Splunk Cloud™ Search Manual 8.2.2104

About transactions

Generated: 6/27/2021 4:12 am

About transactions

A **transaction** is any group of conceptually-related events that spans time, such as a series of events related to the online reservation of a hotel room by a single customer, or a set of events related to a firewall intrusion incident. A **transaction type** is a configured transaction, saved as a field and used in conjunction with the `transaction` command. Any number of data sources can generate transactions over multiple log entries.

Transaction search

A transaction search is useful for a single observation of any physical event stretching over multiple logged events. Use the transaction command to define a transaction or override transaction options specified in `transactiontypes.conf`.

One common use of a transaction search is to group multiple events into a single meta-event that represents a single physical event. For example, an **out of memory problem** could trigger several database events to be logged, and they can all be grouped together into a transaction.

To learn more, see Identify and group events into transactions in this manual.

Transaction search example

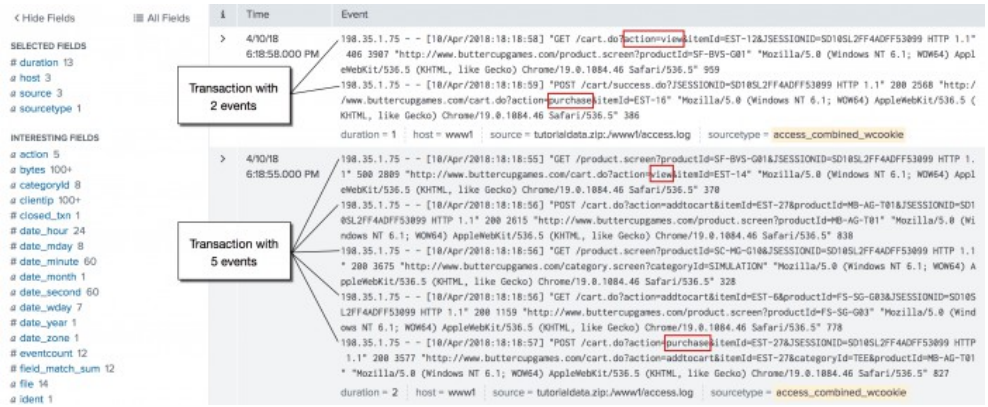
> This example uses the sample data from the Search Tutorial but should work with any format of Apache web access log. To try this example on your own Splunk instance, you must download the sample data and follow the instructions to **get the tutorial data into Splunk**. Use the time range **All time** when you run the search.

This example searches for transactions with the same session ID and IP address. This example defines a transaction as a group of events that have the same session ID, `JSESSIONID`, and come from the same IP address, `clientip`, and where the first event contains the string, "view", and the last event contains the string, "purchase".

```
sourcetype=access_* | transaction JSESSIONID clientip startswith="view" endswith="purchase" | where duration>0
```

The search defines the first event in the transaction as events that include the string, "view", using the `startswith="view"` argument. The `endswith="purchase"` argument does the same for the last event in the transaction.

This example then pipes the transactions into the `where` command and the `duration` field to filter out all of the transactions that took less than a second to complete. The `where` filter cannot be applied before the `transaction` command because the `duration` field is added by the `transaction` command. The values in the `duration` field show the difference, in seconds, between the timestamps for the first and last events in the transaction.



You might be curious about why the transactions took a long time, so viewing these events might help you to troubleshoot. You won't see it in this data, but some transactions may take a long time because the user is updating and removing items from his shopping cart before he completes the purchase. Additionally, this search is run over all events. There is no filtering before the `transaction` command. Anytime you can filter the search before the first pipe, the faster the search runs.

For more examples, see the transaction command.

Using stats instead of transaction

Both the stats command and the transaction command are similar in that they enable you to aggregate individual events together based on field values.

The `stats` command is meant to calculate statistics on events grouped by one or more fields and discard the events (unless you are using eventstats or streamstats). On the other hand, except for the duration between first and last events and the count of events, the `transaction` command does not compute statistics over the grouped events. Additionally, it retains the raw event and other field values from the original event and enables you to group events using much more complex criteria, such as limiting the grouping by time span or delays and requiring terms to define the start or end of a group.

The `transaction` command is most useful in two specific cases:

1. When a unique ID (from one or more fields) alone is not sufficient to discriminate between two transactions. This is the case when the identifier is reused, for example web sessions identified by cookie or client IP. In this case, time spans or pauses are also used to segment the data into transactions. In other cases, when an identifier is reused, for example in DHCP logs, a particular message may identify the beginning or end of a transaction.

2. When it is desirable to see the raw text of the events combined rather than an analysis on the constituent fields of the events.

In other cases, it's usually better to use the `stats` command, which performs more efficiently, especially in a distributed environment. Often there is a unique ID in the events and `stats` can be used.

For example, to compute the statistics on the duration of trades identified by the unique ID `trade_id`, the following searches will yield the same answer:

```
... | transaction trade_id | chart count by duration span=log2
```

and

```
... | stats range(_time) as duration by trade_id | chart count by duration span=log2
```

If however, the `trade_id` values are reused but each trade ends with some text, such as "END", the only solution is to use this `transaction` search:

```
... | transaction trade_id endswith=END | chart count by duration span=log2
```

On the other hand, if `trade_id` values are reused, but not within a 10 minute duration, the solution is to use the following `transaction` search:

```
... | transaction trade_id maxpause=10m | chart count by duration span=log2
```

Read more about "About event grouping and correlation" in an earlier chapter in this manual.

Transactions and macro search

Transactions and macro searches are a powerful combination that allow substitution into your transaction searches. Make a transaction search and then save it with `$field$` to allow substitution.

For an example of how to use macro searches and transactions, see Define and use search macros in the *Knowledge Manager Manual*.