Mask Rules

A mask rule is a type of processing rule that hides irrelevant or sensitive information from logs before ingestion. When you create a mask rule, whatever expression you choose to mask will be replaced with a mask string before it is sent to Sumo Logic. You can provide a mask string, or use the default, \\#.

Ingestion volume is calculated after applying the mask filter. If the mask reduces the size of the log, the smaller size will be measured against ingestion limits. Masking is a good method for reducing overall ingestion volume.

The mask string does not support the use of colon : characters.

For example, to mask the email address dan@demo.com from this log:

```
2018-05-16 09:43:39,607 -0700 DEBUG [hostId=prod-cass-raw-8] [module=RAW] [logger=scala.raw.InboundRawProtocolHandler] [auth=User:dan@demo.com] [remote_ip=98.248.40.103] [web_session=19zefhqy...] [session=80F1BD83AEBDF4FB] [customer=0000000000000005] [call=InboundRawProtocol.getMessages]
```

You could use the following filter expression:

```
auth=User:(.*\.com)\]s
```

With a mask string of AAA would provide the following result:

```
2018-05-16 09:43:39,607 -0700 DEBUG [hostId=prod-cass-raw-8] [module=RAW] [logger=scala.raw.InboundRawProtocolHandler] [auth=AAA] [remote_ip=98.248.40.103] [web_session=19zefhqy...] [session=80F1BD83AEBDF4FB] [customer=0000000000000005] [call=InboundRawProtocol.getMessages]
```

Rules

- Expressions that you want masked must be expressed as a capture group in the regex. Capture groups are identified with enclosed parentheses ()

For example, this log message:

```
{
  "reqHdr":{
    "auth":"Basic ksoe9wudkej2lfj*jshd6sl.cmei=",
    "cookie":"$Version=0; JSESSIONID=6C1BR5DAB897346B70FD2CA7SD4639.localhost_bc;
```

Copyright 2019 Sumo Logic, Inc. All rights reserved.
You would use the following as a mask expression to mask the auth parameter’s token:

```regex
auth\s*:\s*"Basic\s*([\^]+)"
```

- Do not unnecessarily match on more of the log than needed. From the previous example, don't use the following expression as it matches more than necessary,

```regex
(?s).*auth\s*:\s*"Basic\s*([\^]+)".*(?s)
```

- You can use an anchor to detect specific values. For example, if in your logs all user emails can be identified in logs as `User:(user@email.com)` you could use `User:([^)]*)` so that `User:` is the starting anchor and `)` is the ending anchor. The capturing group `([^)]*)` matches anything between these anchors and will mask it.

- You can specify multiple capture groups. Note that if multiple capture groups are specified in one filter, each value will be masked in the same way. So if you create one filter for users’ email addresses and IP addresses both will be replaced with the same mask string. For example, from the example logs above the following regex filter has two capture groups, one for the email address and another for the session ID:

```regex
auth=User\:(.*)\}\s.*\[session=(.*?)]\s
```

- Each capture group matches and masks all occurrences that exist in each log.

- If you'd like to use a different mask for each value, you'll need to create a separate mask rule for each value. For example, if you'd like to mask IP addresses with a string that's different from the user email string, you'd create another filter with the expression `\([d1,3]\.[d1,3]\.[d1,3]\)` and you could use `USER_ADDRESS` as the mask string.

- Make sure you don't specify a regular expression that has a capturing group that matches a full log line. Doing so will result in the entire log line being masked.

- If you need to mask values on multiple lines use single line modifiers (?s). For example:

```regex
auth=User\:\([^)]*\)\]\s.*\[session=(.*)\]\s
```

- A mask string exceeding five times the size of the original string will be trimmed. For example, if a mask rule increases the size from 7 bytes to 38 bytes, it will be trimmed to 35 bytes (which is 5 times 7 bytes), where the last 3 bytes of the message are dropped.

---

**Examples**

Any masking expression should be tested and verified with a sample source file before applying it to your production logs.
**Mask Social Security numbers**

Sumo Logic allows for masking SSN from log messages using a regular expression within a mask rule. Once masked with a known string, you can then perform a search for that string within your logs to detect if SSN may be leaking into your log files.

The following regular expression can be used within a masking filter to mask SSN before your log data is sent to Sumo Logic.

```
(!\219-09-9999|078-05-1120) (?!666|000|9\d{2})\d{3}-(?!00)\d{2}-(?!0{4})\d{4}
```

**Mask credit card numbers**

You can mask credit card numbers from log messages using a regular expression within a mask rule. Once masked with a known string, you can then perform a search for that string within your logs to detect if credit card numbers may be leaking into your log files.

The following regular expression can be used within a masking filter to mask American Express, Visa (16 digit only), Master Card, and Discover credit card numbers:

```
(?!\4\d{3})|(?!\5[1-5]\d{2})|6(?!011|5[0-9]{2})(?!-?\40?)|(?:(?:4\d{3})|(?:5\[1-5]\d{2})|6(?:011|5[0-9]{2}))\d{3}-(?!00)\d{2}-(?!0{4})\d{4}
```

This regular expression covers instances where the number includes dashes, spaces, or is a solid string of numbers.

**Samples include:**

- **American Express:** 3711-078176-01234 | 371107817601234 | 3711 078176 01234
- **Visa:** 4123-5123-6123-7123 | 4123512361237123 | 4123 5123 6123 7123
- **Master Card:** 5123-4123-6123-7123 | 5123412361237123 | 5123 4123 6123 7123
- **Discover:** 6011-0009-9013-9424 | 6500000000000002 | 6011 0009 9013 9424