splunk> Schnellreferenz

KONZEPTE

Indizierungs- und Suchzeitpunkt

Bei der Ausführung zum Zeitpunkt der Indizierung werden Daten aus einer Quelle auf einem Host gelesen und als Sourcetyp klassifiziert. Es werden Zeitstempel extrahiert und die Daten in einzelne Ereignisse geparst. Zum Segmentieren der Ereignisse für die Anzeige in Suchergebnisse werden Zeilenumbruchregeln angewandt. Jedes Ereignis wird in einen Index auf der Festplatte geschrieben, aus dem es später bei Suchanfragen abgerufen wird.

Beim Starten einer Suche werden indizierte Ereignisse von der Festplatte abgerufen. *Felder* werden aus dem Rohtext des Ereignisses extrahiert. Diese Ereignisse können dann mithilfe der in Splunk Enterprise enthaltenen Suchsprache SPL umgewandelt und zum Erstellen von *Berichten* und *Visualisierungen* für die Anzeige in Dashboards verwendet werden.

Indizes

Beim Hinzufügen von Daten parst Splunk Enterprise sie in einzelne Ereignisse, extrahiert den Zeitstempel, wendet Zeilenumbruchregeln an und speichert die Ereignisse in einem Index. Sie können neue Indizes für unterschiedliche Eingaben erstellen. Standardmäßig werden die Daten im Index „main" gespeichert. Bei einer Suche werden Ereignisse aus einem oder mehreren Indizes abgerufen.

Ereignisse

Ein Ereignis besteht aus einem Satz von Werten, denen ein Zeitstempel zugeordnet ist. Es handelt sich um einen einzelnen Dateneintrag, der ein- oder mehrzeilig sein kann. Ein Ereignis kann ein Textdokument, eine Konfigurationsdatei oder auch eine komplette Stack Trace-Datei etc. sein. Dies ist ein Beispiel für ein Ereignis in einem Weblog:

```
173.26.34.223 - - [01/Jul/2009:12:05:27 -0700] „GET/trade/
app?action=logout HTTP/1.1" 200 2953
```

Zum Suchzeitpunkt können indizierte Ereignisse, die mit einer angegebenen Suchzeichenfolge übereinstimmen, Eventtypen zugeordnet werden. Sie können auch Transaktionen für die Suche und Gruppierung von Ereignissen definieren, die konzeptionell zusammenhängen, jedoch über eine gewisse Zeitspanne stattfinden. Transaktionen können aus mehreren Schritten bestehende, geschäftliche Aktivitäten darstellen, wie beispielsweise alle Ereignisse im Zusammenhang mit einer einzelnen Kundensitzung auf der Website eines Online-Händlers.

Host

Als *Host* bezeichnet man das physische oder virtuelle Gerät, von dem ein Ereignis stammt. Mit dem Feld „host" lassen sich ganz leicht alle Daten ermitteln, die von einem bestimmten Gerät stammen.

Quelle und Sourcetyp

Die *Quelle* ist der Name der Datei, des Verzeichnisses, des Datenstreams oder sonstiger Eingabe, aus der ein bestimmtes Ereignis stammt. Quellen werden als Sourcetypen klassifiziert. Diese sind entweder schon als bekannte Formate vordefiniert oder können vom Benutzer individuell definiert werden. Beispiele für bekannte *Sourcetypen* sind HTTP-Webserverlogs und Windows-Ereignisprotokolle. Ereignisse mit demselben Sourcetyp können aus unterschiedlichen Quellen stammen.
Zum Beispiel Ereignisse aus der Datei `source=/var/log/messages` und von einem syslog-Eingabeport `source=UDP:514` haben oftmals denselben Sourcetyp `sourcetype=linux_syslog`.

Felder

Felder sind Paare aus Name und Werte, die durchsucht werden können und Ereignisse voneinander unterscheiden, da nicht alle Ereignisse dieselben Felder und Feldwerte enthalten. Mit Feldern können Sie individuell zugeschnittene Suchen erstellen, um mithilfe der Suchbefehle exakt die gewünschten Ereignisse abzurufen. Wenn Splunk Enterprise Ereignisse bei der Indizierung und Suche verarbeitet, werden dabei auf der Grundlage von Definitionen in der Konfigurationsdatei und vom Benutzer definierter Muster Felder extrahiert.

Tags

Tags sind Aliase für bestimmte Feldwerte. Sie können einer Kombination aus Feldname und -wert, einschließlich Eventtypen, Hosts, Quellen und Sourcetypen, ein oder mehrere Tags zuordnen. Mit Tags können Sie zusammengehörige Feldwerte gruppieren oder abstrakte Feldwerte wie IP-Adressen oder IDs nachverfolgen, indem Sie ihnen aussagekräftige Bezeichnungen zuordnen.

FUNKTIONEN VON SPLUNK ENTERPRISE

Benachrichtigungen

Benachrichtigungen werden ausgelöst, wenn Suchergebnisse historischer oder Echtzeitsuchen bestimmte Bedingungen erfüllen. Benachrichtigungen können so konfiguriert werden, dass sie Aktionen auslösen: Sie können z. B. eine Benachrichtigung an festgelegte E-Mail-Adressen senden, Benachrichtigungsinformationen in einem RSS-Feed posten und benutzerdefinierte Skripte ausführen, um beispielsweise ein „Benachrichtigungsereignis" im Syslog zu erfassen.

Datenmodell

Ein Datenmodell ist eine hierarchisch strukturierte, zum Zeitpunkt der Suche gültige Zuordnung von semantischem Wissen über eine oder mehrere Datenmengen. Ein Datenmodell codiert das Domänenwissen, das zum Erstellen einer Vielzahl spezialisierter Suchen innerhalb der Datenmengen notwendig ist. Splunk Enterprise nutzt diese spezialisierten Suchen wiederum zum Erstellen von Berichten mit der Pivot-Funktion. Datenmodellobjekte stehen für verschiedene Datenmengen innerhalb der größeren, von Splunk Enterprise indizierten Datenmenge.

Pivot

Pivot bezeichnet die Tabellen-, Diagramm- oder Datenvisualisierung, die Sie mit dem Pivot-Editor erstellen. Der Pivot-Editor ermöglicht Benutzern, durch Datenmodellobjekte definierte Attribute einer Tabellen-, Diagramm- oder Datenvisualisierung zuzuordnen, ohne die für deren Erstellung notwendigen Suchen schreiben zu müssen. Pivots können als Berichte gespeichert und für die Anzeige in Dashboards verwendet werden.

Suche

Die Suche ist die Hauptmethode, mit Splunk Enterprise-Benutzer in Daten navigieren. Sie können eine Suche schreiben, um Ereignisse aus einem Index abzurufen, Statistikbefehle zum Berechnen von Metriken und Erstellen von Berichten auszuführen, innerhalb eines rollenden Zeitfensters nach bestimmten Bedingungen zu suchen, Muster innerhalb Ihrer Daten zu erkennen, künftige Trends zu prognostizieren usw. Suchen können als Berichte gespeichert und für die Anzeige in Dashboards verwendet werden.

Berichte

Bei Berichten handelt es sich um gespeicherte Suchen und Pivots. Sie können Berichte ad-hoc oder nach einem Zeitplan in regelmäßigen Intervallen ausführen und für einen geplanten Bericht festlegen, dass Benachrichtigungen ausgelöst werden, wenn die Ergebnisse bestimmte Bedingungen erfüllen. Berichte können als Dashboard-Teilfenster zu Dashboards hinzugefügt werden.

Dashboards

Dashboards bestehen aus Teilfenstern, die bestimmte Module wie Suchfelder, Felder, Diagramme, Tabellen, Formulare etc. enthalten. Dashboard-Teilfenstern sind meist mit gespeicherten Suchen oder Pivots verknüpft. In den Teilfenstern werden die Ergebnisse abgeschlossener Suchen sowie Daten aus im Hintergrund ausgeführten Echtzeitsuchen angezeigt.

KOMPONENTEN VON SPLUNK ENTERPRISE

Apps

Bei Apps handelt es sich um eine Sammlung von Konfigurationen, Wissensobjekten und von Kunden entworfenen Ansichten und Dashboards, die die Splunk Enterprise-Umgebung erweitern und an spezifische Anforderungen operativer Teams wie etwa Administratoren von Unix- und Windows-Systemen, Experten für Netzwerksicherheit, Website-Manager, Business-Analysten usw. anpassen. Innerhalb einer Splunk Enterprise-Installation können mehrere Apps gleichzeitig ausgeführt werden.

Forwarder und Receiver

Ein Forwarder ist eine Splunk Enterprise-Instanz, die Daten an eine andere Splunk Enterprise-Instanz (einen Indexer oder einen anderen Forwarder) oder ein Drittanbietersystem weiterleitet. Ist die Splunk Enterprise-Instanz (ein Indexer oder Forwarder) für den Empfang von Daten von einem Forwarder konfiguriert, nennt man sie auch Receiver.

Indexer

Ein Indexer ist die Splunk Enterprise-Instanz, die Daten indiziert. Der Indexer wandelt die Rohdaten in Ereignisse um und speichert die Ereignisse in einem Index. Außerdem durchsucht der Indexer die indizierten Daten bei Suchanforderungen.

Search Head und Search Peer

In einer verteilten Suchumgebung ist der Search Head die Splunk Enterprise-Instanz, die Suchanforderungen an eine Reihe von Search Peers verteilt und die Ergebnisse beim Benutzer zusammenführt. Die Search Peers sind Indexer, die Suchanforderungen seitens des Search Head erfüllen. Eine Instanz, die nur für Suchen, nicht jedoch für Indizierungen verwendet wird, wird in der Regel als dedizierter Search Head bezeichnet.

Suchsprache

Eine Suche ist eine Abfolge von Kommandos und Argumenten, verkettet über das „ | “ (Pipe) Zeichen welches das Ergebnis eines Kommandos/Suche weitergibt an ein Kommando auf der rechten Seite.

search-args | cmd1 cmd-args | cmd2 cmd-args | ...

Suchkommandos werden verwendet um indizierte Daten zu verarbeiten und unerwünschte Ergebnisse zu filtern, weiter Ergebnisse zu extrahieren, Werte zu berechnen, transformieren und statistisch zu analysieren. Suchergebnisse die von einem Index zurück geliefert werden, können als dynamisch angelegte Tabelle betrachtet werden. Jedes weiter Suchkommando verändert dabei den Inhalt der Tabelle. Jedes indexierte Ereigniss ist eine Reihe, mit Spalten für jeden Feldwert. Spalten beinhalten Basisinformation über die Daten, genauso wie Spalten die während einer Suche dynamisch extrahiert werden.

Am Anfang einer Suche steht immer implizit ein Such-nach-einem-Event Kommando, welches verwendet wird zur Suche nach Schlüsselwörtern (z.B. error), Boolean Ausdrücken (z.B. (error OR failure) NOT success), Formulierungen (z.B. „database error“), Wildcards (z.B. fail* trifft auf fails, failure usw. zu), Feldwerte (z.B. code=404), Ungleichheiten (z.B. code!=404 oder code>200), und Feldern welche einen beliebigen oder keinen Wert haben (z.B. code=* oder NOT code=*).
So gibt z.B. folgende Suche:

sourcetype=“access_combined“ error | top 10 uri

access_combined Events von der Festplatte zurück die den Ausdruck „error“ (AND ist immer implizit zwischen Ausdrücken) enthalten, um dann aus den Ergebnissen die 10 meist verwendeten URI’s zu filtern.

Subsuchen

Eine Subsuche ist eine Argument zu einem Kommando welches als eigenständige Suche läuft und seine Ergebnisse an das Elternkommando weitergibt als einen Argumentwert. Subsuchen stehen in eckigen Klasmmern. Z.B. finde alle syslog Ereignisse des Anwenders der zuletzt einen Loginfehler bekam:

sourcetype=syslog [search login error | return user]

Beachte: die Subsuche gibt nur ein Ergebnis zurück da das return Kommando im Standard nur einen Wert liefert, aber es gibt Optionen für mehr Werte (z.B. | return 5 user).

Relative Zeitmodifikatoren

Neben der Möglichkeit den Suchzeitraum im User Interface zu bestimmen, können Sie auch einen Zeitraum über die earliest oder latest Modifikatoren direkt in der Suche angeben. Relative Zeiten werden in einer Zeichenkette spezifiziert welche die Zeit (Integer und Einheit) angibt, sowie optional ein „Snap to“ beinhalten kann:

[+ | -] <time_integer><time_unit>@<snap_time_unit>

Z.B.: „error earliest=-1d@d latest=-1h@h“ gibt die Ereignisse zurück die error enthalten die zwischen Gestern ab Mitternacht bis Heute zur letzten vollen Stunden auftauchten.

Zeiteinheiten: Angegeben als Sekunden(s), Minuten (m), Stunden (h), Tage (d), Woche (w), Monat (mon), Quartal (q), Jahr (y). <time_integer> ist im Standard immer 1 (z.B. ist „m“ das Selbe wie „1m“).

Snapping: Indiziert den nähesten oder spätesten Zeitpunkt auf die der Zeitwert auf- oder abgerundet wird. Snap rundet ab auf den spätesten Zeitpunkt nach der definierten Zeit. Wenn es z.B. 11:59:00 Uhr ist und Sie „snap to hours“ (@h) angeben wird auf 11:00:00 Uhr gesprungen und nicht auf 12:00:00 Uhr. Sie können auch auf einen expliziten Tag der Woche springen: benutzen Sie „@w0“ für Sonntag,
„@w1“ für Montag, usw.

Allgemeine Suchkommandos

KOMMANDO	BESCHREIBUNG
chart/ timechart	Gibt tabellarische Ergebnisse in zeitbasierter Darstellung zurück
dedup	Entfernt nachfolgende Ergebnisse die eine spezifische Bedingung erfüllen
eval	Berechnet einen Ausdruck (siehe auch EVAL FUNKTIONEN Tabelle)
fields	Entfernt Felder aus einem Suchergebnis
head/tail	Gibt die ersten/letzten N Ergebnisse zurück
lookup	Fügt Feldwerte aus einer externen Quelle hinzu
rename	Benennt ein Feld um; Wildcards möglich um multiple Felder zu spezifizieren
replace	Ersetzt spezifische Werte eines Feldes durch einen neuen Wert
rex	Spezifiziert reguläre Ausdrücke um Felder zu extrahieren
search	Filtert Ergebnisse die auf den Suchausdruck passen
sort	Sortiert Suchergebisse nach den angegeben Feldern
stats	Erzeugt Statistiken, optional gruppiert nach Feldern
top/rare	Stellt die häufigsten/seltensten gemeinsamen Wete eines Feldes dar
transaction	Gruppiert Suchergebnisse nach Transaktionen

Suchoptimierung

Der Schlüssel zu schnelleren Suchen ist die Menge der Daten auf ein Minimum zu limitieren die von der Festplatte geholt werden müssen, um die Ergebnisse dann so früh als möglich zu filtern, sodass die Ausführung auf so wenig als möglich Daten erfolgt.

Teilen Sie Daten auf separate Indexes auf, wenn Sie selten Suchen auf multiplen Datentypen ausführen. Z.B. senden Sie Webdaten in einen Index und Firwalldaten in einen anderen.

- Suchen Sie so spezifisch als möglich (z.B. fatal_error und nicht *error*)
- Limitieren Sie die Suchzeit aus den notwendigen Rahmen (z.B. -1h statt -1w)
- Filtern Sie unnötige Felder so früh als möglich in der Suche aus
- Filtern Sie Ergebnisse so früh als möglich vor Berechnungen aus
- Für Suchen die Reports erzeugen nutzen Sie die „Advanced Charting“ Sichten, statt der Flashtimeline-Sicht, welche zeitbasierte Sichten berechnet
- Deaktivieren Sie in der Flashtimeline „Discover Fields“, sofern es nicht wirklich benötigt wird
- Verwenden Sie „Summary Indexes“ um häufig genutzte Werte vorzuberechnen
- Stellen Sie sicher das die Disk I/O so schnell als möglich ist (> 800 IOPS & 10.000 RPM)

EVAL KOMMANDOS

Das eval Kommando berechnet einen Ausdruck und fügt den Ergebniswert einem Feld zu (z.B. „.. | eval force=mass*acceleration“). Folgende Tabelle listet eval Funktionen auf, gemeinsam mit einfachen arithmetischen Operatoren (+ - * / %), String Verknüpfungen (z.B. „.. | eval name=last.“,“.last‘) und BOOLEAN Operatoren (AND, OR, NOT, XOR, <>, <=, >=, !=, =, ==, LIKE)

FUNKTION	BESCHREIBUNG	BEISPIEL
abs(X)	Gibt den absoluten Wert von X zurück.	abs(number)
case(X,“Y“,...)	Nimmt Argumentpaare von X und Y, wobei X Boolean Ausdrücke sind, die, wenn auf TRUE evaluiert, das korresponierende Y Argument liefern.	case(error == 404,„Not found“, error == 500,„Internal Server Error“, error == 200,„OK“)
ceil(X)	Aufrunden einer Ziffer X auf den nächsten Integer.	ceil(1.9)
cidrmatch(“X“,Y)	Identifiziert IP Adresssen die zu einem spezifischen Subnetz gehören.	cidrmatch(“123.132.32.0/25“, ip)
coalesce(X,...)	Gibt den ersten Wert zurück der nicht Null ist.	coalesce(null(), “Returned Val“, null())
exact(X)	Ermittelt einen Ausdruck X mittels doppelt genauer Floating Point Arithmetik.	exact(3.14*num)
exp(X)	Gibt e^x zurück.	exp(3)
floor(X)	Abrunden einer Ziffer X auf den nächsten Integer.	floor(1.9)
if(X,Y,Z)	Wenn X TRUE, ist das Ergebnis der zweite Wert Y, wenn X FALSE, dann ist das Ergebnis der WERT Z.	if(error==200, “OK“, “Error“)
isbool(X)	Gibt TRUE zurück falls X ein Boolean ist.	isbool(field)
isint(X)	Gibt TRUE zurück falls X ein Integer ist.	isint(field)
isnotnull(X)	Gibt TRUE zurück falls X nicht NULL ist.	isnotnull(field)
isnull(X)	Gibt TRUE zurück falls X NULL ist.	isnull(field)
isnum(X)	Gibt TRUE zurück falls X eine Ziffer ist.	isnum(field)
isstr()	Gibt TRUE zurück falls X ein String ist.	isstr(field)
len(X)	Gibt die Zeichenlänge eines Strings X zurück.	len(field)
like(X,“Y“)	Gibt nur dann TRUE zurück wenn X dem SQLite Muster in Y entspricht.	like(field, “foo%“)
ln(X)	Nimmt eine Ziffer X und liefert den dazugehörenden Log.	ln(bytes)
log(X,Y)	Gibt den Log des ersten Arguments X zurück indem es als Ausgang das zweite Argument Y nutzt. Y ist im Standard 10.	log(number,2)
lower(X)	Gibt den Wert X in Kleinbuchstaben zurück.	lower(username)
ltrim(X,Y)	Gibt den Wert X zurück der mit den Zeichen in Y von Links weg abgeschnitten wird. Y entspricht im Standard Leerzeichen und Tabulatoren.	ltrim(“ ZZZabcZZZ “,“ Z“)
match(X,Y)	Gibt X zurück, falls X dem regulären Ausdruck in Y entspricht.	match(field, “^\d{1,3}\.\d$“)
max(X,...)	Gibt das Maximun zurück.	max(delay, mydelay)
md5(X)	Gibt den md5 Hash eines Stringswertes von X zurück.	md5(field)
min(X,...)	Gibt das Minimum zurück.	min(delay, mydelay)
mvcount(X)	Gibt die Anzahl der Werte von X zurück.	mvcount(multifield)
mvfilter(X)	Filtert ein Multiwertfeld basierend auf dem Boolean-Ausdruck X.	mvfilter(match (email, “net$“))
mvindex(X,Y,Z)	Gibt eine Submenge des Multiwertfeldes X zurück ab der Startposition (nullbasiert) Y bis zu X (optional)	mvindex(multifield, 2)
mvjoin(X,Y)	Vorgegeben sind ein Multiwertfeld X und einen Zeichentrenner Y; Fügt die individuellen Werte von X zusammen unter Verwendung von Y.	mvjoin(foo, “;“)
now()	Gibt die aktuelle Unixzeit zurück.	now()
null()	Diese Funktion erwartet keine Argumente und liefert NULL zurück.	null()
nullif(X,Y)	Vorgegeben sind die Argumente von X und Y; Liefert X zurück wenn die Argumente ungleich sind; Ansonsten wird NULL zurück gegeben.	nullif(fieldA, fieldB)
pi()	Gibt die Konstante pi zurück.	pi()
pow(X,Y)	Gibt X^Y zurück.	pow(2, 10)
random()	Liefert eine Pseudo-Zufallsziffer von 0 bis 2147483647.	random()
relative_time(X,Y)	Vorgegeben sind die Epochzeit X und die relative Zeitvorgabe Y, und liefert die Epochzeit von Y angewandt auf X zurück.	relative_time(now(), “-1d@d“)
replace(X,Y,Z)	Gibt einen String zurück der zusammengebaut wird aus den Ersetzungen des Strings Z für jedes Auftreten des Regex Strings Y im String X.	Gibt ein Datum zurück mit umgekehrtem Monat und Tag, sodass wenn die Eingabe 01/12/2011 ist, der Rückgabewert 12/01/2011 wäre; replace(date, “^(\d{1,2})/(\d{1,2})/“, “\2/\1/“)
round(X,Y)	Gibt den, auf die in Y benannte Anzahl der dezimalen Stellen, gerundeten Wert X zurück; Standard ist auf einen Integer zu Runden.	round*3.5(
rtrim(X,Y)	Gibt den Wert X zurück der mit den Zeichen in Y von Rechts weg abgeschnitten wird. Y entspricht im Standard Leerzeichen und Tabulatoren.	rtrim(“ ZZZabcZZZ “,“ Z“)
searchmatch(X)	Gibt TRUE zurück falls das Event auf den Suchstring X zutrifft	searchmatch(„foo AND bar“)

EVAL KOMMANDOS (Fortsetzung)

FUNKTION	BESCHREIBUNG	BEISPIEL
split(X, "Y")	Gibt X als ein Multiwertfeld zurück, aufgeteilt durch das Trennzeichen Y.	split(foo, " ; „)
sqrt(X)	Gibt die Quadratwurzel von X zurück.	sqrt(9)
strftime(X,Y)	Gibt den Epochzeitwert X, aufbereitet nach dem in Y vorgegeben Format, zurück.	strftime(_time, "%H:%M"))
strptime(X,Y)	Vorgegeben ist eine Zeit repräsentiert durch den String X, und gibt Werte zurück die aus dem Format Y geformt werden.	strptime(timeStr, "%H:%M"))
substr(X,Y,Z)	Gibt eine Untermenge des Substring X, von der Startposition Y (1-basiert) für Z (optional) Zeichen, an.	substr("string", 1, 3)+substr("string", -3)
time()	Gibt die Systemzeit mit einer Mikrosekundenauflösung zurück.	time()
tonumber(X,Y)	Wandelt einen Eingabestring X in eine Ziffer, wobei Y (optional, im Standard 10) die Basisziffer angibt in die verwandelt werden soll.	tonumber("0A4", 16)
tostring(X,Y)	Gibt einen Feldwert X als einen String zurück. Falls der Wert X eine Ziffer ist, wird er inen String umgewandelt; Ebenso falls er ein Boolean ist, entweder „TRUE" oder „FALSE". Wenn X eine Ziffer ist, wird das zweite Argument Y optional und kann entweder „hex" (wandelt X nach Hexadezimal), „commas" (formatiert X mit Kommas und zwei Dezimalstellen), oder „duration" (konvertiert Sekunden X in ein lesbares Zeitformat HH:MM:SS) sein.	Dieses Beispiel gibt zurück: foo=615 und foo2=00:10:15: ... | eval foo=615 | eval foo2= tostring(foo, "duration")
trim(X,Y)	Gibt den Wert X zurück der mit den Zeichen in Y von beiden Seiten abgegelichen wird. Falls Y nicht angegeben ist werden Leerzeichen und Tabulatoren abgeschnitten.	trim(" ZZZabcZZZ "," Z")
typeof(X)	Gibt eine Stringdarstellung des Typs zurück.	Dieses Beispiel gibt zurück: „NumberStringBoolInvalid": typeof(12) + typeof("string") + typeof(1==2) + typeof(badfield)
upper(X)	Gibt den Wert X in Grossbuchstaben zurück.	upper(username)
urldecode(X)	Gibt die URL X dekodiert zurück.	urldecode("http%3A%2F%2Fwww.splunk.com%2Fdownload%3Fr%3Dheader")
validate(X,Y,...)	Vorgegeben sind Paare von Argumenten, Booleanausdrücke X und Strings Y, und liefert den String Y entsprechend dem ersten Ausdruck X der False entspricht; als Standard wird NULL zurück gegeben falls alle Werte TRUE sind.	validate(isint(port), "ERROR: Port is not an integer", port >= 1 AND port <= 65535, "ERROR: Port is out of range")

Allgemeine stats Funktionen

Funktion	Beschreibung
avg (X)	Gibt den Durchschnitt der Werte des Feldes X zurück.
count (X)	Gibt die Anzahl des Auftretens des Feldes X zurück. Um einen spezifischen Feldwert der übereinstimmen muss anzugeben, formatieren Sie X als eval(field="value")
dc (X)	Gibt die Anzahl eindeutiger Werte des Feldes X zurück.
first (X)	Gibt den ersten auftauchenden Wert des Feldes X zurück. Im allgemeinen, enspricht der erste auftauchende Werte des Feldes der chronoligisch bedeutendsten Instanz des Feldes.
last (X)	Gibt den zuletzt auftauchenden Wert des Feldes X zurück.
list (X)	Gibt die Liste aller Werte des Feldes X als einen Multiwerteintrag zurück. Die Reihenfolge der Werte bildet die Reihenfolge der Eingabeereignisse ab.
max (X)	Gibt den höchsten Wert des Feldes X zurück. Falls die Werte von X nicht numerisch sind, wird das Maximum in der lexikographischen Reihenfolge gefunden.
median (X)	Gibt den Mittelwert des Feldes X zurück.
min (X)	Gibt den niedrigsten Wert des Feldes X zurück. Falls die Werte von X nicht numerisch sind, wird das Minimum in der lexikographischen Reihenfolge gefunden.
mode (X)	Gibt den häufigsten Wert des Feldes X zurück.
perc<X> (Y)	Gibt die Xte Perzentile des Feldes Y zurück. Z.B. perc5(total) gibt die 5te Perzentile (5% Grenze) des Feldes „total" zurück.
range (X)	Gibt die Differenz zwischen den minimalen und maximalen Werten des Feldes X zurück.
stdev (X)	Gibt die Standardabweichung des Feldes X zurück.
stdevp (X)	Gibt die Gesamtheit der Standardabweichungen des Feldes X zurück.
sum (X)	Gibt die Summe aller Werte des Feldes X zurück.
sumsq (X)	Gibt die Summe aller Quadrate der Werte des Feldes X zurück.
values (X)	Gibt eine Liste aller eindeutigen Werte des Feldes X als einen Multiwerteintrag zurück. Die Reihenfolge der Werte ist lexikographisch.
var (X)	Gibt die Musterabweichung des Feldes X zurück.

Suchbeispiele

Ergebnisse Filtern

Ergebnisse filtern die explizit "fail" in ihrem Text beinhalten status=0 sind.	... | *search fail status=0*
Duplikate der Ergebnisse mit demselben Host filtern.	... | *dedup host*
Nur solche Ergebnisse erfassen dessen Feld "_raw" non-routable IP Adressen der Klasse A (10.0.0.0/8) beinhalten.	... | *regex _raw="(?<!\d)10.\d{1,3}\.\d{1,3}\.\d{1,3}(?!\d)"*

Ergebnisse Gruppieren

Ergebnisse bündeln, nach "cluster_count" Werten sortieren und danach die 20 größten Cluster (in Datengröße) zurückgeben.	... | *cluster t=0.9 showcount=true | sort limit=20 -cluster_count*
Ergebnisse mit demselben "host" und "cookie" gruppieren, die innerhalb 30 Sek. nacheinander auftauchen und keine Pause von mehr als 5 Sek. zwischen jedem Ereignis aufweisen, und zu einer Transaktion zusammenfassen.	... | *transaction host cookie maxspan=30s maxpause=5s*
Ergebnisse mit derselben IP-Adresse (clientip) gruppieren bei denen das erste Ergebnis "signon" und das letzte Ergebnis "purchase" beinhaltet.	... | *transaction clienttip startswith="signon" endswith="purchase"*

Ergebnisse Anordnen

Gib die ersten 20 Ergebnisse zurück.	... | *head 20*
Drehe die Reihenfolge eines Ergebnisses um.	... | *reverse*
Sortiere Ergebnisse nach IP Werten in aufsteigender Reihenfolge und danach nach "url" Werten in absteigender Reihenfolge.	... | *sort ip, -url*
Gib die letzten 20 Ergebnisse zurück, in umgekehrter Reihenfolge.	... | *tail 20*

Reporting

Gibt Ereignisse zurück mit ungewöhnlichen Werten.	... | *anomalousvalue action=filter ptrresh=0.02*
Gibt das Maximum von "delay" nach "size" zurück, wobei "size" runtergebrochen wird auf ein Maximum von 10 gleichgroßen Buckets.	... | *chart max(delay) by size bins=10*
Gibt max(delay) für jeden Wert von foo gesplittet nach den Werten von bar.	... | *chart max(delay) over foo by bar*
Gibt max(delay) für jeden Wert von foo zurück.	... | *chart max(delay) over foo*
Entfernt alle numerischen, randständigen Werte.	... | *outlier*
Entfernt Duplikate von Ergebnissen mit denselben "host" Werten und gibt die Gesamtzahl der verbleibenden Ergebnisse zurück.	... | *stats dc(host)*
Gib den Durchschnitt jeder Stunde zurück, für jedes einheitliche Feld das mit dem String "lay" endet. z.B. delay, xdelay, relay, usw.)	... | *stats avg(*lay) by date_hour*
Berechne jede Minute den durchschnittliche "CPU" Wert pro "host".	... | *timechart span=1m avg(CPU) by host*
Erzeuge ein "timechart" für den Zähler von "web" Quellen nach "host".	... | *timechart count by host*
Gib die 20 häufigsten Werte für das Feld "url" zurück.	... | *top limit=20 url*
Gib die seltensten Werte für das Feld "url" zurück.	... | *rare url*

Felder hinzufügen

Setze Geschwindigkeit auf Distanz/Zeit.	... | *eval velocity=distance/time*
Extrahiere "from" und "to" Felder durch reguläre Ausdrücke. Falls ein Ereigniss "From: Susan To: David" enthält, dann sind from=Susan und to=David.	... | *rex field=_raw "From: (?<from>.*) To: (?<to>.*)"*
Speichere das laufende Total von "count" in einem Feld mit Namen "total_count".	... | *accum count as total_count*
Für jedes Ereignis in dem "count" vorkommt soll die Differenz zwischen count und seinem vorherigen Wert berechnet und gespeichert werden im Ergebnis "countdiff".	... | *delta count as countdiff*

Felder filtern

Erhalte die "host" und "ip" Felder und stelle sie in die Reihenfolge: "host" + "ip".	... | *fields + host, ip*
Entferne die die "host" und "ip" Felder.	... | *fields - host, ip*

Felder verändern

Benenne das Feld "ip" um in "IPAddress".	... | *rename _ip as IPAddress*
Ändere jeden "host" Wert der mit "localhost" endet in "mylocalhost" um.	... | *replace *localhost with mylocalhost in host*

Multiwertfelder (Multi-Valued)

Kombiniere die multiplen Werte des Empfängerfeldes in einen einzelnen Wert.	... | *nomv recipients*
Separiere die Werte des "recipient" Feldes in Multifeldwerte und stelle die Top-Empfänger dar.	... | *makemv delim=", " recipients | top recipients*
Erzeuge neue Ergebnisse für jeden Wert des Multiwertfeldes "recipient".	... | *mvexpand recipients*
Kombiniere jedes identische Ergebnis, außer für RecordNumber, und bestimme RecordNumber zu einem Multiwertfeld mit allen variierenden Ergebnissen.	... | *fields EventCode, Category, RecordNumber | mvcombine delim", " RecordNumber*
Finde die Zahl der Empfängerwerte.	... | *eval to_count = mvcount(recipients)*
Finde die erste E-Mail Adresse im Empfängerfeld.	... | *eval to_first = mvindex(to, 0)*
Finde alle Empfängerwerte die mit .net oder .org enden.	... | *eval netorg_recipients = mvfilter(match(recipient, "\.net$") OR match(recipient, "\.org$"))*
Finde die Kombination der Werte von foo, "bar" und baz	... | *eval newval = mvappend(foo, "bar", baz)*
Finde den Index des ersten Empfängerwertes der auf "\org$" zutrifft.	... | *eval orgindex = mvfind(recipient," "\.org$")*

Lookup Tabellen

Schlage den Wert für das "user" Feld jedes einzelnen Ereignisses in der Lookup-Tabelle 'usertogroup' nach und besetze damit das "group" Feld des Ereignisses.	... | *lookup usertogroup user output group*
Schreibe das Suchergebnis in die Lookup-Datei "users.csv".	... | *outputlookup users.csv*
Lese in der Lookup-Datei "users.csv" als Suchergebnis	... | *inputlookup usres.csv*

Reguläre Ausdrücke (REGEXES)

REGEX	Bemerkung	Beispiel	Erklärung
\s	Leerzeichen	**\d\s\d**	Ziffer Whitespace Ziffer
\S	Kein Leerzeichen	**\d\S\d**	Ziffer Kein Whitespace Ziffer
\d	Ziffer	**\d\d\d-\d\d-\d\d\d\d**	Sozialversicherungsnummer (US)
\D	Keine Ziffer	**\D\D\D**	Drei Nicht-Ziffern
\w	Wortzeichen (Buchstabe, Zahl, oder _)	**\w\w\w**	Drei Wortzeichen
\W	Kein Wortzeichen	**\W\W\W**	Drei Nicht-Wortzeichen
[. . .]	sämtliche eingeschlossene Buchstaben	**[a-z0-9#]**	Jedes Zeichen von a - z und 0 - 9 oder #
[^ . . .]	keine eingeschlossenen Buchstaben	**[^xyz]**	Jedes Zeichen ausser x, y, z
*	Zero oder mehr	**\w***	Keines oder mehrere Wortzeichen
+	Eins oder mehr	**\d+**	Integer
?	Zero oder Keines	**\d\d\d-?\d\d-?\d\d\d\d**	Sozialversicherungsnummer (US) mit optionalen Trennzeichen (-)
|	Oder	**\w|\d**	Wort-Zeichen oder Ziffer-Zeichen
(? P <var> . . .)	Benannte Extraktion	**(?P<ssn>\d\d\d-\d\d-\d\d\d\d)**	Suche eine Sozialversicherungsnummer (US) und füge sie dem Feld "ssn" hinzu
(? : . . .)	Logische Gruppierung	**(?:\w|\d) | (?:\d|\w)**	Ein Wortzeichen und danach eine Ziffer OR (oder) Ziffer und dann Wortzeichen
^	Beginn einer Zeile	**^\d+**	Zeile beginnt mit einer Ziffer
$	Ende einer Zeile	**\d+$**	Zeile endet mit einer Ziffer
{ . . . }	Anzahl der Wiederholungen	**\d{3,5}**	Zwischen drei und fünf Ziffern
\	Ausnahme (Escape)	**\[**	Beachte das Zeichen [
(? = . . .)	Vorgriff	**(?=\D) (?P . . .)**	Der Extraktion muss ein Nicht-Ziffer Zeichen vorangehen
(? ! . . .)	Negativer Vorgriff	**(?!\D) (?P . . .)**	Die Extraktion kann nicht von einem Nicht-Ziffer Zeichen angeführt werden

Gebräuchliche strptime Formate

strptime Formate sind hilfreich für die Eval-Funktionen strftime() und strptime(), sowie für das Setzen von Zeitstempeln der Ereignisdaten.

Zeit	**%H**	24 Stunden (führende Nullen) (00 - 23)
	%I	12 Stunden (führende Nullen) (01 - 12)
	%M	Minute (00 - 59)
	%S	Sekunde (00 - 59)
	%N	Subsekunden mit Groesse (%3N = Millisekunden, %6N = Mikrosekunden, $9N = Nanosekunden)
	%p	AM oder PM
	%Z	Zeitzone (GMT)
	%s	Sekunden seit 01/01/1970 (1308677092)
Tage	**%d**	Tage des Monats (führende Nullen) (01 - 31)
	%j	Tage des Jahres (001 - 366)
	%w	Wochentag (0 - 6)
	%a	Abgekürzter Wochentag (Son)
	%A	Wochentag (Sonntag)
Monate	**%b**	Abgekürzter Monatsname (Jan)
	%B	Montasname (Januar)
	%m	Monatsnummer (01 -12)
Jahre	**%y**	Jahr ohne Jahrhundert (00 - 99)
	%Y	Jahr (2011)
Beispiele	**%Y - %m - %d**	2011-12-31
	%y - %m - %d	11-12-31
	%b %d, %Y	Jan 24, 2011
	%B %d, %Y	Januar 24, 2011
	q | %d %b ,%y = %Y - %m -%d |	q | 31 Dez'11 = 2011-12-31|

splunk>

Splunk Inc.
250 Brannan Street
San Francisco, CA 94107

www.splunk.com