunk 部署的所有搜索优化，您可以向搜索字符串的末尾添加以下参数以在您运行该搜索时启用所有搜索优化：

```
.... | noop search_optimization=true
```

使用 `noop` 命令管理特定搜索优化

您可以使用 `optimization_type` 参数选择性地禁用或启用特定类型的搜索优化。

此处是禁用搜索的断言合并和断言下推优化的一组 `noop` 参数的示例。

```
.... | noop search_optimization.predicate_merge=f, search_optimization.predicate_push=f
```

只有当您在 `limits.conf` 中启用所有优化时，此示例才有效。

当您为 `limits.conf` 中的 `[search_optimization]` 段落设置 `enabled=false` 时，即为您的 Splunk 平台部署禁用所有搜索优化。使用此 `limits.conf` 配置，您的搜索可以使用 `noop` 以启用所有优化并选择性禁用特定优化类型。

例如：如果您在 `limits.conf` 中将 `[search_optimization]` 段落设置为 `enabled = false`，以下搜索将启用除投影清除外的所有优化。

```
index=_internal | eval c = x * y / z | stats count BY a, b | noop search_optimization=true,
search_optimization.projection_elimination=false
```

但是，当您在 `limits.conf` 中为 `[search_optimization]` 段落设置 `enabled=false` 时，您的搜索无法启用特定优化类型，除非满足特定条件。请参阅“`noop` 如何与 `limits.conf` 搜索优化设置互操作”。

Noop 命令如何与 `limits.conf` 搜索优化设置互操作

在使用 `noop` 命令启用或禁用优化类型之前，查看 `limits.conf` 中 Splunk 平台部署的搜索优化配置情况。只有在启用 `[search_optimization]` 后，搜索处理器才会遵守优化类型的 `limits.conf` 设置。

例如：如果在 `limits.conf` 中 `[search_optimization]` 段落设置为 `enabled=true`，搜索处理器将检查 `limits.conf` 中启用还是禁用了单个优化类型。另一方面，如果 `[search_optimization]` 段落设置为 `enabled = false`，搜索处理器不检查其他优化类型的设置。处理器假定所有优化类型均设置为 `enabled=false`。

当您用处理器为单个搜索启用或禁用搜索优化时，此搜索处理器逻辑将影响 `noop` 命令的工作方式。

例如，想象一下，您在 `limits.conf` 中已经有以下配置：

```
[search_optimization]
enabled=false
```

```
[search_optimization::projection_elimination]
enabled=false
```

通过此配置，搜索处理器将忽略已禁用的投影清除优化。因为 [search_optimization] 已禁用，搜索处理器假定所有优化已禁用。

假设您有了此配置，并运行以下搜索，该搜索使用 noop 命令启用搜索优化：

```
.... | noop search_optimization=true
```

当您这么做时，会启用搜索优化，但搜索处理器会在 limits.conf 中看到投影清除优化为禁用状态。处理器在除投影清除外所有优化类型已启用的情况下运行搜索。

相反，在搜索中使用 noop 命令启用搜索优化，选择性地启用投影清除优化：

```
.... | noop search_optimization=true search_optimization.projection_elimination=true
```

此搜索运行时，会覆盖两个 limits.conf 设置：[search_optimization] 的设置和 [search_optimization::projection_elimination] 的设置。搜索在启用所有优化的情况下运行。

使用 *noop* 为搜索设置调试频道

log_<level> 参数允许您在特定的日志级别（如 debug 或 warn）为搜索设置调试频道。如果您需要为特定的搜索设置日志级别，但是没有对 Splunk 平台实现的 CLI 访问权限，那么可以使用它。

Splunk 平台会在分析 noop 命令之后更改日志级别。它可以在搜索头分析其他搜索命令的参数之前完成这项工作，即使位于搜索字符串中的那些命令之后。例如，下面的搜索将正确地记录来自 makeresults 命令的一些调试消息，尽管它位于 noop 命令之前：

```
| makeresults count=1 | noop log_debug="MakeResultsProcessor"
```

但是，log_<level> 参数不能按操作顺序设置优先于 SPL 参数处理的搜索流程组件的日志级别。例如，LicenseMgr 是其中一个组件。当您运行此搜索时，即使您在 SPL 中指定了 debug，它仍然会以 LicenseMgr 的 info 默认级别记录日志。

```
index=_internal | head 1 | noop log_debug="LicenseMgr"
```

如果您有命令行访问权限并需要调试该组件或类似组件的问题，您可以直接修改 \$SPLUNK_HOME/etc/log-searchprocess.cfg，在派遣搜索并在 search.log 中生成更多详细输出之前，设置日志级别。

noop 命令必须是流管道的一部分。因为 Splunk 软件对搜索头执行参数解析，然后将搜索推送到索引器，所以要确保 noop 命令是流管道的一部分。将 noop 命令放在搜索字符串中的第一个非流命令之前。一种简单的方法是把它放在搜索字符串的第一个命令之后，这就是通常所说的 search。

log_<level> 参数支持通配符匹配。您还可以为相同搜索中的不同调试通道设置不同的日志级别。

```
.... | noop log_debug=Cache* log_info="SearchOperator:kv,SearchOperator:multikv"
```

有关日志和为调试通道设置日志级别的更多信息，请参阅《故障排除手册》中的“Splunk 记录有关它自身的哪些内容”。

使用 *noop* 将 *log-searchprocess.cfg* 附加器属性更改应用于搜索任务

在进行调试时，您可以使用 noop 将 log-searchprocess.cfg 附加器的更改属性应用于单个搜索运行。附加器是日志组件的特定子组的配置块。示例附加器包括 searchprocessAppender、watchdog_appender 和 searchTelemetryAppender。您可以使用 * 通配符选择所有附加器。

例如，以下搜索将 search.log 文件的最大大小更改为 50 MB，并将汇总的 search.log 文件的最大数量设为 99。

```
.... | noop log_appender="searchprocessAppender;maxFileSize=50000000;maxBackupIndex=99"
```

这些更改应用于特定搜索的生存期。这些不会保存或应用于其他搜索。

您只能更改以下附加器属性的值：maxFileSize、maxBackupIndex、ConversionPattern 和 maxMessageSize。您为 maxFileSize、maxBackupIndex 和 maxMessageSize 提供的值必须高于 log-searchprocess.cfg 中这些附加器属性的当前值。

关于更改附加器属性进行日志调试的更多信息，请参阅《故障排除手册》中的“启用日志调试”。

pr job

prjob 命令是不受支持的试验性内部命令。请参阅“关于内部命令”。

描述

使用 `prjob` 命令在分布式搜索环境中实现 SPL 搜索的并行化简搜索处理。`prjob` 命令分析指定的 SPL 搜索，并尝试通过以下方式减少搜索运行时间：在第一个非流 SPL 命令（如搜索中的 `stats` 或 `transaction`）之前自动放置一个 `redistribute` 命令。它提供的功能与 `redistribute` 命令相同，但语法更简单。与 `redistribute` 命令类似，使用 `prjob` 命令可自动加速汇总大量搜索结果的高基数搜索。

语法

```
pr job [<subsearch>]  
或  
pr job [num_ofReducers=<int>] [<subsearch>]
```

必要参数

subsearch

语法：[<subsearch>]

描述：指定 `prjob` 命令尝试并行处理的搜索字符串。

可选参数

num_ofReducers

语法：[num_ofReducers=<int>]

描述：指定索引器池中可用作中间化简器的合格索引器的数量。例如：如果搜索在 10 个索引器上运行，而且配置设置为使用索引器池的 60%（最大值为 5）时，则表示只能将五个索引器用作中间化简器。如果将 `num_ofReducers` 的值设为大于 5，则由于限制，只有五个化简器可用。如果将 `num_ofReducers` 的值设为小于 5，则所使用的化简器数将从上限值 5 缩减。

`num_ofReducers` 的值由两组设置控制：

- `reducers`:
- `maxReducersPerPhase + winningRate`

中间化简器的数量由为 `reducers` 设置的值确定。如果没有为 `reducers` 设置任何值，则搜索会使用为 `maxReducersPerPhase` 和 `winningRate` 设置的值来确定中间化简器的数量。

例如：如果对 Splunk 进行了配置，以确保 `num_ofReducers` 的值设置为索引器池的 50%，而 `maxReducersPerPhase` 的值设置为四个索引器，则在六个搜索对等节点上运行的并行化简搜索会被分配为在三个中间化简器上运行。同样，在四个搜索对等节点上运行的并行化简搜索会被分配为在两个中间化简器上运行。但是，在十个搜索对等节点上运行的搜索将被限制为最多四个中间化简器。

用法

如果想运行并行化简任务，而又不想决定在何处插入 `redistribute` 命令或管理 `by-clause` 字段，请使用 `prjob` 命令而非 `redistribute` 命令。

`prjob` 命令可以只用作搜索的第一条命令。此外，必须在 `prjob` 命令中包括整个搜索。

要使用 `prjob` 命令，请在 `limits.conf` 配置文件的 `[search_optimization::pr_job_extractor]` 段落中将 `phased_execution_mode` 设置为 `multithreaded` 或 `auto`，并将 `enabled` 设置为 `true`。

`prjob` 命令不支持实时或详细模式搜索。使用 `prjob` 命令时可以进行实时或详细模式搜索，但是 `redistribute` 操作会被忽略。另外，不可以在同一个搜索中同时使用 `prjob` 命令和 `redistribute` 命令。

`prjob` 命令支持的命令与 `redistribute` 命令相同。有关更多信息，请参阅 `redistribute`。`prjob` 命令只会减少包含以下至少其中一种非流命令的 SPL 搜索的搜索运行时间：“…”

- `stats`
- `tstats`
- `streamstats`
- `eventstats`
- `sistats`
- `sichart`
- `sitimechart`
- `transaction` （仅在单个字段上）

示例

示例 1：在搜索中使用 `prjob` 命令会自动将 `redistribute` 命令放在搜索中的第一个非流 SPL 命令之前。这会加速要汇总大量结果的 `stats` 搜索。搜索的 `stats count by host` 部分会在中间化简器上处理，而搜索头会汇总结果。

因此，以下搜索：

```
| prjob [search index=myindex | stats count by host]
```

会被转换为：

```
search index=myindex | redistribute | stats count by host
```

示例 2：加速包括 `eventstats` 的搜索，并使用 `sitimechart` 针对 `timechart` 操作进行统计计算。中间化简器处理 `eventstats`、`where` 和 `sitimechart` 操作。搜索头运行 `timechart` 命令，以将简化的 `sitimechart` 统计信息转换为可供可视化使用的已排序结果。

```
| prjob [search index=myindex | eventstats count by user, source | where count>10 | sitimechart max(count) by source | timechart max(count) by source]
```

示例 3：加速使用 `tstats` 来生成事件的搜索。`tstats` 命令必须放在子搜索的开头，并将 `prestats=t` 搭配 `timechart` 命令一起使用。`sitimechart` 命令在中间化简器上处理，而 `timechart` 命令在搜索头上处理。

```
| prjob [search index=myindex | tstats prestats=t count by _time span=1d | sitimechart span=1d count | timechart span=1d count]
```

示例 4：`eventstats` 和 `where` 命令在化简器上进行并行处理，而 `sort` 命令和其他任何后续命令都在搜索头上处理。发生这种情况是因为 `sort` 命令是一种 `prjob` 命令不支持的非流命令。

`prjob` 命令对该搜索没有影响。

```
| prjob [ search index=myindex | eventstats count by user, source | where count >10 | sort 0 -num(count) | ...]
```

runshellscript

`runshellscript` 命令是不受支持的试验性内部命令。请参阅“关于内部命令”。

描述

对于 Splunk Enterprise 部署，执行脚本式告警。该命令不能用作搜索命令。

语法

```
runshellscript <script-filename> <result-count> <search-terms> <search-string> <savedsearch-name> <description>
<results-url> <deprecated-arg> <results_file>
```

用法

脚本文件须位于 `$SPLUNK_HOME/etc/system/bin/scripts` 或 `$SPLUNK_HOME/etc/apps/<app-name>/bin/scripts` 中。下表列出了传递给脚本的参数。这些参数未经验证。

参数	描述
\$0	脚本的文件名。
\$1	结果计数，或返回的事件数。
\$2	搜索词。
\$3	完全限定的查询字符串。
\$4	Splunk 中保存的搜索的名称。
\$5	描述或触发原因。例如，“事件数大于 1”。
\$6	指向保存的搜索结果的链接。
\$7	已弃用 – 空字符串参数。
\$8	结果文件 <code>results.csv</code> 的路径。结果文件包含原始结果。

另请参阅

script

sendalert

sendalert 命令是不受支持的试验性内部命令。请参阅“关于内部命令”。

描述

使用 sendalert 命令调用自定义告警操作。此命令收集 alert_actions.conf 文件中的告警操作的配置、已保存的搜索及通过命令参数传递的自定义参数。然后命令会执行标记替换。命令确定要运行的告警操作脚本和参数，新建告警操作有效负载并执行脚本，使用 STDIN 将有效负载交给脚本进程。

运行自定义脚本时，sendalert 命令优先采用 alert_actions.conf 文件中的 maxtime 设置，且若运行时间超过配置的阈值，将终止进程。默认情况下，阈值设为 5 分钟。

请参阅《开发用于 Splunk Web 的视图和应用》手册中的“使用自定义告警操作的高级选项”。

语法

```
sendalert <alert_action_name> [results_link=<url>] [results_path=<path>] [param.<name>=<"value">...]
```

必要参数

alert_action_name

语法: <alert_action_name>

描述: alert_actions.conf 文件中配置的告警操作的名称

可选参数

results_link

语法: results_link=<url>

描述: 将 URL 链接设置为搜索结果。

results_path

语法: results_path=<path>

描述: 设置包括搜索结果的文件的位置。

param.<name>

语法: param.<name>=<"value">

描述: 参数名称和值。可以使用此名称和值对指定很多事项，如阈值、团队名称或信息文本。

用法

当您在即席搜索中使用 sendalert 命令，如果有大量搜索结果，则会多次调用命令。出现这种情况是因为默认启用预览“统计”选项卡上的搜索结果。如果您正在使用即席搜索测试 sendalert 命令，测试关闭预览以避免多次调用命令。

如果 sendalert 命令包含在已保存的搜索中，如计划的报表或计划的搜索，只会调用一次命令。

搜索结果格式

如果在搜索或告警操作中使用 sendalert 命令，搜索结果会以 CSV 格式存储在 dispatch 目录中的归档文件中。文件名为 results.csv.gz。搜索结果的默认格式为 SRS，这是 Splunk 特定的二进制搜索结果格式。以 CSV 格式存储归档文件，这样脚本可处理结果文件。脚本不可分析默认的 SRS 格式。

通过 forceCsvResults 设置控制归档的搜索结果格式。此设置位于 alert_actions.conf 文件的 [default] 段落中。

示例

示例 1: 调用不带任何参数的告警操作。告警操作脚本检查是否包含缺失且上报错误的必需参数。

```
... | sendalert myaction
```

示例 2: 触发 hipchat 自定义告警操作并将空间和消息作为自定义参数传递。

```
... | sendalert hipchat param.room="SecOps" param.message="There is a security problem!"
```

示例 3：触发 servicenow 告警选项。

```
... | sendalert servicenow param.severity="3" param.assigned_to="DevOps" param.short_description="Splunk Alert: this is a potential security issue"
```

在 CLI 中搜索

关于在 CLI 中搜索

如果使用 Splunk Enterprise，则可以使用 Splunk CLI 从命令行发出搜索命令。本主题将讨论如何使用 CLI 进行搜索。如果您要了解如何访问 CLI 并需要获取相关帮助信息，请参阅《管理员手册》中的“关于 CLI”。

搜索的 CLI 帮助

您可以使用 `search` 命令运行历史搜索，并使用 `rtsearch` 命令进行实时搜索。下表列出了一些有用的与搜索相关的 CLI 帮助对象。要查看每个对象的完整帮助信息，可在 CLI 中键入：

```
./splunk help <object>
```

对象	描述
<code>rtsearch</code>	返回实时搜索的参数和语法。
<code>search</code>	返回历史搜索的参数和语法。
<code>search-commands</code>	返回可在 CLI 中使用的搜索命令的列表。
<code>search-fields</code>	返回默认字段的列表。
<code>search-modifiers</code>	返回可用于缩小搜索范围的基于搜索和时间的修饰符列表。

在 CLI 中搜索

CLI 中的历史搜索和实时搜索与 Splunk Web 中的搜索工作方式相同，只是不会随搜索结果一起呈现时间线，也没有默认的时间范围。而是将结果显示为一个原始事件列表或一个表，具体取决于搜索类型。

- 有关更多信息，请参阅《搜索手册》中“搜索概述”一章的“搜索类型”部分。

CLI 搜索的语法与 Splunk Web 搜索的语法相似，只是您可以传递查询以外的参数来指定搜索的时间限制，在哪里运行搜索以及如何显示结果。

- 有关 CLI 搜索选项的更多信息，请参阅本章中的下一主题“CLI 搜索语法”。
- 有关如何从本地服务器搜索远程 Splunk 服务器的更多信息，请参阅《管理员手册》中的“访问和使用远程服务器上的 CLI”。

CLI 中的搜索语法

本主题将快速介绍在 CLI 中使用 `search` 和 `rtsearch` 命令时可以使用的语法和选项。

CLI 搜索的语法与从 Splunk Web 运行的搜索语法相似，只是您可以传递搜索对象以外的参数来控制搜索的时间限制，指定要在其中运行搜索的服务器，并指定结果显示的方式。

```
search | rtsearch [object][-parameter <value>]
```

搜索默认

默认情况下，当您运行 CLI 中的搜索时，搜索将“所有时间”用作时间范围。您可以使用其中一个 CLI 搜索参数，如 `earliest_time`、`index_earliest` 或 `latest_time` 指定时间范围。

当您使用 CLI 运行历史搜索时，返回前 100 个事件。使用 `maxout` 搜索参数以指定要返回的事件数量。

搜索对象

搜索对象可以括在单引号 (' ') 中，可以是关键字、表达式或一系列搜索命令。在 Windows 操作系统上，搜索对象需要括在双引号 (" ") 中。

- 有关搜索的更多信息，请参阅《搜索教程》中的“开始搜索”。
- 有关每个搜索命令的简单描述，请参阅《搜索参考》中的“命令快速参考”。
- 有关 Splunk 概念、特性、搜索命令和函数的快速参考，请参阅《搜索参考》中的“快速参考指南”。

搜索对象不仅可以包括关键字和搜索命令，还可以包括字段和修饰符，以指定您想要检索的事件以及要生成的结果。

- 有关字段的更多信息，请参阅《搜索教程》中的“使用字段搜索”。
- 有关默认字段及其使用方法的更多信息，请参阅《知识管理器手册》中的“使用默认和内部字段”。
- 有关时间调节器的更多信息，请参阅《搜索参考》中的“与搜索结合使用的时间调节器”。

搜索参数

搜索参数是用于控制搜索运行方式或搜索结果显示方式的选项。所有这些参数均为可选参数。接受布尔值的参数支持 `0`, `false`, `f`, `no` 作为否定值, `1`, `true`, `t`, `yes` 作为肯定值。

在指定所有命令和命令参数后，在搜索结束时指定这些搜索参数。请参阅示例 4。

参数	值	默认	描述
<code>app</code>	<code><app_name></code>	<code>search</code>	指定要在其中运行搜索的应用的名称。
<code>batch</code>	<code><bool></code>	<code>F</code>	指示如何在预览模式下处理更新。
<code>detach</code>	<code><bool></code>	<code>F</code>	触发异步搜索并显示搜索的任务 ID 和 TTL。
<code>earliest_time</code>	<code><time-modifier></code>	—	搜索开始时间的相对时间调节器。对于 <code>search</code> 是可选项，对于 <code>rtsearch</code> 是必填项。
<code>header</code>	<code><bool></code>	<code>T</code>	指示是否在表输出模式下显示标题。
<code>index_earliest</code>	<code><time-modifier></code>		搜索的开始时间。对于搜索语言，这可以 <code>epoch</code> 或相对时间调节器表示，并使用与 “earliest” 和 “latest” 时间调节器相同的语法。对于 <code>search</code> 和 <code>rtsearch</code> ，都是可选项。
<code>index_latest</code>	<code><time-modifier></code>		搜索的结束时间。对于搜索语言，这可以 <code>epoch</code> 或相对时间调节器表示，并使用与 “earliest” 和 “latest” 时间调节器相同的语法。对于 <code>search</code> 和 <code>rtsearch</code> ，都是可选项。
<code>latest_time</code>	<code><time-modifier></code>	—	搜索结束时间的相对时间调节器。对于 <code>search</code> ，如果未指定此参数，则默认为最后的时间（或数据中最后一个事件的时间），以便同时将任何“将来”事件包含在内。 对于 <code>rtsearch</code>，这是必要参数 ，如果未指定此参数，则实时搜索将不会运行。
<code>max_time</code>	<code><number></code>	<code>0</code>	搜索任务在完成之前运行的时间长度（秒）。值为 <code>0</code> 表示没有时间限制。
<code>maxout</code>	<code><number></code>	<code>search, 100</code> <code>rtsearch, 0</code>	导出事件时返回或发送到 <code>stdout</code> 的事件数量上限。值为 <code>0</code> 表示对输出的事件数没有限制。
<code>output</code>	<code>rawdata</code> 、 <code>table</code> 、 <code>csv</code> 、 <code>auto</code>	非转换搜索 请使用 <code>rawdata</code> 。 转换搜索 请使用 <code>table</code> 。	指示如何显示任务。
<code>preview</code>	<code><bool></code>	<code>T</code>	指示应对报表搜索进行预览（显示为计算的结果）。
<code>timeout</code>	<code><number></code>	<code>0</code>	搜索任务在运行之后允许存在的时间长度（秒）。值为 <code>0</code> 表示任务在运行之后立即取消。
<code>uri</code>	<code>[http https]://name_of_server:management_port</code>		指定服务器名称和管理端口。 <code>name_of_server</code> 可以为 Splunk 服务器的完全解析域名或 IP 地址。 默认 <code>uri</code> 值为 <code>mgmtHostPort</code> 值，即您在 Splunk 服务器的 <code>web.conf</code> 中定义的值。 更多信息请参阅《管理员手册》中的“访问和使用远程 Splunk 服务器上的 CLI”。
<code>wrap</code>	<code><bool></code>	<code>T</code>	指示行的长度超过终端宽度时是否自动换行。

示例

您可以在 CLI 帮助信息中看到更多示例。

1. 检索昨天发生的满足根会话的事件

```
./splunk search "session root daysago=1"
```

2. 检索与 Web 访问错误匹配的事件并分离搜索

```
./splunk search 'eventtype=webaccess error' -detach true
```

3. 运行带窗口的实时搜索

```
./splunk rtsearch 'index=_internal' -earliest_time 'rt-30s' -latest_time 'rt+30s'
```

请参阅《管理员手册》中的“CLI 中的实时搜索和报表”部分所提供的更多示例。

4. 返回唯一主机名列表

您可以有两种建议的方式进行。第一种方式是使用 stats 命令：

```
./splunk search 'index=* | stats count by host | fields - count' -preview true
```

或者，由于仅对主机字段感兴趣，您可以使用 metadata 命令：

```
./splunk search '| metadata type=hosts | fields host' -preview true
```

这里，*-preview* 标记可选，并用于查看返回的结果。与此相反，table 命令不像 fields 命令，一般需要所有输入后，才能进行非预览输出。在这种情况下，您将需要使用 *preview* 标记以能够查看搜索结果。

5. 返回昨天的内部事件

```
./splunk search 'index=_internal' -index_earliest -1d@d -index_latest @d
```