## Logagent FAQ

### Is there a verbose / debug mode and how is it enabled?

Yes. Add the following property to your pattern definition file (patterns.yml):

```
debug: true
```

You could also create a file containing only the debug setting and load it via command line.

```
echo "debug: true" > debug-enable.yml
logagent -f patterns.yml -f ./debug-enable.yml  -g '/var/log/**/*.log'
```

When debug is enabled, Logagent will print every pattern match, e.g.:

```
Pattern match: log #4 /^([\w|\s]+\s\d{2}\s[\d|\:|\.]+)\s+(<.+?>)\s(.*)/ ["ts","service","mess
Pattern match: system_log #3 /^([\w|\s]+\s+\d{1,2}\s[\d|\:|\.]+)\s+(\S+)\s+(.*)\:\s(.*)/ ["ts
Pattern match: system_log #3 /^([\w|\s]+\s+\d{1,2}\s[\d|\:|\.]+)\s+(\S+)\s+(.*)\:\s(.*)/ ["ts
{"logSource":"/var/log/wifi.log","_type":"log","service":"<kernel>","message":"IO80211Interf
```

To load multiple pattern files make sure the pattern file with the the debug option enabled is the last file loaded because each loaded config could overwrite settings of the previously loaded pattern files:

```
echo "debug: true" > debug-enable.yml
logagent -f patterns.yml -f ./debug-enable.yml  -g '/var/log/**/*.log'
```

### Why does Logagent use stderr for its own logs?

Logagent can be used as a command line tool with other Linux tools. It can read data from stdin and output processed data to stdout. Logagent writes its own log messages to stderr in order to avoid any interference with data processing pipeline.

Plugin developers should use the console.error function for logs produced by the plugin itself.

### Where are Logagent's own logs?

When Logagent is installed as a service, Logagent log files are captured by upstart or systemd.

- systemd - `journalctl -u logagent`
- upstart - `/var/log/upstart/logagent`
- Mac OS X / launchd - `/Library/Logs/logagent.log`
- docker - `docker logs container-name`
- Windows - Windows does not capture stderr stream of services

### Where are Logagent service scripts and how to restart the service?

Location of service scripts:

- upstart: /etc/init/logagent.conf
- systemd: /etc/systemd/system/logagent.service
- launchd: /Library/LaunchDaemons/com.sematext.logagent.plist

Restart Logagent service:

- upstart: `service logagent restart`
- systemd: `systemctl stop logagent && systemctl start logagent`
- launchd: `launchctl stop com.sematext.logagent && launchctl stop com.sematext.logagent`

Default location of Logagent service configuration file:

- Linux and Mac OS X: `/etc/sematext/logagent.conf` The location can be changed by setting the LOGAGENT_CONFIG environment variable.
- Windows: `%ProgramData%\Sematext\logagent.conf` The location can be changed with following registry key: `HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Ses Manager\Environment\LOGAGENT_CONFIG`

### How do I tail multiple files?

On the command line you could use one or more glob patterns:

```
logagent -g '/var/log/**/*.log'
logagent -g '{/var/log/**/*.log, /myapp/logs/*.log}'
```

Logagent configuration files use a list of glob patterns in the input.files section. Each glob pattern might result in watching multiple files. New files are detected automatically after periodical scans (once a minute):

```
input:
  files:
    - '/var/log/**/*.log'
    - '/myapp/logs/*.log'
    - '/opt/another-log-directory/another.log'
```

### How do I ship logs to multiple destinations / Sematext Logs apps?

Logagent supports multiple instances of output plugins (Kafka, Elasticsearch, Files, …).

The Elasticseach plugin supports routing to different indices (or Sematext Logsene Tokens), by configuring a list of patterns matching the log file name.

The following example ships logs from wireless devices and authentication log to a local Elasticsearch server and other server logs to multiple Logsene apps.

```
input:
  files:
      - '/var/log/**/*.log'

output:
  # index logs in Elasticsearch or Logsene
  local-elasticsearch:
    module: elasticsearch
    url: http://localhost:9200
   # default index to use, for all logs that don't match any other configuration
    index: other_logs
    # specific indices to use per logSource field of parsed logs
    indices:
    wireless_logs: # use regex to match log source e.g. /var/log/wifi.log
        - wifi|bluetooth
      security_logs:
        - auth\.log
   logsene-saas:
        module: elasticsearch
        url: https://logsene-receiver.sematext.com
        indices:
          bb308f80-0000-0000-894c-f80c054a0f10:
              - [nginx|httpd]\.log
          a0ca5032-0000-467d-b6d5-e465a7ce45bb
              - mysql|postgres|oracle
          969020b4-0000-0000-86e4-24e67759cdb3
              - mongo.*\.log
              - myapp1\/app.log
              - myapp2\/app.log
```

**How do I ship only error logs?**

Use the "grep" input filter:

```
input:
  files:
    - '/var/log/**/*.log'

inputFilter:
  - module: grep
    config:
      matchSource: !!js/regexp /.*log/
```

```
      include: !!js/regexp /failed|error|exception/i
      exclude: !!js/regexp /super noisy error messages/i


output:
  elasticsearch:
    module: elasticsearch
    url: https://logsene-receiver.sematext.com
    index: YOUR_LOGSENE_TOKEN_HERE
```

**How do I drop logs that match a certain pattern?**

Use the "grep" input filter:

```
input:
  files:
    - '/var/log/**/*.log'

inputFilter:
  - module: grep
    config:
      matchSource: !!js/regexp /.*log/
      include: !!js/regexp /A.*|B.*|C.*/i
      # exclude: !!js/regexp /debug/i

output:
  elasticsearch:
    module: elasticsearch
    url: https://logsene-receiver.sematext.com
    index: YOUR_LOGSENE_TOKEN_HERE
```

**How do I ship logs that match different patterns to different destinations / Sematext Logs apps?**

An output filter function could do the trick, by setting data._index field, depending on various conditions. The following example creates an output filter with a configurable field name, a regular expression to match the content of the given field, and index name for the output.

```
outputFilter:
  - config:
      fieldName: message
      includeRegex: !!js/regexp /exception|error/i
      indexName: my_index_for_errors
    module: !!js/function >>
```

```
function (context, config, eventEmitter, data, callback)  {
  if (config.includeRegex.test(data[config.fieldName])) {
    data._index = config.indexName
  }
  cb(null, data)
 }
```