ONAP application[1] logging guidelines – Revision 1.0 (4/11/2017)

[1] "Application logging" refers to logs written by ONAP component "applications". In contrast, say, "system/infrastructure logging" refers to the separate/related set of logs produced by software components not developed for ONAP (e.g. DBMS, application container, web servers, 'middle boxes', JVM, OS, hypervisor, etc.) that are used in the implementation of these components.

Contents

About this document

This document specifies a common logging format to be following by all ONAP component applications. ONAP logging is intended to support operability, debugging and reporting on ONAP. It provides common logging guidelines to address:

- “occurrences” that are written by ONAP components
- platform application (i.e. component or sub-component) to a log file type
- local file system directories/names for each log file type
- log record structure

These guidelines may evolve over time.

Introduction

The purpose of ONAP logging is to capture information needed to operate, troubleshoot and report on the performance of the ONAP platform and its constituent components. Log records may be viewed and consumed directly by users and systems, loaded into a database, and used to compute metrics or KPIs about the platform and components.

The processing of, and possibly response to, a client request often involve multiple ONAP components and/or their subcomponents (interchangeably referred to as 'application' in this document). The ability to track such processing flows across components is critical to understanding ONAP's behavior and performance. ONAP logging uses a universally unique RequestID value in log records to track the processing of every single client request across all the ONAP components involved in its processing.

An application should output log records as appropriate in the following four log[2] files:

Audit Log: is required and provides a summary view of the processing of a (e.g., transaction) request within an application. It captures activity requests that are received by an ONAP component, and includes such information as the time the activity is initiated, when it finishes, and the API that is invoked at the component.

Metrics Log: is required and provides a more detailed view into the processing of a transaction within an application. It captures the beginning and ending of activities needed to complete it. These can include calls to or interactions with other ONAP or non-ONAP entities.

Error Log: is required and is intended to capture info, warn, error and fatal conditions sensed ("exception handled") by the software components.

Debug Log: is optional and is intended to capture whatever data may be needed to debug and correct abnormal conditions of the application.

Audit log records are intended to capture the high level view of activity within an ONAP component. Specifically, an API request handled by an ONAP component is reflected in a single Audit log record that captures the time the request was received, the time that processing was completed, as well as other information about the API request (e.g., API name, on whose behalf it was invoked, etc). Suboperations invoked as part of the processing of the API request are logged in the Metrics log. For example, when a call is made to another ONAP component or external (i.e., non-ONAP) entity, a Metrics log record captures that call. In such a case, the Metrics log record indicates (among other things) the time the call is made, when it returns, the entity that is called, and the API invoked on that entity. The Metrics log record contain the same RequestID as the Audit log record so the two can be correlated.

[2] "console logging" may also be present and is intended to capture "system/infrastructure" records. That is stdout and stderr assigned to a single "engine.out" file in a directory configurable (e.g. as an environment/shell variable) by operations personnel.

Note that a single request may result in multiple Audit log records at an ONAP component and may result in multiple Metrics log records generated by the component when multiple suboperations are required to satisfy the API request captured in the Audit log record.

General guidelines and conventions:

1. An application should provide the ability to change the logging level for each (all?) types of logs with no/minimum impact to the running application.

2 Application logging program logic should be modularized to facilitate least time/effort enhancement. This may be as a result of ONAP operational experience and may necessitate changes such as using a different logging infrastructure, different log record structure, swapping log4j with SLF4J, etc.

3. A configuration file should be used by logging application to determine the actual directory specification where to write the log file. Directory specifications should have the following syntax:

   <logs-directory> ::= <log-directory> "**/**" <ONAP-component-name> "**/**" [*<ONAP-subcomponent-name> "**/**"] <log-type-name>] "**.log**"

   <log-directory> ::= <debug-dir> | <log-dir> ; 1 of 2 directories

   <debug-dir>::= string directory path for debugging type logs from

   <log-directory> ::= string directory path for all/each type of application logs

   <ONAP-component-name>::= "MSO" | "DCAE" | "ASDC" | "AAI" |"Policy" | "SDNC" | "Portal" | "APPC"

   <ONAP-subcomponent-name> ::= string without reserved characters identifying the reporting entity (e.g. varying granularity such as module, class, method,…)

   <log-type-name>::= "error" | "metrics" | "audit" | "debug"

4. All application logs are sequential files with records that:

   a. consist of some number of variable length text fields that comply with left-to-right field ordering. The fields and their meanings are described below in each of the four sections pertaining to the different kind of log file.

   b. use the (reserved) "|" character as field separator/delimiter. **NB: do not use field delimiter ('|') or log record terminator ('\n') characters embedded in the field values.**

   c. use consecutive "||" character sequence to represent an unavailable, empty or not applicable field value.

5. ONAP components written in Java should use the Event and Error Logging Framework (EELF) library to write to their logs.

The following sections detail each respective log file's record format.

Audit Log

The Audit log captures the high level activities ONAP components carry out. ONAP activities are typically invoked by an API request, but may also be triggered by other actions, such as receipt of a message or execution of a cron job. An Audit log record contains timestamps to indicate when the activity begins (e.g., receipt of the API call or message) and ends, and an elapsed time. There is typically one Audit log record associated with a single API call or other triggering activity. In the case of asynchronous calls, two Audit log records are generated at different points in time. The first indicates when the activity begins and is acknowledged as received. The second record indicates when the activity is actually completed and a subsequent callback is made by the logging component in response.

When processing terminates abnormally (e.g., due to a timeout), a log record must be written indicating when processing began and when it terminated.

The format of the Audit Log is given in Table 1. Detailed description of the fields are provided in the subsequent subsections.

Ord.	Field Name	All field values are required to be supplied in every log record except where explicitly noted as "*Optional*"
1	BeginTimestamp	Date-time of the start of a request activity
2	EndTimestamp	Date-time of the end of a request activity
3	RequestID	Universally unique transaction request ID (UUID)
4	serviceInstanceID	Uniquely identifies a service instance (e.g., "service graph"). It is used as a primary key (e.g., in AAI) to reference/manage the service instance as a unit
5	threadId	*Optional*: used if wanting to trace processing of a request over a number of threads of a single ONAP component
6	physical/virtual server name	*Optional*: empty if determined that its value can be added by the agent collecting log files
7	serviceName	Externally advertised API invoked by clients of this component
8	PartnerName	Client or user invoking the API
9	StatusCode	High level success or failure of the request (COMPLETE or ERROR)
10	ResponseCode	Application-specific error code
11	Response Description	Human readable description of the application specific response code
12	instanceUUID	Universally unique identifier used to differentiate between multiple instances of the same (named) log writing component
13	Category log level	Enum: "INFO" | "WARN" |"DEBUG" | "ERROR" | "FATAL"
14	Severity	*Optional*: 0, 1, 2, 3 see Nagios monitoring/alerting for specifics/details
15	Server IP address	*Optional*: the logging component host server's IP address

16	ElapsedTime	Elapsed time to complete processing of an API or request at the granularity available to the component system. This value should be the difference between BeginTimestamp and EndTimestamp fields.
17	Server	VM FQDN if virtualized else host name of the logging component
18	ClientIPaddress	Requesting remote client application's IP address
19	class name	*Optional*: for OO programing languages that support this concept. This is the name of the class that has caused the log record to be created.
20	Unused	This field should be left blank
21	ProcessKey	Optional
22	CustomField1	Optional (specific attributes exposed by developers)
23	CustomField2	Optional (specific attributes exposed by developers)
24	CustomField3	Optional (specific attributes exposed by developers)
25	CustomField4	Optional (specific attributes exposed by developers)
26	detailMessage	*Optional*: the rightmost ("last") field in a log record. When present, its value may be formatted if/as useful to meet specific/individual use case(s).

Table 1 – ONAP Audit Log Record Structure

1. BeginTimestamp

Date-time that processing activities being logged begins. The value should be represented in UTC and formatted per ISO 8601, such as "2015-06-03T13:21:58+00:00". The time should be shown with the maximum resolution available to the logging component (e.g., milliseconds, microseconds) by including the appropriate number of decimal digits. For example, when millisecond precision is available, the date-time value would be presented as, as "2015-06-03T13:21:58.340+00:00".

This field is required.

The use of UTC avoids the need for time zone conversion when processing multiple records related to the same request that may be produced by ONAP components in different time zones.

2. EndTimestamp

Date-time that processing for the request or event being logged ends. Formatting rules are the same as for the BeginTimestamp field above.

This field is required.

In the case of a request that merely logs an event and has not subsequent processing, the EndTimestamp value may equal the BeginTimestamp value.

When processing ends abnormally due to a timeout, this field should indicate when the timeout occurred.

3. RequestID

A requestID is a universally unique value that identifies a single transaction request within the ONAP platform. Its value is conformant to RFC4122 UUID. This value is readily and easily obtained in most programming environments. The requestID value is passed using a REST API from one ONAP component to another.

It is expected that requests that originate from components/applications external to the ONAP platform (e.g., Operations Support Systems or Business Support Systems) provide this value in every distinct ONAP REST API request.

If the requestID UUID value:

1. is missing from an API call or
2. is somehow deemed unreliable or
3. the request actually originates in an ONAP component (e.g., as a consequence of some internally detected condition)

a new UUID value should be obtained by the sensing component and transmitted with any subsequent request made as part of processing that request. Generation or substitution of requestID UUID value is an action that should be logged by the effecting component. When doing so, the component would create and write an additional log record to its log files. The record would contain the component generated UUID value in the "RequestID" field and, in its "detailMessage" field (see below) one of the following, appropriate messages:

"Missing requestID. Assigned <insert new UUID here>."
or
"Replaced invalid requestID of <insert received requestID value here>."

NB:

1. Please note that the replacement value used appears in the requestID field of this log record.
2. If EELF is being used to format the detailMessage field, then the messages above should appear in a place consistent with its format**.**

What is a transaction requestID used for?

Each request along with certain related data is recorded by every component that handles it in its logs. The requestID (also referred to as a "Transaction ID") can be used to correlate and form an audit trail

from any number of log records from any number of ONAP components. This trail can be used for various purposes including:

1) enabling metrics that show how many completed requests flow through ONAP in a given period of time,
2) tracing of the flow of processing that tells what ONAP components and subcomponents are involved in its processing,
3) enabling identification and planning to determine which components represent bottlenecks (by examination of associated timestamps),
4) fault pattern detection that pinpoints the source of requests which ultimately fail somewhere in the process flow and the components in which they fail.

When collected in a central location, logs can be enriched with the identity of the creating component. That identity is obtained as the by-product of a separate log collection mechanism usually by extraction from the log file directory path. In other words, a "request processing path" can be established across any ONAP component instances that handled that request by finding and sorting in ascending timestamp order all the log records with a common requestID UUID value. Other log record fields may then be used to identify the outcome of a component's handling.

How is a transaction requestID passed from one component to another?

Even though the use of "X-" prefixed HTTP headers is deprecated (see http://tools.ietf.org/html/rfc6648), one may continue to use that format for passing the Request ID value in an HTTP request 'extended header' named "**X-ECOMP-RequestID:"** .
For example:

POST /onapcomponent/v1/api-handler HTTP/1.1
Host: host1.onap.org
X-ECOMP-RequestID: 724229c0-9945-11e5-bcde-0002a5d5c51b

How is a transaction requestID logged?

There are some cases where an ONAP (sub)component (e.g., MSO) must, for the same given transaction (i.e., requestID), make multiple, related, subsequent API (sub)requests to a second ONAP component. The (sub)request receiving component has elected, in some cases, to interpret the HTTP header as a nonce value in order to detect (and possibly reject) duplicate API requests. In these cases, the X-ECOMP-RequestID HTTP header can use a composite value of the form "**UUID:suffix**". For example

POST **/onapcomponent/v1/api-handler** HTTP/1.1
Host: **host1.onap.org**
X-ECOMP-RequestID: **8305e770-f2db-11e5-a837-0800200c9a66:1234**

When receiving a composite requestID value of the form UUID-1:UUID-2, the receiving component should only use the **UUID-1** portion (i.e., remove the "**:**" and any trailing suffix, e.g. UUID-2) as the requestID field value for its log files. Using the example immediately above, the requestID value to be used in the log record is only the "**8305e770-f2db-11e5-a837-0800200c9a66**" portion.

4. ServiceInstanceID

This field is optional and should only be included if the information is readily available to the logging component.

Transaction requests that create or operate on a particular instance of a service/resource can identify/reference it via a unique "serviceInstanceID" value. This value can be used as a primary key for obtaining or updating additional detailed data about that specific service instance from the inventory (e.g., AAI). In other words:

- In the case of processing/logging a transaction request for creating a new service instance, the serviceInstanceID value is determined by either a) the MSO client and passed to MSO or b) by MSO itself upon receipt of a such a request.

- In other cases, the serviceInstanceID value can be used to reference a specific instance of a service as would happen in a "MACD"-type request.

- ServiceInstanceID is associated with a requestID in log records to facilitate tracing its processing over multiple requests and for a specific service instance. Its value may be left "empty" in subsequent record to the 1st record where a requestID value is associated with the serviceInstanceID value.

NOTE: AAI won't have a serviceInstanceUUID for every service instance. For example, no serviceInstanceUUID is available when the request is coming from an application that may import inventory data.

5. ThreadID

Optional: used if wanting to trace processing of a request over a number of sub-components of a single ONAP component. It should be preceded by a log record that establishes its chaining back to the corresponding requestID.

6. Physical/virtual server name

Optional: empty if determined that its value can be added by the agent that collects the log files collecting.

7. serviceName

This field is required.

For Audit log records that capture API requests, this field contains the name of the API invoked at the component creating the record (e.g., Layer3ServiceActivateRequest).

For Audit log records that capture processing as a result of receipt of a message, this field should contain the name of the module that processes the message.

8. PartnerName

This field contains the name of the client application user agent or user invoking the API if known.

9. StatusCode

This field indicates the high level status of the request. It must have the value *COMPLETE* when the request is successful and *ERROR* when there is a failure.

10. ResponseCode

This field contains application-specific error codes. For consistency, common error categorizations should be used. The table below provides a recommended categorization which all new ONAP components must adhere to.

Error type	Notes
0	Success
100	Permission errors
200	Availability errors/Timeouts
300	Data errors
400	Schema errors
500	Business process errors
900	Unknown errors

11. ResponseDescription

This field contains a human readable description of the ResponseCode.

12. instanceUUID

If known, this field contains a universally unique identifier used to differentiate between multiple instances of the same (named) log writing service/application. Its value is set at instance creation time (and read by it, e.g., at start/initialization time from the environment). This value should be picked up by the component instance from its configuration file and subsequently used to enable differentiation of log records created by multiple, locally load balanced ONAP component or subcomponent instances that are otherwise identically configured.

13. Category log level

One of the following Enum: “INFO” | “WARN” |”DEBUG” | “ERROR” | “FATAL”.

14. Severity

Optional: 0, 1, 2, 3 see Nagios monitoring/alerting for specifics/details.

15. Server IP address

This field contains the logging component host server’s IP address if known (e.g. Jetty container’s listening IP address). Otherwise it is empty.

16. ElapsedTime

This field contains the elapsed time to complete processing of an API call or transaction request (e.g., processing of a message that was received). This value should be the difference between EndTimestamp and BeginTimestamp fields and must be expressed in milliseconds. This field is required.

17. Server

This field contains the Virtual Machine (VM) Fully Qualified Domain Name (FQDN) if the server is virtualized. Otherwise, it contains the host name of the logging component.

18. ClientIPaddress

This field contains the requesting remote client application’s IP address if known. Otherwise this field can be empty.

19. ClassName

Optional: if available for OO programing languages that support this concept. This is the name of the class that has caused the log record to be created.

20. Unused

This field is deprecated and should be left empty.

21. ProcessKey

This field is optional. This field can be used to capture the flow of a transaction through the system by indicating the components and operations involved in processing. If present, it can be denoted by a comma separated list of components and applications.

22. CustomField1

This field is optional and can be used by developers to include additional application-specific information to support operation and troubleshooting of the system.

23. CustomField2

This field is optional and can be used by developers to include additional application-specific information to support operation and troubleshooting of the system.

24. CustomField3

This field is optional and can be used by developers to include additional application-specific information to support operation and troubleshooting of the system.

25. CustomField4

This field is optional and can be used by developers to include additional application-specific information to support operation and troubleshooting of the system.

26. detailMessage

This field is optional and can be used by developers to extend the record. It may be formatted as is deemed useful to support application-specific needs and use cases.

Please note:

- The field separator character should not appear in this field.
- The field may be used to carry an EELF formatted message.
- Operations may use values in this field as a way of categorizing certain classes of occurrences in a way that improves their efficiency.

Metrics Log Format

The Metrics log captures the detailed suboperations and activities needed to complete the processing of an API request or other activity by an ONAP component. While the Audit log tracks requests received by a component, the Metrics log captures the suboperations needed to fulfill those requests. As such, the Audit log tracks inbound activity (i.e., requests received from external systems or other ONAP components), and the Metrics log tracks outbound activity. The suboperations reflected in the Metrics log may be calls to a subcomponent of the logging ONAP component, calls to other ONAP components, or calls to external (non-ONAP) systems.

The format of the Metrics Log is given in Table 2. Detailed description of the fields are provided in the subsequent sections.

Ord.	Field Name	All field values are required to be supplied in every log record except where explicitly noted as "*Optional*"
1	BeginTimestamp	Date-time when a suboperation activity is begun
2	EndTimestamp	Date-time when a supoperation activity is completed
3	RequestID	Universally unique transaction request ID (UUID)
4	serviceInstanceID	Uniquely identifies a service instance (e.g., "service graph"). It is used as a primary key (e.g., in AAI) to reference/manage the service instance as a unit.
5	threadId	*Optional*: used if wanting to trace processing of a request over a number of threads of a single ONAP component.
6	physical/virtual server name	*Optional*: empty if determined that its value can be added by the agent collecting the log files
7	serviceName	Externally advertised API invoked by clients of this component
8	PartnerName	Client or user invoking the API
9	TargetEntity	ONAP component/subcomponent or non-ONAP entity which is invoked for this suboperation
10	TargetServiceName	External API/operation activities invoked on TargetEntity (ONAP component/subcomponent or non-ONAP entity)
11	StatusCode	High level success or failure of the suboperation activities(COMPLETE or ERROR)
12	ResponseCode	Specific response code returned by the suboperation activities
13	Response Description	Human readable description of the response code
14	instanceUUID	*Optional*: universally unique identifier used to differentiate between multiple instances of the same (named) log writing component.
15	Category log level	Enum: "INFO" | "WARN" |"DEBUG" | "ERROR" | "FATAL".
16	Severity	*Optional*: 0, 1, 2, 3 see Nagios monitoring/alerting for specifics/details.
17	Server IP address	The logging component host server's IP address.

18	ElapsedTime	Elapsed time to complete processing of the suboperation activtities at the granularity available to the component system. This value should be the difference between EndTimestamp and BeginTimestamp fields.
19	Server	VM FQDN if virtualized else host name of the logging component
20	ClientIP	Requesting remote client application's IP address
21	class name	*Optional*: for OO programing languages that support this concept. This is the name of the class that has caused the log record to be created.
22	Unused	
23	ProcessKey	Optional
24	TargetVirtualEntity	Target VNF or VM being acted upon by the component
25	CustomField1	Optional (specific attributes exposed by developers)
26	CustomField2	Optional (specific attributes exposed by developers)
27	CustomField3	Optional (specific attributes exposed by developers)
28	CustomField4	Optional (specific attributes exposed by developers)
29	detailMessage	*Optional*: the rightmost ("last") field in a log record. When present, its value may be formatted if/as useful to meet specific/individual use case(s).

Table 2 – ONAP Metrics Log Record Structure

1. BeginTimestamp

Date-time that processing for the activities begins. See description in the Audit Log section for formatting requirements.

This field is required.

2. EndTimestamp

Date-time that processing for the activities being logged ends. Formatting rules are the same as for the BeginTimestamp field.

This field is required.

When processing for an activity does not return successfully and terminates due to a timeout, this field should indicate when the timeout occurred.

3. RequestID

This field is required. It is meant to enable tracing of processing through different operation activities. When an operation activity invoked by an API call results in multiple suboperations that are logged in Metrics Log records, the Metrics Log records all have the same RequestID as the Audit log record that captures the API call.

See the description of the RequestID field in the Audit Log section for further details.

4. ServiceInstanceID

See the description of this field in the Audit Log section above.

5. ThreadID

See the description of this field in the Audit Log section above.

6. Physical/virtual server name

See the description of this field in the Audit Log section above.

7. serviceName

This field is required. It contains the name of the API invoked at the logging component.

8. PartnerName

This field contains the name of the client or user invoking the API in the prior field, if known.

9. TargetEntity

This field is required. It contains the name of the ONAP component or sub-component, or external entity, at which the operation activities captured in this metrics log record is invoked.

10. TargetServiceName

This field is required. It contains the name of the API or operation activities invoked at the TargetEntity.

11. StatusCode

This field is required. It contains a value of COMPLETE or ERROR to indicate high level success or failure of the operation activities that is invoked.

12. ResponseCode

This field is required. It contains the application-specific response code returned by the operation activities.

13. ResponseDescription

This field contains a human readable description of the response code.

14. instanceUUID

Optional: universally unique identifier used to differentiate between multiple instances of the same (named), log writing service/application. Its value is set at instance creation time (and read by it, e.g. at start/initialization time from the environment). This value should be picked up by the component instance from its configuration file and subsequently used to enable differentiating log records created by multiple, locally load balanced ONAP component or subcomponent instances that are otherwise identically configured.

15. Category log level

One of the following Enum: “INFO” | “WARN” |”DEBUG” | “ERROR” | “FATAL”.

16. Severity

Optional: 0, 1, 2, 3 see Nagios monitoring/alerting for specifics/details.

17. Server IP address

This field contains the logging component host server’s IP address if known (e.g. Jetty container’s listening IP address). Otherwise it is empty.

18. ElapsedTime

This field contains the elapsed time to complete processing of the operation activities. This value should be the difference between EndTimestamp and BeginTimestamp fields and must be expressed in milliseconds. This field is required.

19. Server

This field contains the Virtual Machine (VM) Fully Qualified Domain Name (FQDN) if the server is virtualized. Otherwise, it contains the host name of the logging component.

20. IP address

21. ClassName

Optional: if available for OO programing languages that support this concept. This is the name of the class that has caused the log record to be created.

22. Unused

This field has been deprecated and should be left empty.

23. ProcessKey

This field is optional. This field can be used to capture the flow of a transaction through the system by indicating the components and operations involved in processing. If present, it can be denoted by a comma separated list of components and applications.

24. TargetVirtualEntity

For operation activities that act on a virtual entity (e.g., VNF, VM) this field contains the virtual entity that is the target of the action. Otherwise, the field is empty.

25. CustomField1

This field is optional and can be used by developers to include additional application-specific information to support operation and troubleshooting of the system.

26. CustomField2

This field is optional and can be used by developers to include additional application-specific information to support operation and troubleshooting of the system.

27. CustomField3

This field is optional and can be used by developers to include additional application-specific information to support operation and troubleshooting of the system.

28. CustomField4

This field is optional and can be used by developers to include additional application-specific information to support operation and troubleshooting of the system.

29. detailMessage

This field is optional and can be used by developers to extend the record. It may be formatted as is deemed useful to support application specific needs and use cases.

Please note:

- The field separator character should not appear in this field.
- The field may be used to carry an EELF formatted message.
- Operations support organizations may use values in this field as a way of categorizing certain classes of occurrences in a way that improves their efficiency.

Error Log

The error log records contains information about error conditions encountered by the logging application. The format of the Error Log is specified in Table 3.

Ord.	Field Name	All field values are required to be supplied in every log record except where explicitly noted as "*Optional*"
1	Timestamp	Date-time when error occurs
2	RequestID	Universally unique transaction requestID (UUID).
3	ThreadId	Used if wanting to trace processing of a request over several threads of a single ONAP component.
4	ServiceName	Externally advertised API invoked by clients of this component
5	PartnerName	Client or user invoking the API
6	TargetEntity	ONAP component/subcomponent or non-ONAP entity which is invoked for this suboperation activities
7	TargetServiceName	Known name of External API/operation activities invoked on TargetEntity (ONAP component/subcomponent or non-ONAP entity)
8	ErrorCategory	WARN, ERROR or FATAL
9	ErrorCode	Code representing the error condition
10	ErrorDescription	Human readable description of the error
11	detailMessage	This field may contain any additional information that may be useful in describing the error condition.

Table 3 – ONAP Error Log Record Structure

1. Timestamp

Date-time that the error condition occurred. Formatting rules are the same as for the timestamp fields in the Audit and Metrics logs. This field is required.

2. RequestID

If the error occurs during processing activities for which there is a known RequestID, this field contains that RequestID. Otherwise it is empty.

3. ThreadID

See the description of this field in the Audit Log section above.

4. serviceName

This field is required. It contains the name of the API invoked at the logging component.

5. PartnerName

This field contains the name of the client applicationm user agent or user invoking the API if known.

6. TargetEntity

This field shall be populated if the error occurred while invoking an operation at another entity. In this case, it contains the name of the ONAP component or sub-component, or external entity at which the error occurred. Otherwise it shall be empty.

7. TargetServiceName

This field shall be populated if the error occurred while invoking an operation at another entity. In this case, it contains the name of the API or operation activities invoked at the TargetEntity. Otherwise it shall be empty.

8. ErrorCategory

This field indicates the type of error message. It must be one of *ERROR*, *WARN* or *FATAL*. This field is required.

9. ErrorCode

This field is required and contains an error code representing the error condition. The codes can be chose by the logging application but they should adhere to the guidelines embodied in the following table:

Error type	Notes
100	Permission errors
200	Availability errors/Timeouts
300	Data errors
400	Schema errors
500	Business process errors
900	Unknown errors

10. ErrorDescription

This field is required and contains a human readable description of the error condition.

11. detailMessage

This field is optional and may contain any additional information relevant to the error condition. See the detailMessage field description in the Audit log record for additional formatting information.

Debug Log

The Debug Log is optional and may be used to capture any data that may be needed to debug and correct abnormal conditions of the application.

Ord.	Field Name	All field values are required to be supplied in every log record except where explicitly noted as "*Optional*"
1	Timestamp	Date-time of the log record
2	RequestID	Universally unique transaction requestID (UUID).
3	DebugInfo	Debug Information
	End of Record	Designate the logical end of a multi-line debug record

Table 4 – ONAP Debug Log Record Structure

1. Timestamp

Date-time that the debug record is created. Formatting rules are the same as for the timestamp fields in the Audit and Metrics logs. This field is required.

2. RequestID

This field contains the RequestID associated with the operation being processed for which the debug record is created.

3. DebugInfo

This field is required and contains information that may be useful in debugging. Format of the information is at the discretion of the application.

4. End of debug record

Because this information may include embedded new lines, an explicit end of record ‘marker (“|^\n”) should be added as a ‘last field’ to designate the logical end of the debug record and facilitate automated parsing.