# VDAB Log Data Processing

VDAB flows can be used to sample data from log file and react to errors and other data written to log files. This is done by continuously sampling what is written to the log file. Once sampled that data can be handled as either a) unstructured log lines or b) structured log records. Creating structured log records requires that the structure of the log file be defined using the *ParseLogRecord* node which creates and event that itself includes information regarding the severity and time of the logging event.

| **Before This** | You should have reviewed the VDAB introductory tutorial and documentation to understand how to create basic flows. |
|---|---|

## Related Documentation

The following document and tutorials either are a) available or b) being developed to further support this subject.

Those available are highlighted in blue while those under development are not highlighted.

| Related Guides | Details |
|---|---|
| Alerts and Notifications | This guide illustrates how to add alerts to monitoring flows and how to create flows to respond to the alerts. |

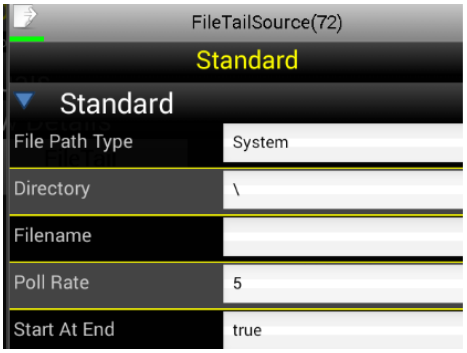
| Related Tutorial | Details |
|---|---|
| Monitoring And Alerts<br>https://vdabtec.com/vdab/tutorials | Tutorial video which demonstrates how to monitor and create alerts. This includes some more complicated examples of alerts and alert handling. |

## Sampling Unstructured Log Data

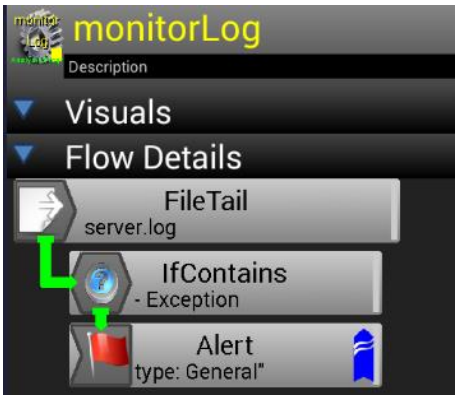
The *FileTailSource* samples all data written to a log file and creates an event for each line in the log file. Add the *IfContains* node to search for specific text data in the log record.

1. Create flow and add a *FileTailSource* node to the flow.


2. Select the edit option on the *FileTailSource* node and set the file type, directory and location. Set Start at End to true if you only wish to sample newly written lines. Set to false to replay all the lines in the file whenever the flow is started.




3. Once started the *FileTailSource* will send an event containing each line written to the file. Add and configure an *IfContains* node and *AlertTarget* node to create an Alert when specific text is found in the line. For example you can send an alert whenever the text *Exception* is written to the file. The Alert can contain the log record that required the alert. See *Alerts and Notification Guide* for details using Alerts.



## Converting to Structured Log File Data

Most log file data is written in a format that can be interpreted to identify the log time and the severity of the log record. The *ParseLogRecord* node allows the identification of the log time and severity and can create an event that included these details.

Before starting you need to determine whether you have a DELIMITED or FIXED header record structure.

The VDAB log exhibits a header structure that can be interpreted using DELIMITED option. The log header records can be easily separated by parsing between the spaces because spaces are not used within the header fields themselves.

```
09May16-20:04:58 OBJ=-1 (1) (TASK PurgeDatabase)  Did not execute properly
09May16-20:04:58 OBJ=-1 (1) (TASK UpdateEventInfoCache)  Did not execute properly
09May16-20:04:58 OBJ=-1 (1) (TASK SetContainerStats)  Executed successfully execution time 0 msecs
09May16-20:05:12 OBJ=-1 (2) (AnalysisContainer._init()) INITIALIZE CALLED
```

When the log file does not restrict the use of specific characters within the fields and the fields are in specific defined character locations, the FIXED option can be used to find the fields.

```
arks-ultra          2017-04-22 10:11:21,489 INFO ENV not specified
arks-ultra          2017-04-22 10:11:21,563 INFO Starting parse security config...
```

A parsing template for fixed files is created by copying a line from the file and replacing the first character in the fixed position field with a [ and the last position in the fixed position field with a ] .

```
[rks-ultra          ][017-04-22 10:11:2],489 [NFO]ENV not specified
```

If the log file does not have a specific reserved parsing character and does not have fixed locations for fields neither approach will work properly and it is recommended that you alter the logging structure. In logs written by log4j this can be accomplished by modifying the logging patterns to eliminate variable sized fields so the FIXED file structure parsing can be used.

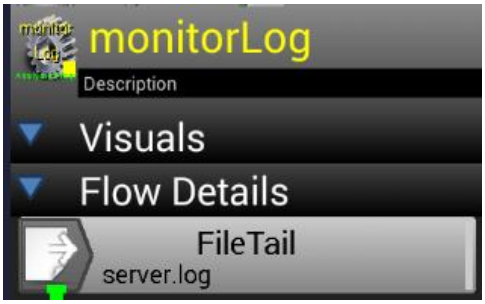
The following log4j conversion pattern includes a variable length (8 to 12 character) header field.

```
log4j.appender.RuleEngineLog.layout.ConversionPattern=%-9.9X{field1} %-8.12X{field2}
```

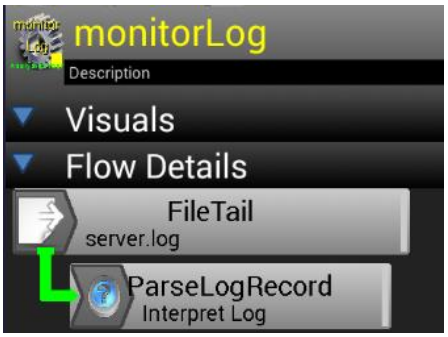
Change the pattern to move to a fixed leng (10 character) header field.

```
log4j.appender.RuleEngineLog.layout.ConversionPattern=%-9.9X{field1} %-10.10X{field2}
```

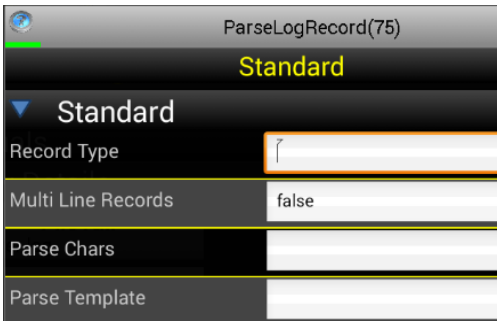
1. Create flow with *FileTailSource* node and configure the node to read a specific file.



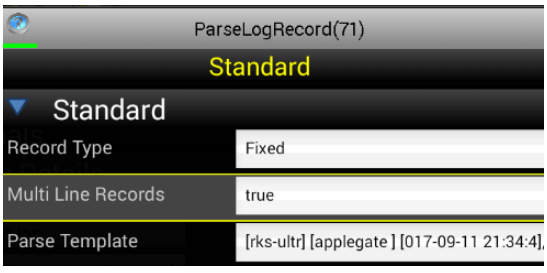

2. Add a *ParseLogRecord* node to the *FileTailSource* node. Click on this node and select the Edit option.



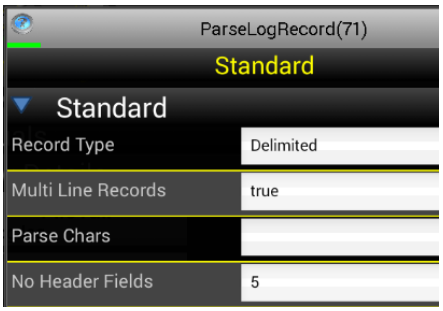

3. Start the editing by selecting either a FIXED header structure or a DELIMITED log structure.



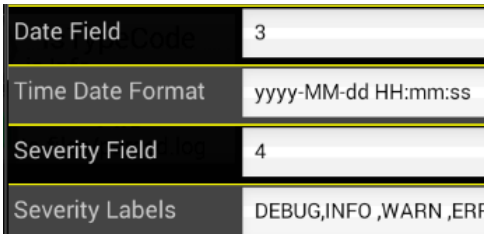

4. If you have selected a FIXED log structure create a parse template by putting a [ at the start of every fields and a ] at the end of every field. (Make sure your fields are fixed in size. See documentation on log4j patterns.)



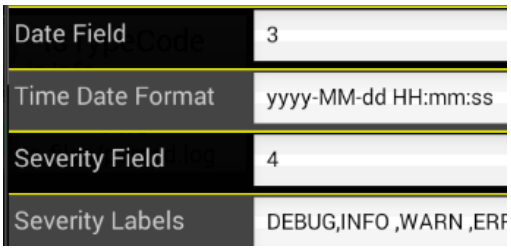
5. If you have selected a DELIMITED log structure, set the characters that will be used to parse the records (Parse Chars) and the number of fields in the header (No Header Fields).



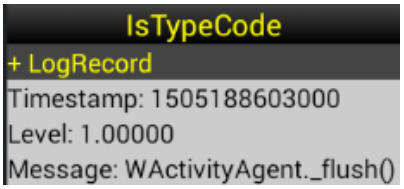

6. Identify the number of the field that contains the date information (Date Field). Field numbering starts at 1.
7. Using the JAVA data format specifications, define the date format (Time Date Format).



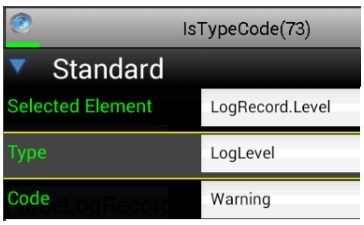

8. Identify the number of the field that contains the log record severity (Severity Field). Field numbering starts at 1.
9. Define what labels are expected for severity separated by a comma. Four severities are defined going from lowest (debug) to highest (error).




10. The *ParseLogRecord* node will output an event that includes a timestamp, severity level and the message content.




11. The log event can be filtered and handled like any other event. For example the *IsTypeCode* node can be used to select only Warning log records.

## Appendix: Documentation for Nodes

*This same information is available for all nodes by going to your local vdab web server at [http://localhost/vdab/docs_node](http://localhost/vdab/docs_node).*

### FileTailSource

Alpha

| | |
|---|---|
| FilePathType | Selects a system or VDAB file path. The VDAB file path only accesses files in the VDAB directories. |
| Directory | The directory where the file or files are located. |
| Filename | The name of the file. |
| PollRate | The frequency that data is sampled in seconds |
| StartAtEnd | If set to true, only data added to end of the file will be processed. If set to false, the data in the file will be reread from the start of the file whenever the flow is restarted. |
| Comments | Additional comments about this node, flow or container. |

### ParseLogRecord

Alpha

Parses data in a fixed or delimited log record, reading the timestamp on the record entry.

| | |
|---|---|
| RecordType | Selects either a FIXED or DELIMITED record format. FIXED records need to have a template defined. |
| MultiLineRecords | If set some records may include more than one line. If not set only one line is parsed. |
| ParseChars | The characters that are delimiters between fields in DELIMITED headers. |
| ParseTemplate | A template used to parse the FIXED header. A square bracket marks the beginning [ and end ] of each field. |
| NoHeaderFields | Identifies the number of header fields expected. |
| DateField | The number of the field that contains the date and time. Field numbering starts at 1. |
| TimeDateFormat | The format expected for the data using the JAVA Simple Date Format format. |
| SeverityField | The number of the field that contains the severity information. Field numbering starts at 1. |
| SeverityLabels | A comma delimited set of the labels used to indicate severity in order of increasing severity. |
| Comments | Additional comments about this node, flow or container. |

### IsTypeCode

Beta

Compares an incoming VDAB datatype code with a specific code of for that type.

| | |
|---|---|
| SelectedElement | Select the data element that should be operated on in this node |
| Type | The VDAB datatype that will be checked for a code match. |
| Code | The code associated with the VDAB datatype that will be tested |
| OutputType | The type of output that will published by this node. Supported types include Complete Events,Triggers and a Boolean value |
| Comments | Additional comments about this node, flow or container. |