parse-field-option.md

github.com/SumoLogic/sumologic-documentation/blob/7b53abb04b0e84a5189b3ef4320a95c27c2a5a2f/docs/search/search-query-language/parse-operators/parse-field-option.md

kimsauce

```
---
id: parse-field-option
title: Parse field option
---
```

Sumo Logic allows you to parse on previously extracted fields, or initial parsing on a metadata field value (`_collector`, `_source`, etc.) using the additional parse syntax of `field<field_name>`.

This additional syntax is available with the standard [Parse Anchor](/docs/search/search-query-language/parse-operators/parse-predictable-patterns-using-an-anchor) as well as the [Parse Regex](parse-variable-patterns-using-regex.md) operations.

Syntax

* `parse field=<field> "<start_anchor>*<stop_anchor>" as <field>`

* `parse regex field=<field> "<start expression>(?<fieldname><field expression>)<stop expression>"`

:::note

Characters quoted with double quotes (not single quotes) are string literals. Use a backslash to escape double quotes in the string. For example:

```sql

| parse field=input "\"tier\" : *," as tier

```

:::

Examples

Sample log message:

```

Aug 2 04:06:08: host=10.1.1.124: local/ssl2 notice mcpd[3772]: User=jsmith@demo.com: severity=warning: 01070638:5: Pool member 172.31.51.22:0 monitor status down.

```

First, use a parse statement such as the following to get the User from the log message, which will return a field called `user_email` with a value of `jsmith@demo.com`:

```sql

parse "User=*:" as user_email

```

Now that we have this field, we want to additionally parse out just the name and domain from the email address. We can do this by adding the additional syntax of `field<field_name>` to a follow-up parse operation:

```sql

parse "User=*:" as user_email

| parse field=user_email "*@*" as user_name, domain

```

The result of the above query would be:

user_email	user_name	domain
jsmith@demo.com	jsmith	demo.com

The `field<field_name>` syntax is not just limited to fields that have been specifically parsed from the logs. This syntax can also be used to parse the predefined metadata fields such as `_collector`, `_source`, `_sourceName`, etc. For example, if we have a long list of Collectors all with the same naming format of HostName_10.10.10.1 we can parse this metadata field value to just get the IP address.

```sql

parse field=_collector "HostName_*" as host_ip

```