# Logging Enhancements Project Proposal



## Project Name

- Proposed name for projects: **Logging and Analytics**
- Proposed name for the repository: **logging-analytics**

## Why a Logging Project

Keywords:

Centralized, Role based access, tracking, streaming, reporting

| Feature / Requirement | Description |
|---|---|
| F1 | Centralized Logs - all the logs available streamed in one place - search, visualize and report ready |
| F2 | Role based access - logs are accessible from outside the container - no ssh into the pod or tailing of the logs per microservice |
| F3 | Tracking - If a unique ID is passed across microservices we can trace a transaction pass/failure through the system<br>requestID, InvocationID, Timestamp indexed |
| F4 | Reporting - track multiple transaction patterns and generate emergent or correlated behavior |
| F5 | Control over log content - collect, index, filter |
| F6 | Machine readable json oriented elasticsearch storage of logs |
| | |
| R1 | Logs are in the same (currently 29 field) format |
| R2 | Logging library requires minimal changes for use - using Spring AOP to get MARKER entry/exit logs for free without code changes |
| R3 | Logs are streamed to a central ELK stack |
| R4 | ELK stack provides for tracing, dashboards, query API |

## Project Description

ONAP consists of many components and containers, and consequently writes to many logfiles. The volume of logger output may be enormous, especially when debugging. Large, disparate logfiles are difficult to monitor and analyze, and tracing requests across many files, file systems and containers is untenable without tooling.

The problem of decentralized logger output is addressed by analytics pipelines such as Elastic Stack (ELK). Elastic Stack consumes logs, indexes their contents in Elasticsearch, and makes them accessible, queryable and navigable via a sophisticated UI, Kibana Discover. This elevates the importance of standardization and machine-readability. Logfiles can remain browsable, but output can be simplified.

Logger configurations in ONAP are diverse and idiosyncratic. Addressing these issues will prevent costs from being externalized to consumers such as analytics. It also affords the opportunity to remedy any issues with the handling and propagation of contextual information such as transaction identifiers (presently passed as **X-ONAP-RequestID**. This propagation is critical to tracing requests as they traverse ONAP and related systems, and is the basis for many analytics functions.

Rationalized logger configuration and output also paves the way for other high-performance logger transports, including publishing directly to analytics via SYSLOG (RFC3164, RFC5425, RFC5426) and streams, and mechanisms for durability.

Each change is believed to be individually beneficial:

1. The intention is to consolidate the required setup within this project however some changes and bug fixes might have to be applied in the relative component project, requiring each components' co-operation on the contribution.
2. There is an economy of scale if everything can happen under a single remit.
3. Standardization benefits all, including those who want to deviate from defaults.

# Scope

In scope:

1. ONAP-wide changes to regularize logger providers, logger configuration and logger output. The changes are largely cosmetic, but greater coherence will simplify deployment orchestration and customization, and improve extensibility and support for analytics.
2. A reference analytics pipeline configuration consisting of:
    - Filebeat shipping
    - Logstash indexing
    - Elasticsearch datastore
    - Kibana Discover UI
    - Prometheus Metrics capture
3. Documentation:
    - Configuration
    - Operations
    - Updates to Active Logging Specifications
4. Other example configurations, including:
    - JSON output.
    - Non-file transports, including SYSLOG and TCP.
    - Durable shipping transports, such as Logstash persistent queues, or Kafka or Redis or similar if there's interest.
    - Kibana Dashboards.
5. Audit:
    a. Specifically Post orchestration model based audit - Logging Scope Change for POMBA seed code

Out of scope:

- No impact on EELF or its use. Some ONAP components log via EELF, and some do not. This won't change.
- No prescription of logging providers. Standardization is encouraged but not mandatory. Note that certain providers (e.g. legacy Log4j 1.X) may not support all output options.
- No changes to any ONAP component without consultation with its owners. Note that components whose provider configuration is NOT aligned may not have their logs indexed without (potentially costly) ad hoc indexing configuration.

---

# Deliverables

## Standardization

All changes *augment* ONAP application logging guidelines on the ONAP wiki.

Commits to 2 repos

https://gerrit.onap.org/r/#/q/status:merged+project:+logging-analytics
https://gerrit.onap.org/r/#/q/status:merged+project:+oom

### Default logfile locations

All logfiles to default to beneath **/var/log**, and beneath **/var/log/ONAP** in the case of core ONAP components.

All logger provider configuration document locations namespaced by component and (if applicable) subcomponent by default:

```
/var/log/ONAP/<component>[/<subcomponent>]/*.log
```

- **Currently**: varies (though a default is loosely prescribed in ONAP application logging guidelines).
- **Affects**: most ONAP components, superficially.
- **Priority**: HIGH.
- **Why**:
  - Reduced complexity.
  - Fewer special cases:
    - Easier configuration.
    - Easier indexing.
    - Easier orchestration.
- **Notes**:
  - These paths may be on a mounted persistent volume so that:
    - Logs are accessible to shippers such as Filebeat or Fluentd.
    - Logs are persistent, and independent of the lifecycle of individual containers.
  - Any indirection for multiple instances of a container writing to a shared volume can be managed in one of two ways:
    - By including a container identifier in the above spec.
    - By deployment automation; in volume mappings, so that tenant applications don't need to know.

## Default provider configuration locations

All logger provider configuration documents to default to beneath **/etc/ONAP**.

All logger provider configuration document locations namespaced by component and (if applicable) subcomponent by default:

```
/etc/onap/<component>[/<subcomponent>]/<provider>.xml
```

- **Currently**: varies (no default prescribed by ONAP application logging guidelines).
- **Affects**: most ONAP components, superficially.
- **Priority**: HIGH.
- **Why**:
  - Reduced complexity.
  - Fewer special cases, simplified automation.
  - Certain components are presently not configurable, e.g. Portal, for which even trivial reconfiguration of logging requires reassembly of its WAR.
  - See Support for analytics pipelines.
- **Notes**:
  - This greatly simplifies deployment automation, reconfiguration, etc.
  - These paths will normally be local to a container, not on a mounted volume, but either is fine.
  - The exact path can be fine-tuned, but:
    - The benefit of an ONAP-wide convention should be evident.
    - This corresponds to the convention implemented (updated for ONAP) by MSO and others.
    - It should be aligned with conventions for other component- and subcomponent-specific configuration documents.

## Default provider configuration

With the various simplifications discussed in this document, the logger provider configuration for most components can be rationalized.

Proposals include:

- Standardized locations, etc., see Default provider configuration locations.
- Standardized line formats, see Machine-readable output.
- Standardized rollover and retention (e.g 30 days, rollover at 50MB and at least daily, etc.)
- Standardized sifting into event files, if required.

A reductive example for the Logback provider, omitting some finer details:

```
<configuration scan="true" scanPeriod="5 seconds" debug="false">

    <property name="component" value="component1"></property>
    <property name="subcomponent" value="subcomponent1"></property>
    <property name="outputDirectory" value="/var/log/ONAP/${component}/${subcomponent}" />
    <property name="outputFilename" value="all" />

    <property name="defaultPattern" value="%nopexception%logger
    \t%date{yyyy-MM-dd'T'HH:mm:ss.SSSXXX,UTC}
    \t%level
    \t%replace(%replace(%message){'\t','\\\\t'}){'\n','\\\\n'}
    \t%replace(%replace(%mdc){'\t','\\\\t'}){'\n','\\\\n'}
    \t%replace(%replace(%rootException){'\t','\\\\t'}){'\n','\\\\n'}
    \t%replace(%replace(%marker){'\t','\\\\t'}){'\n','\\\\n'}
    \t%thread
    \t%n"/>

    <property name="queueSize" value="256"/>
    <property name="maxFileSize" value="50MB"/>
    <property name="maxHistory" value="30"/>
    <property name="totalSizeCap" value="10GB"/>

    <appender name="console" class="ch.qos.logback.core.ConsoleAppender">
        <encoder>
            <charset>UTF-8</charset>
            <pattern>${defaultPattern}</pattern>
        </encoder>
    </appender>

    <appender name="consoleAsync" class="ch.qos.logback.classic.AsyncAppender">
        <queueSize>${queueSize}</queueSize>
        <appender-ref ref="console" />
    </appender>

    <appender name="file" class="ch.qos.logback.core.rolling.RollingFileAppender">
        <file>${outputDirectory}/${outputFilename}.log</file>
        <rollingPolicy class="ch.qos.logback.core.rolling.SizeAndTimeBasedRollingPolicy">
            <fileNamePattern>${outputDirectory}/${outputFilename}.%d{yyyy-MM-dd}.%i.log.zip</fileNamePattern>
            <maxFileSize>${maxFileSize}</maxFileSize>
            <maxHistory>${maxHistory}</maxHistory>
            <totalSizeCap>${totalSizeCap}</totalSizeCap>
        </rollingPolicy>
        <encoder>
            <charset>UTF-8</charset>
            <pattern>${defaultPattern}</pattern>
        </encoder>
    </appender>

    <appender name="fileAsync" class="ch.qos.logback.classic.AsyncAppender">
        <queueSize>${queueSize}</queueSize>
        <appender-ref ref="file" />
    </appender>

    <root level="info" additivity="false">
        <appender-ref ref="consoleAsync"/>
        <!--<appender-ref ref="fileAsync"/>-->
    </root>

    <logger name="org.onap.example.component1.subcomponent2" level="debug" additivity="false">
        <appender-ref ref="fileAsync" />
    </logger>

</configuration>
```

- **Currently**: varies, but mostly based on EELF **logback.xml** at https://github.com/att/EELF/blob/master/EELF-Samples/src/main/resources/logback.xml.
- **Affects**: all ONAP components (but their configuration documents will be changing anyway, to accommodate a machine-readable new line format).
- **Priority:** HIGH.
- **Why**:

    - Reduced complexity. Fewer special cases.
    - Standardization allows dramatic simplification of *other* configuration, such as Elastic Stack shipping and indexing. Non-standard behavior externalizes costs.
    - Devolution to deployment automation and other tooling.
- **Notes**:
    - In theory provider configuration could be reduced to a single ONAP-wide document servicing all Java-based components.
    - Some suggestions in this proposal will be rejected by the ONAP community. Configuration can be updated accordingly, but the benefits of standardized configuration remain.

### Default logger provider

Encourage Logback for Java-based components, porting any components that agree to switch.

- **Currently**: Logback and Log4j.
- **Affects**:
    - The small number of ONAP components which use Log4j, and especially the smaller number which use (EOL) Log4j 1.X.
    - These include SDNC and APPC.
- **Priority**: LOW.
- **Why**:
    - Reduced complexity.
    - Log4j 1.X encoding support is limited, especially in critical areas like escaping. (Log4j 2.X is fine).
    - Fewer configuration documents to manage, fewer special cases to maintain.
- **Notes**:
    - Standardizing on a single provider within the core set of ONAP components reduces overall complexity. Logback has the critical mass.
    - Engaging more than one provider does no actual harm. What ultimately matters is the format and location of logging output, and this doesn't demand that every ONAP component uses the same provider. Consequently if some Log4J providers remain, then that's OK.
    - This also applies to components which log without EELF, e.g. directly via SLF4J. No changes proposed in this area.

### Provider configuration overrides

All configuration locations to be overrideable by a system or file property passed to (or read by) the JVM. (No specific convention suggested, because it'll only divert attention from the importance of *having* a convention).

- **Currently**: varies.
- **Affects**: most ONAP components, superficially.
- **Priority**: LOW.
- **Why**:
    - Reduced complexity.
    - Allow the management of configuration to be further devolved to deployment automation.
- **Notes**:
    - Can augment other component- or provider-specific mechanisms for locating logger configuration.
    - Doesn't have to be a system property. Approaches currently vary widely between ONAP components.
    - All components should nonetheless respond to a standard setting.

---

## Extensibility

We've already discussed key aspects of extensibility:

1. Standardized configuration, simplifying unilateral reconfiguration and automated updates.
2. Standardized output, simplifying the configuration of analytics pipelines.

### MDCs

The MDCs in ONAP application logging guidelines are closely matched to existing MDC usage in seed ONAP components, but there may be many arbitrarily many others, and there's no need for them to be listed in configuration. Itemizing them means changing all configuration documents when new MDCs are introduced.

The change is to the serialization of MDCs in the Logback provider **<pattern/>** to a single expression which emits whatever is present:

```
%replace(%replace(%mdc){'\t','\\\\t'}){'\n','\\\\n'}
```

- **Currently**: pipe-delimited, no escaping of delimiters in MDC values.
- **Affects**: all logger configuration documents, all components, all providers.
- **Priority**: HIGH.
- **Why**:
    - Adding MDCs should not require global or local logger provider configuration.
    - Delimited MDC values require logfile parsers to know the ordinal position of the MDC value of interest. (And this would change as MDCs are added).
    - Logging only MDCs that have been set may result in output that's slightly less verbose, and slightly more readable.
    - We can do much better than semantically-challenged MDCs like **CustomField1-4** and **Unused**.
- **Notes**:
    - This example also includes tab and newline replacement.

- See Machine-readable output**.**

### Markers

Markers can be used to characterize log entries. Their use is widespread, and future ONAP components and/or customizations might emit Markers with their log messages. If and when they do, we want them indexed for analytics, and like MDCs, we don't want them to need to be declared in logger provider configuration in advance.

The change is to add the serialization of Markers to **<pattern/>** in Logback provider configuration:

```
%replace(%replace(%marker){'\t','\\\\t'}){'\n','\\\\n'}
```

- **Currently:** Markers not written to ONAP logs.
- **Affects**: future ONAP and ONAP-based components.
- **Priority**: MEDIUM.
- **Why**:
  - Useful and commonplace. See https://stackoverflow.com/questions/4165558/best-practices-for-using-markers-in-slf4j-logback.
  - A more appropriate way of indicating **metric**, **audit** and other event types (though that change isn't proposed here).
  - There's a reasonable expectation on the part of current and potential ONAP developers that Markers be supported.
- **Notes**:
  - This example also includes tab and newline replacement.
  - See Machine-readable output**.**

---

## Analytics

### Support for analytics pipelines

Regularizing output locations and line format simplifies shipping and indexing of logs, and enables many options for analytics. See Default logfile locations, MDCs, Markers, Machine-readable output, and Elastic Stack reference configuration.

- **Currently**: varies.
- **Affects**: most ONAP components.
- **Priority**: HIGH.
- **Why**: that's what this whole document is about!

### Machine-readable output

Shipper and indexing performance and durability depends on logs that can be parsed quickly and reliably.

The proposal is:

- The use of tab or ASCII record separator (**0x1E**) as a delimiter. Tab is preferred.
- Escaping all messages, exceptions, MDC values, Markers, etc. to replace the delimiter. If it's tab, **\t**.
- Escaping all newlines with **\n**. This means one line per log entry.

For example (tab- and newline-separated, MDCs, a nested exception, a marker, with newlines self-consciously added between each attribute for readability):

```
org.onap.example.component1.subcomponent1.LogbackTest
\t2017-06-06T16:09:03.594Z
\tERROR
\tHere's an error, that's usually bad
\tkey1=value1, key2=value2 with space, key5=value5"with"quotes, key3=value3\nwith\nnewlines,
key4=value4\twith\ttabs
\tjava.lang.RuntimeException: Here's Johnny
\n\tat org.onap.example.component1.subcomponent1.LogbackTest.main(LogbackTest.java:24)
\nWrapped by: java.lang.RuntimeException: Little pigs, little pigs, let me come in
\n\tat org.onap.example.component1.subcomponent1.LogbackTest.main(LogbackTest.java:27)
\tAMarker1
\tmain
```

- **Currently**:
  - Newlines not escaped.
  - Log entry attributes pipe-delimited.
  - MDC values pipe-delimited, accessed by ordinal position.
  - Delimiters not escaped in messages, MDCs and other values.
- **Affects**: all logger provider configurations.
- **Priority**: HIGH.
- **Why**:

- Existing output line format is difficult to parse.
- Shipper configuration is simplified if logfile output is uniform.
- Indexing configuration is simplified if logfile output is uniform.
- SYSLOG and other transports obviate the need for files, but file output will likely remain the default.

## Elastic Stack reference configuration

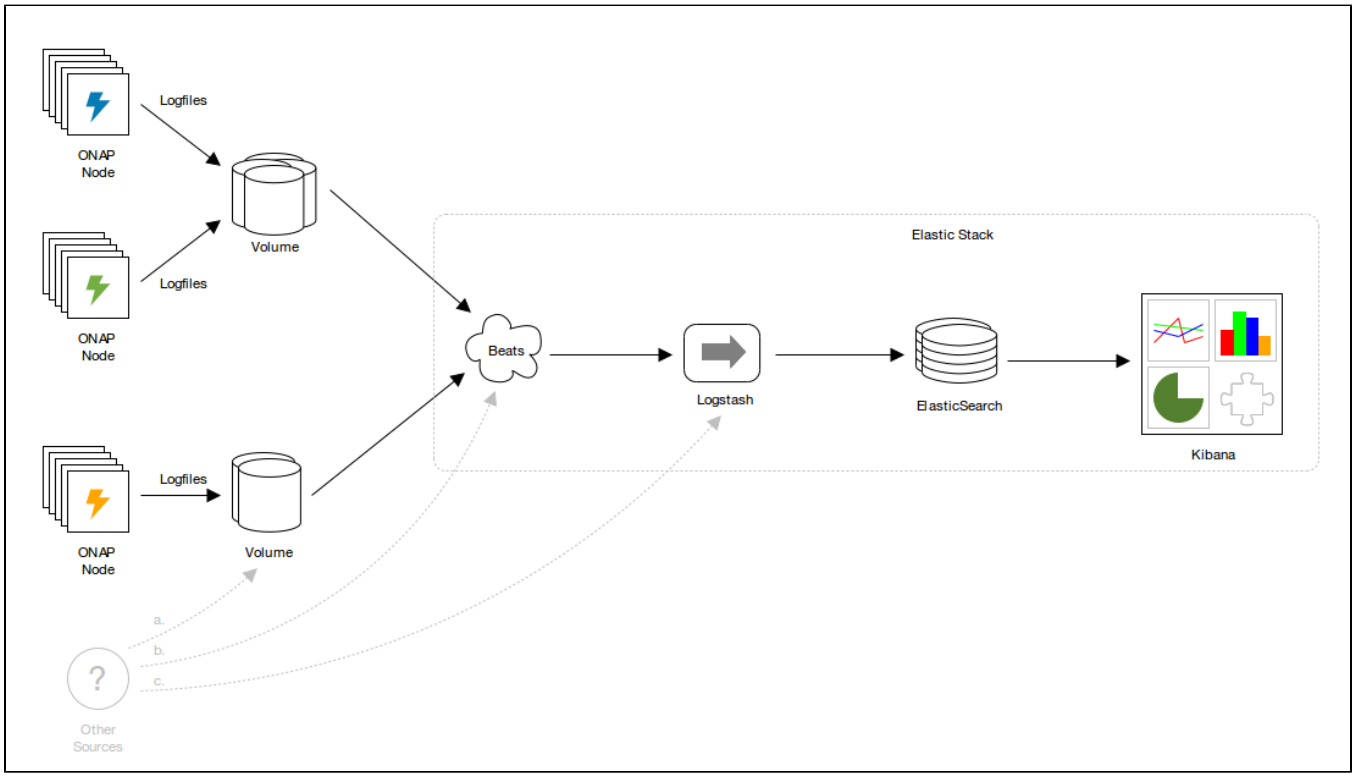
On the basis of regularized provider configuration and output, and the work of the ONAP Operations Manager project, a complete Elastic Stack pipeline can be deployed automatically.

This achieves two things:

1. It demonstrates what has been made possible.
2. It provides a ONAP with a viable analytics platform.

The proposal is:

- **Filebeat** shipping:
  - Line format and locations as discussed.
  - Other shipping transports are simplified by regularized logger provider configuration.
- **Logstash** indexing.
- **Elasticsearch** in a TBD minimal three-node cluster - currently a single container.
- **Kibana Discover** UI.
- Fullly automated Kubernetes deployment based on ONAP Operations Manager.




- **Currently**: Available via 20170914 master build
- **Affects**:
  - Adds analytics capability, but no impact on individual ONAP components.
  - Reconfiguration, including replacement with OTHER analytics pipelines will be similarly transparent.
- **Priority**: HIGH.
- **Why**:
  - Demonstrates a bolt-on analytics capability.
  - Provides a working analytics OOB.
  - Elastic Stack is available as FOSS.
- **Notes**:
  - Files + Filebeat are the lowest common denominator. They provide a simple mechanism for resilience, and yet require no disruptive (for example classpath) changes.
  - Other sources of logs can be indexed, with appropriate configuration. Generally these will not be able to be correlated with an ONAP transaction, however.

---

# Transactions

### Propagate transaction ID

Reliable propagation of transaction identifiers is critical to tracing requests through ONAP.

- **Currently**: patchy.
- **Affects**: to be determined, but at least a few components and integration points.
- **Why**: reliable propagation is a prerequisite for analytics.
- **Notes**:
    - This is normally achieved:
        - Through MDCs in-process.
        - Through REST headers between components. See Rename X-ECOMP-RequestID.
    - This also requires the generation of transaction IDs in initiating components.
    - For this to be economical, a remit is sought to make minor changes across the ONAP codebase.

### Rename X-ECOMP-RequestID

Update **X-ECOMP-RequestID** to **X-ONAP-RequestID** throughout.

- **Currently**: X-ECOMP-RequestID.
- **Affects**: all ONAP components.
- **Why**: ECOMP renamed to ONAP.
- **Notes**:
    - This may already have been fixed in another branch.
    - If not, it can be dealt with at the same time as applying changes to defaults, etc.

---

# Key Project Facts

- Repository: **logging-analytics**
- Mirror: https://github.com/onap
- Jenkins: https://jenkins.onap.org/view/logging-analytics/
- Sonar: https://sonar.onap.org/dashboard?id=org.onap.logging-analytics%3Alogging-analytics
- nexus-IQ: https://nexus-iq.wl.linuxfoundation.org/assets/index.html#/reports/logging-analytics/cead786ad2ac408c884d6cb790ae7e91
- JIRA project name: **logging-analytics**
- JIRA project prefix: **LOG**
- Mailing list tag: **log**
- Project Lead: Prudence Au / Backup/co-PTL: Luke Parker
- Committers (Name - email) Resources and Repositories (Deprecated)#LoggingEnhancements
    - Luke Parker - luke.parker@amdocs.com
    - Avdhut Kholkar - avdhut.kholkar@amdocs.com
    - Prudence Au
    - Lee Breslau - breslau@research.att.com (alumni)

- Contributors (in order of last contribution)
    - Michael O'Brien - michael@obrienlabs.cloud
    - Michael O'Brien - frank.obrien@amdocs.com
    - Steve Smokowski
    - Lorraine Welch
    - Dave Williamson
    - Shishir Thakore
    - Liang Ke
    - James MacNider
    - Shane Daniel
    - J. Ram Balasubramanian
    - Geora Barsky
    - Borislav Glozman
    - Yury Novitsky
    - Stela Stoykova
    - Alka Choudhary
    - Anup Marathe
    - Pranav Dixit
    - Vidya Shinde
    - Itay Hassid
    - Matthew Harfy
    - Karen Joseph
    -

---

# Resources

Existing logging guidelines:

- https://wiki.onap.org/download/attachments/1015849/ONAP%20application%20logging%20guidelines.pdf?api=v2

Logging providers:

- Logback: https://logback.qos.ch/
- Log4j: https://logging.apache.org/log4j/2.x/

EELF:

- https://github.com/att/EELF

Elastic Stack:

- Elastic Stack - https://www.elastic.co/products
- Filebeat - https://www.elastic.co/products/beats/filebeat
  https://www.elastic.co/videos/getting-started-with-filebeat?baymax=rtp&storm=beats&elektra=product&iesrc=ctr
- Logstash - https://www.elastic.co/products/logstash
- Kibana Discover - https://www.elastic.co/guide/en/kibana/current/discover.html
- FOSS Description for Logging Enhancements

ONAP Operations Manager: ONAP Operations Manager Project

OPNFV: https://wiki.opnfv.org/display/doctor

https://kubernetes.io/docs/concepts/cluster-administration/logging/