Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: January 13, 2014

F. Le Faucheur, Ed.
Cisco Systems
G. Bertrand, Ed.
I. Oprescu, Ed.
Orange
R. Peterkofsky
Skytide, Inc.
July 12, 2013

CDNI Logging Interface
draft-ietf-cdni-logging-05

Abstract

This memo specifies the Logging interface between a downstream CDN (dCDN) and an upstream CDN (uCDN) that are interconnected as per the CDN Interconnection (CDNI) framework. First, it describes a reference model for CDNI logging. Then, it specifies the CDNI Logging File format and the actual protocol for exchange of CDNI Logging Files.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 13, 2014.

Copyright Notice

Table of Contents

1. Introduction

This memo specifies the CDNI Logging interface between a downstream CDN (dCDN) and an upstream CDN (uCDN). First, it describes a reference model for CDNI logging. Then, it specifies the CDNI Logging File format and the actual protocol for exchange of CDNI Logging Files.

The reader should be familiar with the following documents:

- CDNI problem statement [RFC6707] and framework [I-D.ietf-cdni-framework] identify a Logging interface,
- Section 8 of [I-D.ietf-cdni-requirements] specifies a set of requirements for Logging,
- [RFC6770] outlines real world use-cases for interconnecting CDNs. These use cases require the exchange of Logging information between the dCDN and the uCDN.

As stated in [RFC6707], "the CDNI Logging interface enables details of logs or events to be exchanged between interconnected CDNs".

The present document describes:

- The CDNI Logging reference model (Section 2),
- The CDNI Logging File format (Section 3),

o The CDNI Logging File Exchange protocol (Section 4).

1.1. Terminology

In this document, the first letter of each CDNI-specific term is capitalized. We adopt the terminology described in [RFC6707] and [I-D.ietf-cdni-framework], and extend it with the additional terms defined below.

For clarity, we use the word "Log" only for referring to internal CDN logs and we use the word "Logging" for any inter-CDN information exchange and processing operations related to CDNI Logging interface. Log and Logging formats may be different.

CDN Logging information: logging information generated and collected within a CDN

CDNI Logging information: logging information exchanged across CDNs using the CDNI Logging Interface

Logging information: logging information generated and collected within a CDN or obtained from another CDN using the CDNI Logging Interface

CDNI Logging Field: an atomic element of information that can be included in a CDNI Logging Record. The time an event/task started, the IP address of an End user to whom content was delivered, and the URI of the content delivered are examples of CDNI Logging Fields.

CDNI Logging Record: an information record providing information about a specific event. This comprises a collection of CDNI Logging Fields.

CDNI Logging File: a file containing CDNI Logging Records, as well as additional information facilitating the processing of the CDNI Logging Records.

CDN Reporting: the process of providing the relevant information that will be used to create a formatted content delivery report provided to the CSP in deferred time. Such information typically includes aggregated data that can cover a large period of time (e.g., from hours to several months). Uses of Reporting include the collection of charging data related to CDN services and the computation of Key Performance Indicators (KPIs).

CDN Monitoring: the process of providing content delivery information in real-time. Monitoring typically includes data in real time to provide visibility of the deliveries in progress, for service

operation purposes. It presents a view of the global health of the services as well as information on usage and performance, for network services supervision and operation management. In particular, monitoring data can be used to generate alarms.

1.2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

2. CDNI Logging Reference Model

2.1. CDNI Logging interactions

The CDNI logging reference model between a given uCDN and a given dCDN involves the following interactions:

- customization by the uCDN of the CDNI logging information to be provided by the dCDN to the uCDN (e.g. control of which logging fields are to be communicated to the uCDN for a given task performed by the dCDN, control of which types of events are to be logged). The dCDN takes into account this CDNI logging customization information to determine what logging information to provide to the uCDN, but it may, or may not, take into account this CDNI logging customization information to influence what CDN logging information is to be generated and collected within the dCDN (e.g. even if the uCDN requests a restricted subset of the logging information, the dCDN may elect to generate a broader set of logging information). The mechanism to support the customisation by the uCDN of CDNI Logging information is outside the scope of this document and left for further study. We note that the CDNI Control interface or the CDNI Metadata interface appear as candidate interfaces on which to potentially build such a customisation mechanism in the future. Before such a mechanism is available, the uCDN and dCDN are expected to agree off-line on what CDNI logging information is to be provide by dCDN to UCDN and rely on management plane actions to configure the CDNI Logging functions to generate (respectively, expect) in dCDN (respectively, in uCDN).

- generation and collection by the dCDN of logging information related to the completion of any task performed by the dCDN on behalf of the uCDN (e.g., delivery of the content to an end user) or related to events happening in the dCDN that are relevant to the uCDN (e.g., failures or unavailability in dCDN). This takes place within the dCDN and does not directly involve CDNI interfaces.

- communication by the dCDN to the uCDN of the logging information collected by the dCDN relevant to the uCDN. This is supported by the CDNI Logging interface and in the scope of the present document. For example, the uCDN may use this logging information to charge the CSP, to perform analytics and monitoring for operational reasons, to provide analytics and monitoring views on its content delivery to the CSP or to perform trouble-shooting.

- customization by the dCDN of the logging to be performed by the uCDN on behalf of the dCDN. The mechanism to support the customisation by the dCDN of CDNI Logging information is outside the scope of this document and left for further study.

- generation and collection by the uCDN of logging information related to the completion of any task performed by the uCDN on behalf of the dCDN (e.g., serving of content by uCDN to dCDN for acquisition purposes by dCDN) or related to events happening in the uCDN that are relevant to the dCDN. This takes place within the uCDN and does not directly involve CDNI interfaces.

- communication by the uCDN to the dCDN of the logging information collected by the uCDN relevant to the dCDN. For example, the dCDN might potentially benefit form this information for security auditing or content acquisition troubleshooting. This is outside the scope of this document and left for further study.

Figure 1 provides an example of CDNI Logging interactions (focusing only on the interactions that are in the scope of this document) in a particular scenario where 4 CDNs are involved in the delivery of content from a given CSP: the uCDN has a CDNI interconnection with dCDN-1 and dCDN-2. In turn, dCDN2 has a CDNI interconnection with dCDN3. In this example, uCDN, dCDN-1, dCDN-2 and dCDN-3 all participate in the delivery of content for the CSP. In this example, the CDNI Logging interface enables the uCDN to obtain logging information from all the dCDNs involved in the delivery. In the example, uCDN uses the Logging data:

- to analyze the performance of the delivery operated by the dCDNs and to adjust its operations (e.g., request routing) as appropriate,

- to provide reporting (non real-time) and monitoring (real-time) information to CSP.

For instance, uCDN merges Logging data, extracts relevant KPIs, and presents a formatted report to the CSP, in addition to a bill for the content delivered by uCDN itself or by its dCDNs on his behalf. uCDN may also provide Logging data as raw log files to the CSP, so that the CSP can use its own logging analysis tools.

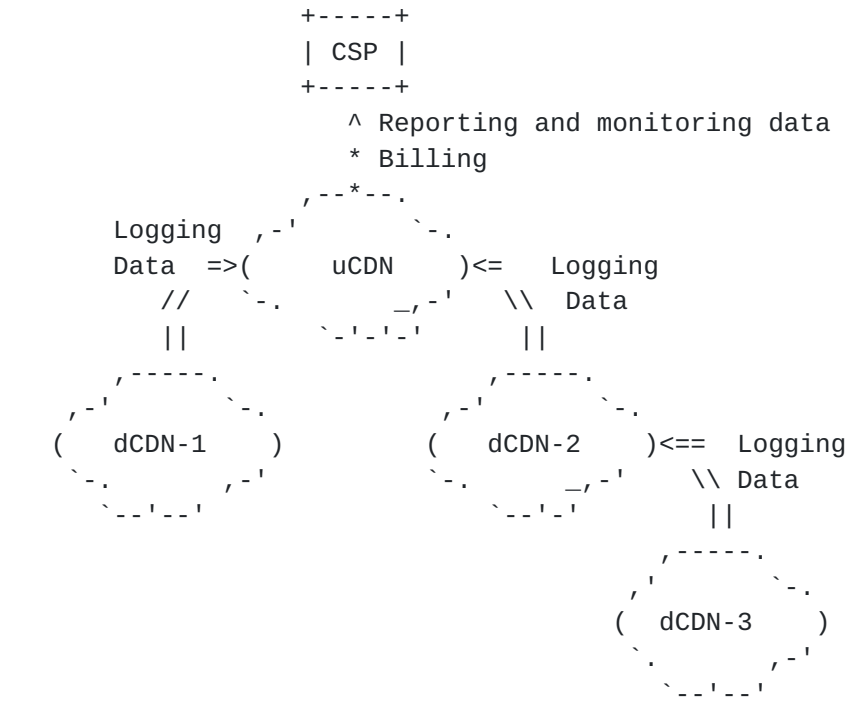


Figure 1: Interactions in CDNI Logging Reference Model

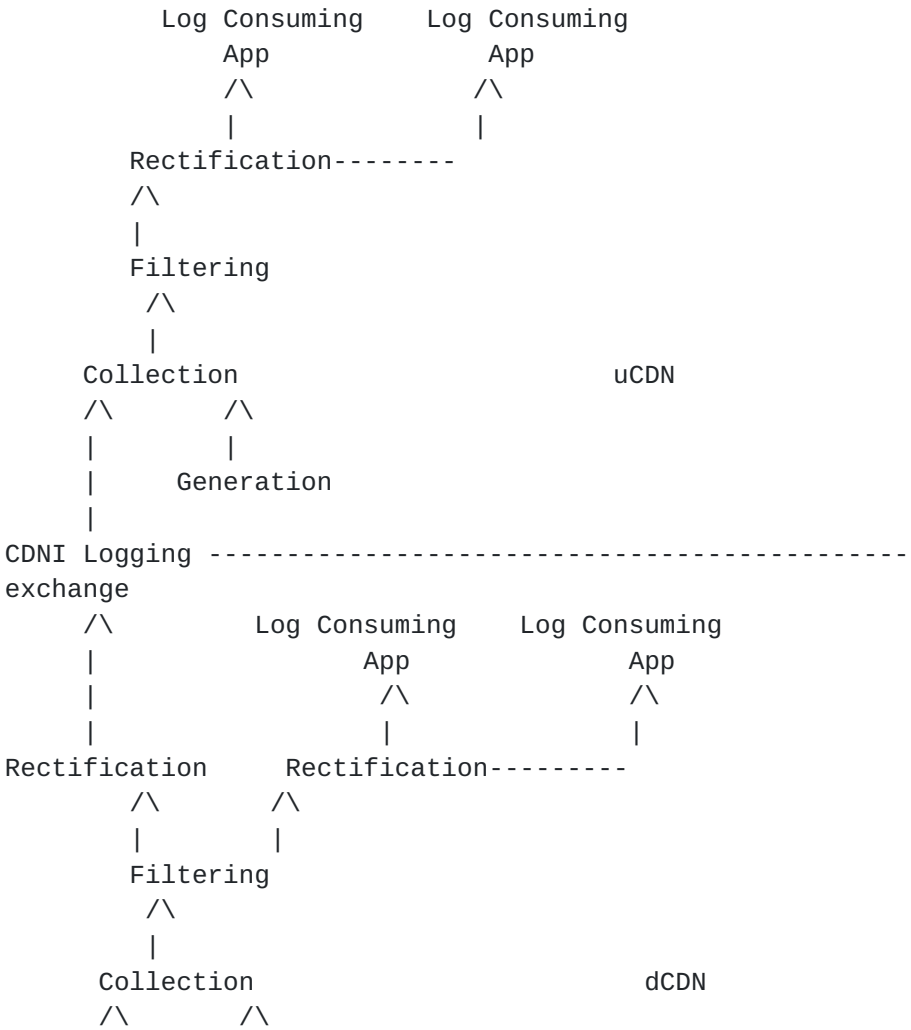
A dCDN (e.g., dCDN-2) integrates the relevant logging information obtained from its dCDNs (e.g., dCDN-3) in the logging information that it provides to the uCDN, so that the uCDN ultimately obtains all logging information relevant to a CSP for which it acts as the authoritative CDN.

Note that the format of Logging information that a CDN provides over the CDNI interface might be different from the one that the CDN uses internally. In this case, the CDN needs to reformat the Logging information before it provides this information to the other CDN over the CDNI Logging interface. Similarly, a CDN might reformat the Logging data that it receives over the CDNI Logging interface before injecting it into its log-consuming applications or before providing some of this logging information to the CSP. Such reformatting operations introduce latency in the logging distribution chain and

introduce a processing burden. Therefore, there are benefits in specifying CDNI Logging format that are suitable for use inside CDNs and also are close to the CDN Log formats commonly used in CDNs today.

2.2. Overall Logging Chain

This section discusses the overall logging chain within and across CDNs to clarify how CDN Logging information is expected to fit in this overall chain. Figure 2 illustrates the overall logging chain within the dCDN, across CDNs using the CDNI Logging interface and within the uCDN. Note that the logging chain illustrated in the Figure is obviously only indicative and varies depending on the specific environments. For example, there may be more or less instantiations of each entity (i.e., there may be 4 Log consuming applications in a given CDN). As another example, there may be one instance of Rectification process per Log Consuming Application instead of a shared one.



```
      |       |
Generation    Generation
```

Figure 2: CDNI Logging in the overall Logging Chain

The following subsections describe each of the processes potentially involved in the logging chain of Figure 2.

2.2.1. Logging Generation and During-Generation Aggregation

CDNs typically generate logging information for all significant task completions, events, and failures. Logs are typically generated by many devices in the CDN including the surrogates, the request routing system, and the control system.

The amount of Logging information generated can be huge. Therefore, during contract negotiations, interconnected CDNs often agree on a Logging retention duration, and optionally, on a maximum size of the Logging data that the dCDN must keep. If this size is exceeded, the dCDN must alert the uCDN but may not keep more Logs for the considered time period. In addition, CDNs may aggregate logs and transmit only summaries for some categories of operations instead of the full Logging data. Note that such aggregation leads to an information loss, which may be problematic for some usages of Logging (e.g., debugging).

[I-D.brandenburg-cdni-has] discusses logging for HTTP Adaptive Streaming (HAS). In accordance with the recommendations articulated there, it is expected that a surrogate will generate separate logging information for delivery of each chunk of HAS content. This ensures that separate logging information can then be provided to interconnected CDNs over the CDNI Logging interface. Still in line with the recommendations of [I-D.brandenburg-cdni-has], the logging information for per-chunck delivery may include some information (a Content Collection IDentifier and a Session IDentifier) intended to facilitate subsequent post-generation aggregation of per-chunk logs into per-session logs. Note that a CDN may also elect to generate aggregate per-session logs when performing HAS delivery, but this needs to be in addition to, and not instead of, the per-chunk delivery logs. We note that this may be revisited in future versions of this document.

Note that in the case of non real-time logging, the trigger of the transmission or generation of the logging file appears to be a synchronous process from a protocol standpoint. The implementation algorithm can choose to enforce a maximum size for the logging file

beyond which the transmission is automatically triggered (and thus allow for an asynchronous transmission process).

2.2.2. Logging Collection

This is the process that continuously collects logs generated by the log-generating entities within a CDN.

In a CDNI environment, in addition to collecting logging information from log-generating entities within the local CDN, the Collection process also collects logging information provided by another CDN, or other CDNs, through the CDNI Logging interface. This is illustrated in Figure 2 where we see that the Collection process of the uCDN collects logging information from log-generating entities within the uCDN as well as logging information coming through CDNI Logging exchange with the dCDN through the CDNI Logging interface.

2.2.3. Logging Filtering

A CDN may require to only present different subset of the whole logging information collected to various log-consuming applications. This is achieved by the Filtering process.

In particular, the Filtering process can also filter the right subset of information that needs to be provided to a given interconnected CDN. For example, the filtering process in the dCDN can be used to ensure that only the logging information related to tasks performed on behalf of a given uCDN are made available to that uCDN (thereby filtering all the logging information related to deliveries by the dCDN of content for its own CSPs). Similarly, the Filtering process may filter or partially mask some fields, for example, to protect End Users' privacy when communicating CDNI Logging information to another CDN. Filtering of logging information prior to communication of this information to other CDNs via the CDNI Logging interface requires that the downstream CDN can recognize the set of log records that relate to each interconnected CDN.

The CDN will also filter some internal scope information such as information related to its internal alarms (security, failures, load, etc).

In some use cases described in [RFC6770], the interconnected CDNs do not want to disclose details on their internal topology. The filtering process can then also filter confidential data on the dCDNs' topology (number of servers, location, etc.). In particular, information about the requests served by every Surrogate may be confidential. Therefore, the Logging information must be protected so that data such as Surrogates' hostnames is not disclosed to the

uCDN. In the "Inter-Affiliates Interconnection" use case, this information may be disclosed to the uCDN because both the dCDN and the uCDN are operated by entities of the same group.

2.2.4. Logging Rectification and Post-Generation Aggregation

If Logging is generated periodically, it is important that the sessions that start in one Logging period and end in another are correctly reported. If they are reported in the starting period, then the Logging of this period will be available only after the end of the session, which delays the Logging generation.

A Logging rectification/update mechanism could be useful to reach a good trade-off between the Logging generation delay and the Logging accuracy. Depending on the selected Logging protocol(s), such mechanism may be invaluable for real time Logging, which must be provided rapidly and cannot wait for the end of operations in progress.

In the presence of HAS, some log-consuming applications can benefit from aggregate per-session logs. For example, for analytics, per-session logs allow display of session-related trends which are much more meaningful for some types of analysis than chunk-related trends. In the case where the log-generating entities have generated during-generation aggregate logs, those can be used by the applications. In the case where aggregate logs have not been generated, the Rectification process can be extended with a Post-Generation Aggregation process that generates per-session logs from the per-chunk logs, possibly leveraging the information included in the per-chunk logs for that purpose (Content Collection IDentifier and a Session IDentifier). However, in accordance with [I-D.brandenburg-cdni-has], this document does not define exchange of such aggregate logs on the CDNI Logging interface. We note that this may be revisited in future versions of this document.

2.2.5. Log-Consuming Applications

2.2.5.1. Maintenance/Debugging

Logging is useful to permit the detection (and limit the risk) of content delivery failures. In particular, Logging facilitates the resolution of configuration issues.

To detect faults, Logging must enable the reporting of any CDN operation success and failure, such as request redirection, content acquisition, etc. The uCDN can summarize such information into KPIs. For instance, Logging format should allow the computation of the number of times during a given epoch that content delivery related to a specific service succeeds/fails.

Logging enables the CDN providers to identify and troubleshoot performance degradations. In particular, Logging enables the communication of traffic data (e.g., the amount of traffic that has been forwarded by a dCDN on behalf of an uCDN over a given period of time), which is particularly useful for CDN and network planning operations.

2.2.5.2. Accounting

Logging is essential for accounting, to permit inter-CDN billing and CSP billing by uCDNs. For instance, Logging information provided by dCDNs enables the uCDN to compute the total amount of traffic delivered by every dCDN for a particular Content Provider, as well as, the associated bandwidth usage (e.g., peak, 95th percentile), and the maximum number of simultaneous sessions over a given period of time.

2.2.5.3. Analytics and Reporting

The goal of analytics is to gather any relevant information to track audience, analyze user behavior, and monitor the performance and quality of content delivery. For instance, Logging enables the CDN providers to report on content consumption (e.g., delivered sessions per content) in a specific geographic area.

The goal of reporting is to gather any relevant information to monitor the performance and quality of content delivery and allow detection of delivery issues. For instance, reporting could track the average delivery throughput experienced by End-Users in a given region for a specific CSP or content set over a period of time.

2.2.5.4. Security

The goal of security is to prevent and monitor unauthorized access, misuse, modification, and denial of access of a service. A set of information is logged for security purposes. In particular, a record of access to content is usually collected to permit the CSP to detect infringements of content delivery policies and other abnormal End User behaviors.

2.2.5.5. Legal Logging Duties

Depending on the country considered, the CDNs may have to retain specific Logging information during a legal retention period, to comply with judicial requisitions.

2.2.5.6. Notions common to multiple Log Consuming Applications

2.2.5.6.1. Logging Information Views

Within a given log-consuming application, different views may be provided to different users depending on privacy, business, and scalability constraints.

For example, an analytics tool run by the uCDN can provide one view to an uCDN operator that exploits all the logging information available to the uCDN, while the tool may provide a different view to each CSP exploiting only the logging information related to the content of the given CSP.

As another example, maintenance and debugging tools may provide different views to different CDN operators, based on their operational role.

2.2.5.6.2. Key Performance Indicators (KPIs)

This section presents, for explanatory purposes, a non-exhaustive list of Key Performance Indicators (KPIs) that can be extracted/ produced from logs.

Multiple log-consuming applications, such as analytics, monitoring, and maintenance applications, often compute and track such KPIs.

In a CDNI environment, depending on the situation, these KPIs may be computed by the uCDN or by the dCDN. But it is usually the uCDN that computes KPIs, because uCDN and dCDN may have different definitions of the KPIs and the computation of some KPIs requires a vision of all the deliveries performed by the uCDN and all its dCDNs.

Here is a list of important examples of KPIs:

- Number of delivery requests received from End-Users in a given region for each piece of content, during a given period of time (e.g., hour/day/week/month)

- Percentage of delivery successes/failures among the aforementioned requests

- Number of failures listed by failure type (e.g., HTTP error code) for requests received from End Users in a given region and for each piece of content, during a given period of time (e.g., hour/day/week/month)

- Number and cause of premature delivery termination for End Users in a given region and for each piece of content, during a given period of time (e.g., hour/day/week/month)

- Maximum and mean number of simultaneous sessions established by End Users in a given region, for a given Content Provider, and during a given period of time (e.g., hour/day/week/month)

- Volume of traffic delivered for sessions established by End Users in a given region, for a given Content Provider, and during a given period of time (e.g., hour/day/week/month)

- Maximum, mean, and minimum delivery throughput for sessions established by End Users in a given region, for a given Content Provider, and during a given period of time (e.g., hour/day/week/month)

- Cache-hit and byte-hit ratios for requests received from End Users in a given region for each piece of content, during a given period of time (e.g., hour/day/week/month)

- Top 10 of the most popularly requested content (during a given day/week/month),

- Terminal type (mobile, PC, STB, if this information can be acquired from the browser type header, for example).

Additional KPIs can be computed from other sources of information than the Logging, for instance, data collected by a content portal or by specific client-side application programming interfaces. Such KPIs are out of scope for the present memo.

The KPIs used depend strongly on the considered log-consuming application -- the CDN operator may be interested in different metrics than the CSP is. In particular, CDN operators are often interested in delivery and acquisition performance KPIs, information related to Surrogates' performance, caching information to evaluate the cache-hit ratio, information about the delivered file size to compute the volume of content delivered during peak hour, etc.

Some of the KPIs, for instance those providing an instantaneous vision of the active sessions for a given CSP's content, are useful essentially if they are provided in real-time. By contrast, some

other KPIs, such as the one averaged on a long period of time, can be provided in non-real time.

3. CDNI Logging File

3.1. Rules

This specification uses the Augmented Backus-Naur Form (ABNF) notation and core rules of [RFC5234]. In particular, the present document uses the following rules from [RFC5234]:

```
CR = %x0D ; carriage return

DIGIT = %x30-39 ; 0-9

DQUOTE = %x22 ; " (Double Quote)

CRLF = CR LF ; Internet standard newline

HEXDIG = DIGIT / "A" / "B" / "C" / "D" / "E" / "F"

HTAB = %x09 ; horizontal tab

LF = %x0A ; linefeed

OCTET = %x00-FF ; 8 bits of data
```

The present document also uses the following rules from [RFC3986]:

host = as specified in section 3.2.2 of [RFC3986].

IPv4address = as specified in section 3.2.2 of [RFC3986].

IPv6address = as specified in section 3.2.2 of [RFC3986].

The present document also defines the folowing additional rules:

```
ADDRESS = IPv4address / IPv6address

DATE = 4DIGIT "-" 2DIGIT "-" 2DIGIT
```

Dates are recorded in the format YYYY-MM-DD where YYYY, MM and DD stand for the numeric year, month and day respectively. All dates are specified in Universal Time Coordinated (UTC).

```
DEC = 1*DIGIT ["." *DIGIT]

QSTRING = DQUOTE *NDQUOTE DQUOTE ; where
```

NDQUOTE = <any OCTET excluding DQUOTE> / 2DQUOTE ; whereby a DQUOTE is conveyed inside a QSTRING unambiguously by repeating it.

NHTABSTRING = *NHTAB ; where

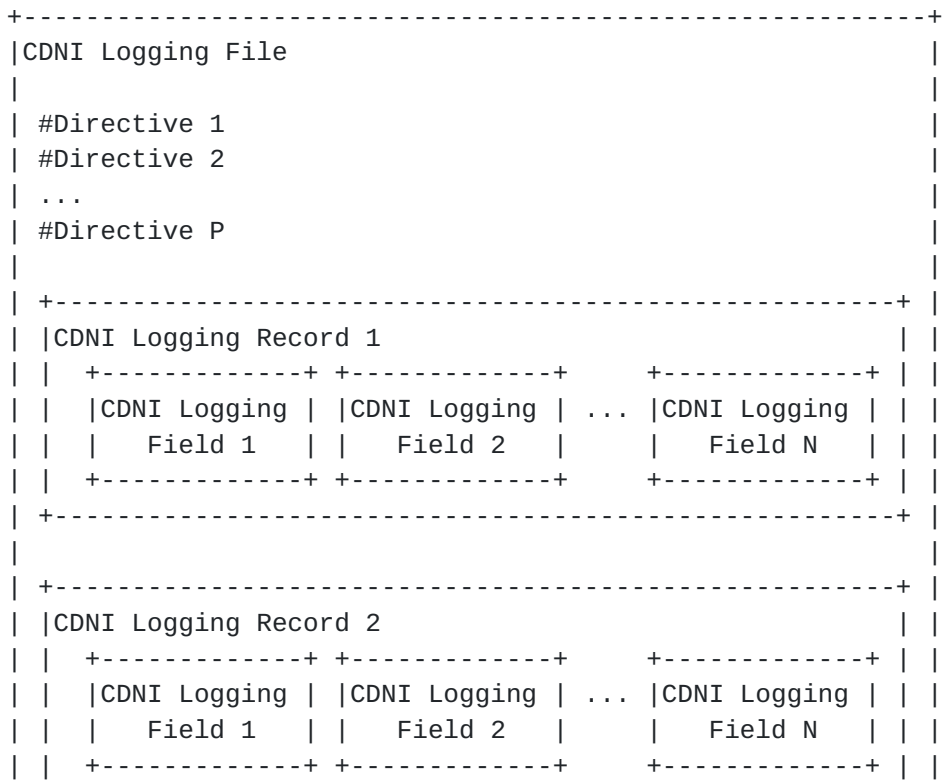
NHTAB = <any OCTET excluding HTAB>

TIME = 2DIGIT ":" 2DIGIT ":" 2DIGIT ["." *DIGIT]

Times are recorded in the form HH:MM:SS or HH:MM:SS.S where HH is the hour in 24 hour format, MM is minutes and SS is seconds. All times are specified in Universal Time Coordinated (UTC).

3.2. CDNI Logging File Structure

As defined in Section 1.1 a CDNI logging field is as an atomic logging information element and a CDNI Logging Record is a collection of CDNI Logging Fields containing all logging information corresponding to a single logging event. This document defines a third level of structure, the CDNI Logging File, that is a collection of CDNI Logging Records. This structure is illustrated in Figure 3. The use of a file structure for transfer of CDNI Logging information is selected since this is the most common practise today for exchange of logging information within and across CDNs.



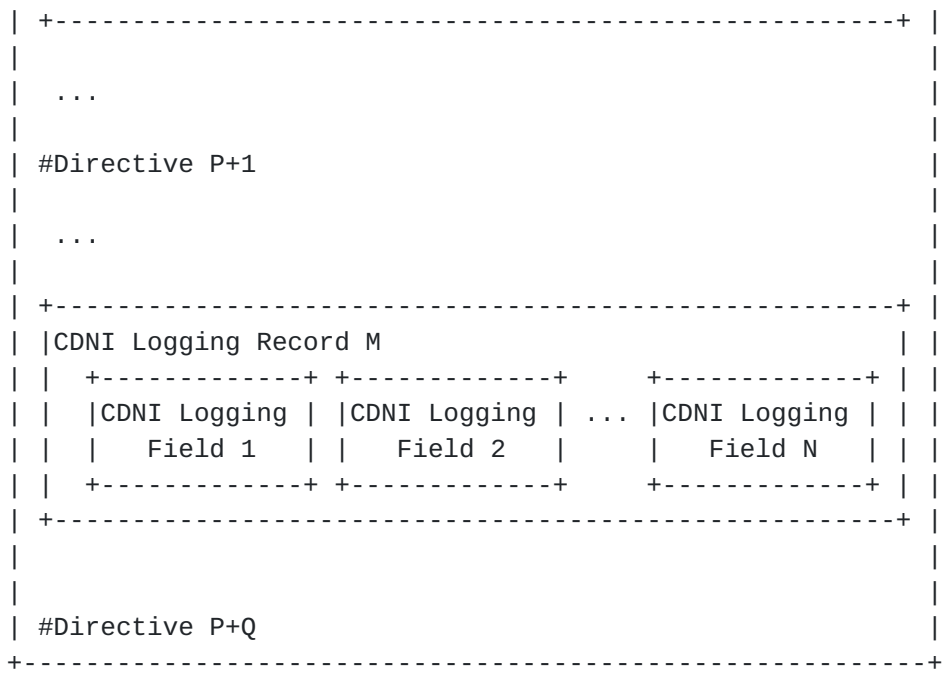


Figure 3: Structure of Logging Files

The CDNI Logging File format is inspired from the W3C Extended Log File Format [ELF]. However, it is fully specified by the present document. Where the present document differs from the W3C Extended Log File Format, an implementation of CDNI Logging MUST comply with the present document.

A CDNI Logging File MUST contain a sequence of lines containing US-ASCII characters [CHAR_SET] terminated by CRLF.

Each line of a CDNI Logging File MUST contain either a directive or a CDNI Logging Record.

Directives record information about the CDNI Logging process itself. Lines containing directives MUST begin with the "#" character. Directives are specified in Section 3.3.

Logging Records provide actual details of the logged event. Logging Records are specified in Section 3.4.

The CDNI File structure is defined by the following rules:

```
DIRLINE = "#" directive CRLF

DIRGROUP = 1*DIRLINE
```

RECLINE = <CDNI Logging Record> CRLF

RECGROUP = *RECLINE

<CDNI Logging File> = 1*<DIRGROUP RECGROUP>

3.3. CDNI Logging File Directives

The CDNI Logging File directives are defined by the following rules:

directive = DIRNAME ":" HTAB DIRVAL

DIRNAME = any CDNI Logging Directive name registered in the CDNI Logging Directive Names registry (Section 6.1).

DIRVAL = <directive value as specified below for each directive name>

An implementation of the CDNI Logging interface MUST support the following directives, listed below by their directive name:

- Version:
  - format: "CDNI" "/" 1*DIGIT "." 1*DIGIT
  - directive value: indicates the version of the CDNI Logging File format. The value MUST be "CDNI/1.0" for the version specified in the present document.
  - occurrence: there MUST be one and only one instance of this directive. It MUST be the first line of the CDNI Logging file.
- UUID:
  - format: NHTABSTRING
  - directive value: this a Universally Unique IDentifier (UUID) from the UUID Uniform Resource Name (URN) namespace specified in [RFC4122]) for the CDNI Logging File .
  - occurrence: there MUST be one and only one instance of this directive.
- Claimed-Origin:
  - format: host

  * directive value: this contains the claimed identification of the entity transmitting the CDNI Logging File (e.g. the host in a dCDN supporting the CDNI Logging interface) or the entity responsible for transmitting the CDNI Logging File (e.g. the dCDN).

  * occurrence: there MUST be zero or one instance of this directive. This directive MAY be included by the dCDN. It MUST NOT be included or modified by the uCDN.

- Verified-Origin:

  * format: host

  * directive value: this contains the identification, as established by the entity receiving the CDNI Logging file, of the entity transmitting the CDNI Logging File (e.g. the host in a dCDN supporting the CDNI Logging interface) or the entity responsible for transmitting the CDNI Logging File (e.g. the dCDN).

  * occurrence: there MUST be zero or one instance of this directive. This directive MAY be added by the uCDN. It MUST NOT be included by the dCDN. The mechanisms used by the uCDN to establih and validate the entity respondible for the CDNI Logging File is outside the scope of the present document. We observe that, in particular, this may be achieved through authentication mechanisms that are part of the CDNI Logging File pull mechanism (Section 4.2).

- Record-Type:

  * format: NHTABSTRING

  * directive value: indicates the type of the CDNI Logging Records that follow this directive, until another Record-Type directive (or the end of the CDNI Logging File). This can be any CDNI Logging Record type registered in the CDNI Logging Record-types registry (Section 6.2). "cdni_http_request_v1" MUST be indicated as the Record-Type directive value for CDNI Logging records corresponding to HTTP request (e.g. a HTTP delivery request) as specified in Section 3.4.1.

  * occurrence: there MUST be at least one instance of this directive. The first instance of this directive MUST precede a Fields directive and precede any CDNI Logging Record.

- Fields:

  * format: FIENAME *<HTAB FIENAME> ; where FIENAME can take any CDNI Logging field name registered in the CDNI Logging Field Names registry (Section 6.3).

  * directive value: this lists the names of all the fields for which a value is to appear in the CDNI Logging Records that are after this directive. The names of the fields, as well as their possible occurrences, are specified for each type of CDNI Logging Records in Section 3.4.

  * occurrence: there MUST be at least one instance of this directive per Record-Type directive. The first instance of this directive for a given Record-Type MUST precede any CDNI Logging Record for this Record-Type.

- Integrity-Hash:

  * format: 32HEXDIG

  * directive value: This directive permits the detection of a corrupted CDNI Logging File. This can be useful, for instance, if a problem occurs on the filesystem of the dCDN Logging system and leads to a truncation of a logging file. The Integrity-Hash value is computed, and included in this directive by the entity that transmits the CDNI Logging File. It is computed by applying the MD5 ([RFC1321]) cryptographic hash function on the CDNI Logging File, including all the directives and logging records, up to the Intergrity-Hash directive itself, excluding the Integrity-Hash directive itself and, when present, also excluding the Non-Repudiation-Hash directive. The Integrity-Hash value is represented as a US-ASCII encoded hexadecimal number, 32 digits long (representing a 128 bit hash value). The entity receiving the CDNI Logging File also computes in a similar way the MD5 hash on the received CDNI Logging File and compares this hash to the value of the Integrity-Hash directive. If the two values are equal, then the received CDNI Logging File MUST be considered non-corrupted. If the two values are different, the received CDNI Logging File MUST be considered corrupted. The behavior of the entity that received a corrupted CDNI Logging File is outside the scope of this specification; we note that the entity MAY attempt to pull again the same CDNI Logging file from the transmitting entity. If the entity receiving the CDNI Logging File adds a Verified-Origin directive, it MUST recompute and update the Integrity-Hash directive so it also protects the added Verified-Origin directive.

* occurrence: there MUST be zero or one instance of this directive. There SHOULD be one instance of this directive. One situation where that directive could be omitted is where integrity protection is already provided via another mechanism (for example if an integrity hash is associated to the CDNI Logging file out of band through the CDNI Logging Logging Feed Section 4.1 leveraging ATOM extensions such as those proposed in [I-D.snell-atompub-link-extensions]. When present, this field MUST be the last line of the CDNI Logging File when the Non-Repudiation-Hash is absent, and MUST be the one before last line of the CDNI Logging File when the Non-Repudiation-Hash is present.

3.4. CDNI Logging Records

A CDNI Logging Record consists of a sequence of CDNI Logging Fields relating to that single CDNI Logging Record.

CDNI Logging Fields MUST be separated by the "horizontal tabulation (HTAB)" character.

To facilitate readability, a prefix scheme is used for CDNI Logging field names in a similar way to the one used in W3C Extended Log File Format [ELF] . The semantics of the prefix in the present document is:

- c: refers to the User Agent that issues the request (corresponds to the "client" of W3C Extended Log Format)
- d: refers to the dCDN (relative to a given CDN acting as a uCDN)
- s: refers to the dCDN Surrogate that serves the request (corresponds to the "server" of W3C Extended Log Format)
- u: refers to the uCDN (relative to a given CDN acting as a dCDN)
- cs: refers to communication from the User-Agent towards the dCDN Surrogate
- sc: refers to communication from the dCDN Surrogate towards the User-Agent

An implementation of the CDNI Logging interface as per the present specification MUST support the CDNI HTTP Delivery Records as specified in Section 3.4.1. [Editor's Note": other types of CDNI Logging records will be listed here if we specify other types for this version eg Request Routing].

A CDNI Logging Record is defined by the following rules:

FIEVAL = <CDNI Logging Field value>

<CDNI Logging Record> = FIEVAL *<HTAB FIEVAL> ; where FIEVAL contains the CDNI Logging field values corresponding to the CDNI Logging field names (FIENAME) listed is the last Fields directive predecing the present CDNI Logging Record.

3.4.1. HTTP Request Logging Record

The HTTP Request Logging Record is a CDNI Logging Record of Record-Type "cdni_http_request_v1". It contains the following CDNI Logging Fields, listed by their field name:

- date:
  - format: DATE
  - field value: the date at which the processing of request completed on the Surrogate.
  - occurrence: there MUST be one and only one instance of this field.
- time:
  - format: TIME
  - field value: the time at which the processing of request completed on the Surrogate.
  - occurrence: there MUST be one and only one instance of this field.
- time-taken:
  - format: DEC
  - field value: decimal value of the duration, in seconds, between the start of the processing of the request and the completion of the request processing (e.g. completion of delivery) by the Surrogate.
  - occurrence: there MUST be one and only one instance of this field.
- c-ip:

  * format: ADDRESS

  * field value: the source IPv4 or IPv6 address (i.e. the "client" address) in the request received by the Surrogate.

  * occurrence: there MUST be one and only one instance of this field.

- c-ip-anonimizing:

  * format: 1*DIGIT

  * field value: the number of rightmost bits of the address in the c-ip field that are zeroed-out in order to anonymize the logging record. The mechanism by which the two ends of the CDNI Logging interface agree on whether anonimization is to be supported and the number of bits that need to be zeroed-out for this purpose are outside the scope of the present document.

  * occurrence: there MUST be zero or one instance of this field.

- c-port:

  * format: 1*DIGIT

  * field value: the source TCP port (i.e. the "client" port) in the request received by the Surrogate.

  * occurrence: there MUST be zero or exactly one instance of this field.

- s-ip:

  * format: ADDRESS

  * field value: the IPv4 or IPv6 address of the Surrogate that served the request (i.e. the "server" address).

  * occurrence: there MUST be zero or exactly one instance of this field.

- s-hostname:

  * format: host

  * field value: the hostname of the Surrogate that served the request (i.e. the "server" hostname).

  * occurrence: there MUST be zero or exactly one instance of this field.

- s-port:
  * format: 1*DIGIT
  * field value: the destination TCP port (i.e. the "server" port) in the request received by the Surrogate.
  * occurrence: there MUST be zero or exactly one instance of this field.

- cs-method:
  * format: NHTABSTRING
  * field value: this is the HTTP method of the HTTP request received by the Surrogate.
  * occurrence: There MUST be one and only one instance of this field.

- cs-uri:
  * format: NHTABSTRING
  * field value: this is the complete URL of the request received by the Surrogate. It is exactly in the format of a http_URL specified in [RFC2616]) or, when the request was a HTTPS request ([RFC2818]), it is in the format of a http_URL but with the scheme part set to "https" instead of "http".
  * occurrence: there MUST be zero or exactly one instance of this field.

- u-uri:
  * format: NHTABSTRING
  * field value: this is a complete URL, derived from the complete URI of the request received by the Surrogate (i.e. the cs-uri) but transformed by the entity generating or transmitting the CDNI Logging Record, in a way that is agreed upon between the two ends of the CDNI Logging interface, so the transformed URI is meaningful to the uCDN. For example, the two ends of the CDNI Logging interface could agree that the u-uri is constructed from the cs-uri by removing the part of the

hostname that exposes which individual Surrogate actually performed the delivery. The details of modification performed to generate the u-uri, as well as the mechanism to agree on these modifications between the two sides of the CDNI Logging interface are outside the scope of the present document.

* occurrence: there MUST be one and only one instance of this field.

- protocol:
  * format: NHTABSTRING
  * field value: this is value of the HTTP-Version field as specified in [RFC2616] of the Request-Line of the request received by the Surrogate (e.g. "HTTP/1.1").
  * occurrence: there MUST be one and only one instance of this field.
- sc-status:
  * format: 3DIGIT
  * field value: this is the HTTP Status-Code in the HTTP response from the Surrogate.
  * occurrence: There MUST be one and only one instance of this field.
- sc-total-bytes:
  * format: 1*DIGIT
  * field value: this is the total number of bytes of the HTTP response sent by the Surrogate in response to the request. This includes the bytes of the Status-Line (including HTTP headers) and of the message-body.
  * occurrence: There MUST be one and only one instance of this field.
- sc-entity-bytes:
  * format: 1*DIGIT
  * field value: this is the number of bytes of the message-body in the HTTP response sent by the Surrogate in response to the

request. This does not include the bytes of the Status-Line (and therefore does not include the bytes of the HTTP headers).

* occurrence: there MUST be zero or exactly one instance of this field.

- cs(<HTTP-header-name>):

  * format: QSTRING

  * field value: the value of the HTTP header (identified by the <HTTP-header-name> in the CDNI Logging field name) as it appears in the request processed by the Surrogate. For example, when the CDNI Logging field name (FIENAME) listed in the prededing Fields directive is "cs(User-Agent"), this CDNI Logging field value contains the value of the User-Agent HTTP header as received by the Surrogate in the request it processed.

  * occurrence: there MUST be zero, one or any number of instance of this field.

- sc(<HTTP-header-name>):

  * format: QSTRING

  * field value: the value of the HTTP header (identified by the <HTTP-header-name> in the CDNI Logging field name) as it appears in the response issued by the Surrogate to serve the request.

  * occurrence: there MUST be zero, one or any number of instance of this field.

- s-ccid:

  * format: QSTRING

  * field value: this contains the value of the Content Collection IDentifier associated by the uCDN to the content served by the Surrogate via the CDNI Metadata interface ([I-D.ietf-cdni-metadata]).

  * occurrence: there MUST be zero or exactly one instance of this field.

- s-sid:

  * format: QSTRING

  * field value: this contains the value of a Session IDentifier generated by the dCDN for a specific HTTP Adaptive Streaming (HAS) session and whose value is included in the Logging record for every content chunk delivery of that session in view of facilitating the later correlation of all the per content chunk log records of a given HAS session. See section 3.4.2.2. of [I-D.brandenburg-cdni-has] for more discussion on the concept of Session IDentifier.

  * occurrence: there MUST be zero or exactly one instance of this field.

- s-cached:

  * format: 1DIGIT

  * field value: this characterises whether the Surrogate could serve the request using content already stored on its local cache. The allowed values are "0" (for miss) and "1" for hit). "1" MUST be used when the Surrogate could serve the request using exclusively content already stored on its local cache. "0" MUST be used otherwise (including cases where the Surrogate served the request using some, but not all, content already stored on its local cache). Note that a "0" only means a cache miss in the Surrogate and does not provide any information on whether the content was already stored, or not, in another device of the dCDN i.e. whether this was a "dCDN hit" or "dCDN miss".

  * occurrence: there MUST be zero or exactly one instance of this field.

- s-uri-signing:

  * format: 1DIGIT

  * field value: this characterises the uri signing validation performed by the Surrogate on the request. The allowed values are:

  *

    + "0" : no uri signature validation performed

    + "1" : uri signature validation performed and validated

+ "2" : uri signature validation performed and rejected

* occurrence: there MUST be zero or exactly one instance of this field.

The "Fields" directive corresponding to a HTTP Request Logging Record MUST list all the fields name whose occurrence is specified above as "There MUST be one and only one instance of this field". The corresponding fields value MUST be present in every HTTP Request Logging Record.

The "Fields" directive corresponding to a HTTP Request Logging Record MAY list all the fields value whose occurrence is specified above as "there MUST be zero or exactly one instance of this field" or "there MUST be zero, one or any number of instance of this field". The set of such fields name actually listed in the "Fields" directive is selected by the implementation generating the CDNI Logging File based on agreements between the interconnected CDNs established through mechanisms outside the scope of this specification (e.g. contractual agreements) . When such a field name is not listed in the "Fields" directive, the corresponding field value MUST NOT be included in the Logging Record. When such a field name is listed in the "Fields" directive, the corresponding field value MUST be included in the Logging Record; in that case, if the value for the field is not available, this MUST be conveyed via a dash character ("-").

The fields name listed in the "Fields" directive MAY be listed in the order in which they are listed in Section 3.4.1 or MAY be listed in any other order.

3.5. CDNI Logging File Example

#Version:<HTAB>CDNI/1.0<CRLF>

#UUID:<HTAB>"urn:uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6"<CRLF>

#Claimed-Origin:<HTAB>cdni-logging-entity.dcdn.example.com<CRLF>

#Record-Type:<HTAB>cdni_http_request_v1<CRLF>

#Fields:<HTAB>date<HTAB>time<TAB>time-taken<HTAB>c-ip<HTAB>cs-method<HTAB>u-uri<HTAB>protocol<HTAB>sc-status<HTAB>sc-total-bytes<HTAB>cs(User-Agent)<HTAB>cs(Referer)<HTAB>s-cached<CRLF>

2013-05-17<HTAB>00:38:06.825<HTAB>9.058<HTAB>10.5.7.1<HTAB>GET<HTAB>http://cdni-ucdn.dcdn.example.com/video/movie100.mp4<HTAB>HTTP/1.1<HTAB>200<HTAB>6729891<HTAB>"Mozilla/5.0 (Windows; U; Windows NT 6.0; en-US) AppleWebKit/533.4 (KHTML, like Gecko) Chrome/5.0.375.127 Safari /533.4"<HTAB>"host1.example.com"<HTAB>1<CRLF>

2013-05-17<HTAB>00:39:09.145<HTAB>15.32<HTAB>10.5.10.5<HTAB>GET<HTAB>http://cdni-ucdn.dcdn.example.com/video/movie118.mp4<HTAB>HTTP/1.1<HTAB>200<HTAB>15799210<HTAB>"Mozilla/5.0 (Windows; U; Windows NT 6.0; en-US) AppleWebKit/533.4 (KHTML, like Gecko) Chrome/5.0.375.127 Safari /533.4"<HTAB>"host1.example.com"<HTAB>1<CRLF>

2013-05-17<HTAB>00:42:53.437<HTAB>52.879<HTAB>10.5.10.5<HTAB>GET<HTAB>http://cdni-ucdn.dcdn.example.com/video/picture11.mp4<HTAB>HTTP/1.0<HTAB>200<HTAB>97234724<HTAB>"Mozilla/5.0 (Windows; U; Windows NT 6.0; en-US) AppleWebKit/533.4 (KHTML, like Gecko) Chrome/5.0.375.127 Safari /533.4"<HTAB>"host5.example.com"<HTAB>0<CRLF>

#Integrity-Hash: 9e107d9d372bb6826bd81d3542a419d6 [Editor's Note: include the correct MD5-hash value for the actual example]<CRLF>

4. CDNI Logging File Exchange Protocol

This document specifies a protocol for the exchange of CDNI Logging Files as specified in Section 3.

This protocol comprises:

- a CDNI Logging feed, allowing the dCDN to notify the uCDN about the CDNI Logging files that can be retrieved by that uCDN from the dCDN, as well as all the information necessary for retrieving each of these CDNI Logging File. The CDNI Logging feed is specified in Section 4.1.

- a CDNI Logging File pull mechanism, allowing the uCDN to obtain from the dCDN a given CDNI Logging File at the uCDN convenience. The CDNI Logging File pull mechanisms is specified in Section 4.2.

An implementation of the CDNI Logging interface as per the present document generating CDNI Logging file (i.e. on the dCDN side) MUST support the server side of the CDNI Logging feed and the server side of the CDNI Logging pull mechanism.

An implementation of the CDNI Logging interface as per the present document consuming CDNI Logging file (i.e. on the uCDN side) MUST support the client side of the CDNI Logging feed and the client side of the CDNI Logging pull mechanism.

We note that implementations of the CDNI Logging interface MAY also support other mechanisms to exchange CDNI Logging Files, for example in view of exchanging logging information with minimum time-lag (e.g. sub-minute or sub-second) between when the event occurred in the dCDN and when the corresponding Logging Record is made available to the uCDN (e.g. for log-consuming applications requiring extremely fresh logging information such as near-real-time content delivery monitoring). Such mechanism might be defined in future version of the present document.

4.1. CDNI Logging Feed

The server-side implementation of the CDNI Logging feed produces an Atom feed [RFC4287]. This feed is used to advertise log files that are available for the client-side to retrieve using the CDNI Logging pull mechanism.

4.1.1. Atom Formatting

A CDNI Logging feed MUST be structured as an Archived feed, as defined in [RFC5005], and MUST be formatted in Atom [RFC4287]. This means it consists of a subscription document that is regularly updated as new CDNI logging files become available, and information about older CDNI Logging files is moved into archive documents. Once created, archive documents are never modified.

Each CDNI Logging file listed in an Atom feed MUST be described in an atom:entry container element.

The atom:entry MUST contain an atom:content element whose "src" attribute is a link to the CDNI Logging file and whose "type" attribute is the MIME Media Type indicating that the entry if a CDNI Logging File. We define this MIME Media Type as "application/cdni.LoggingFile" (See Section 6.4).

For compatibility with some Atom feed readers the atom:entry MAY also contain an atom:link entry whose "href" attribute is a link to the CDNI Logging file and whose "type" attribute is the MIME Media Type indicating that the entry if a CDNI Logging File. We define this MIME Media Type as "application/cdni.LoggingFile" (See Section 6.4).

The IRI used in the atom:id of the atom:entry MUST contain the UUID of the CDNI Logging file.

The atom:updated in the atom:entry MUST indicate the time at which the CDNI Logging file was last updated.

4.1.2. Updates to Log Files and the Feed

CDNI Logging files MUST NOT be modified by the dCDN once published in the CDNI Logging feed.

The frequency with which the subscription feed is updated, the period of time covered by each CDNI Logging file or each archive document, and timeliness of publishing of CDNI Logging files is outside the scope of the present document and is expected to be agreed upon by uCDN and dCDN via other means (e.g. human agreement).

The server-side implementation SHOULD use HTTP cache control headers on the subscription feed to indicate the frequency at which the client-side is to poll for updates.

4.1.3. Redundant Feeds

The server-side implementation MAY present more than one CDNI Logging feed and for redundancy, CDNI Logging files MAY be published in more than one feed.

A client-side implementation MAY support such redundant CDNI Logging feeds. If it supports redundant CDNI Logging feed, the client-side SHOULD use the UUID of the CDNI Logging file, presented in the atom:id element of the Atom feed, to avoid uncessarily pulling each CDNI Logging file more than once.

4.1.4. Example CDNI Logging Feed

Figure 4 illustrates an example of the subscription document of a CDNI Logging feed.

```
<?xml version="1.0" encoding="utf-8"?>
<feed xmlns="http://www.w3.org/2005/Atom"
<http://www.w3.org/2005/Atom%22>>
  <title type="text">CDNI Logging Feed</title>
  <updated>2013-03-23T16:21:11Z</updated>
  <id>urn:uuid:663ae677-40fb-e99a-049d-c5642916b8ce</id>
  <link href="https://dcdn.example/logfeeds/ucdn1"
     rel="self" type="application/atom+xml" />
  <link href="https://dcdn.example/logfeeds/ucdn1"
     rel="current" type="application/atom+xml" />
  <link href="https://dcdn.example/logfeeds/ucdn1/201303231400"
     rel="prev-archive" type="application/atom+xml" />
  <generator version="example version 1">CDNI Log Feed
     Generator</generator>
  <author><name>dcdn.example</name></author>
  <entry>
```

```
      <title type="text">CDNI Logging File for uCDN at
        2013-03-23 14:55:00</title>
        <id>urn:uuid:12345678-1234-abcd-00aa-01234567abcd</id>
        <updated>2013-03-23T14:55:00Z</updated>
        <content src="https://dcdn.example/logs/ucdn/
           http-requests-20130323145500000000"
           type="application/cdni.LoggingFile" />
        <summary>CDNI Logging File for uCDN at
        2013-03-23 14:55:00</summary>
    </entry>
    <entry>
      <title type="text">CDNI Logging File for uCDN at
        2013-03-23 15:55:00</title>
        <id>urn:uuid:87654321-4321-dcba-aa00-dcba7654321</id>
        <updated>2013-03-23T15:55:00Z</updated>
        <content src="https://dcdn.example/logs/ucdn/
           http-requests-20130323155500000000"
           type="application/cdni.LoggingFile" />
        <summary>CDNI Logging File for uCDN at
        2013-03-23 15:55:00</summary>
    </entry>
    <entry>
      ...
    </entry>
     ...
  ?</feed>
```

Figure 4: Example subscription document of a CDNI Logging Feed

4.2. CDNI Logging File Pull

A client-side implementation of the CDNI Logging interface pulls, at its convenience, any CDNI Logging File that is published by the server-side in the CDNI Logging Feed. To do so, the client-side:

- MUST use HTTP v1.1
- SHOULD use TLS (i.e. use what is loosely referred to as "HTTPS") whenever protection of the CDNI LOgging information is required (see Section 7.1)
- MUST use the URI associated to the CDNI Logging File (in the "src" attribute of the corresponding atom:content element) in the CDNI Logging Feed
- SHOULD indicate the compression schemes it supports

Note that a client-side implementation of the CDNI Logging interface MAY pull a CDNI Logging File that it has already pulled, as long as the file is still published by the server-side in the subscription document of CDNI Logging Feed.

[Editor's note: if a given Logging file is moved away from subscription document to an archive document, do we agree it may no longer be accessible to uCDN?]

The server-side implementation MUST respond to any valid pull request by a client-side implementation for a CDNI Logging File published by the server-side in the subscription document of the CDNI Logging Feed. The server-side implementation:

- MUST handle the client-side request as per HTTP v1.1
- MUST include the CDNI Logging File identified by the request URI inside the body of the HTTP response
- MUST support the gzip and deflate compression schemes
- MAY support other compression schemes
- when the client-side request indicates client-supported compression schemes, the server-side SHOULD use a compression scheme that it supports and is supported by the client-side

5. Open Issues

- Compression: <Ben>When we say the server MUST support gzip & deflate we probably need to think through whether we mean content-encoding, transfer-encoding or both. The semantics get a little confusing so we probably just need to think them through to ensure we allow a server to store compressed logs as transmit them compressed.
- Handling of Event logs and notifications: There are two aspects of that question:
  - non-real-time exchange of event logs from dCDN to uCDN for audit purposes. This could be added to current spec presumably in the form of additional Record-Types and without requiring a significant change to the current CDNI Logging file exchange approach. It is proposed that this be handled as a [MED] requirement. e.g. try first specify what events and what information needs to be exchanged; and depending on progress, decide to include in initial logging spec or not

* real-time exchange of event notification from dCDN to uCDN for immediate operational action (eg on notification by dCDN that dCDN request routing is down, uCDN stops redirecting to this dCDN). This would presumably require definition/extension of another CDNI interface or significant change/extension to the current CDNI logging spec. It is proposed that this be kept out of the scope of the current cdni-logging spec .

Another question is what set of events should be logged/notified. The first type of events relates to "service-level" events i.e. high level events that affect the service that the dCDN is providing to the uCDN (e.g.dCDN request routing is down, dCDN is overloaded). There is general agreements that it is desirable to be able to log/notify such service-level events. The second type of events is "atomic-level" events i.e. low level events that may be useful to identify or track a component issue or a delivery issue. logging/notifying about such events may be useful in some situations (eg uCDN and dCDN have a particular relationship allowing them to share detailed operational information) and may not be useful in some situations (because the dCDN does not want to expose details of its CDN operation). Ideal approach is to define both types of events and have the first type as MUST and the second type as MAY. Fall back approach would be to only define the first type initially.

o Add precise definition of what must be supported by transmitting implementation and what must be implemented by receiving application (regardless of what may actually be used in a given deployment). For example, it may be reasonable to mandate that a receiving implementation be able to receive all the directives specified in the doc and all fields.

o Clean-up MUST/SHOULD/MAY to clarify (where needed) implementations requirements (ie what any implementation must be capable of doing) vs deployment requirements (ie what must be used in a given environment).

6. IANA Considerations

6.1. CDNI Logging Directive Names Registry

The IANA is requested to create a new registry, CDNI Logging Directive Names.

The initial contents of the CDNI Logging File Directives registry comprise the names of the directives specified in Section 3.3 of the present document, and are as follows:

Directive Name	Reference
Version	RFC xxxx
UUID	RFC xxxx
Claimed-Origin	RFC xxxx
Verified-Origin	RFC xxxx
Record-Type	RFC xxxx
Fields	RFC xxxx
Integrity-Hash	RFC xxxx

Figure 5

[Instructions to IANA: Replace "RFC xxxx" above by the RFC number of the present document]

Within the registry, names are to be allocated by IANA according to the "Specification Required" policy specified in [RFC5226].

6.2. CDNI Logging Record-Types Registry

The IANA is requested to create a new registry, CDNI Logging Record-Types.

The initial contents of the CDNI Logging Record-Types registry comprise the names of the CDNI Logging Record types specified in Section 3.4 of the present document, and are as follows:

Record-Types	Reference
cdni_http_request_v1	RFC xxxx

Figure 6

[Instructions to IANA: Replace "RFC xxxx" above by the RFC number of the present document]

Within the registry, Record-Types are to be allocated by IANA according to the "Specification Required" policy specified in [RFC5226].

6.3. CDNI Logging Field Names Registry

The IANA is requested to create a new registry, CDNI Logging Field Names.

The initial contents of the CDNI Logging Fields Names registry comprise the names of the CDNI Logging fields specified in Section 3.4 of the present document, and are as follows:

Field Name	Reference
date	RFC xxxx
time	RFC xxxx
time-taken	RFC xxxx
c-ip	RFC xxxx
c-ip-anonimizing	RFC xxxx
c-port	RFC xxxx
s-ip	RFC xxxx
s-hostname	RFC xxxx
s-port	RFC xxxx
cs- method	RFC xxxx
cs-uri	RFC xxxx
u-uri	RFC xxxx
protocol	RFC xxxx
sc-status	RFC xxxx
sc- total-bytes	RFC xxxx
sc-entity-bytes	RFC xxxx
cs(<HTTP-header>)	RFC xxxx
sc(<HTTP-header>)	RFC xxxx
s-ccid	RFC xxxx
s-sid	RFC xxxx
s-cached	RFC xxxx
s-uri-signing	RFC xxxx

Figure 7

[Instructions to IANA: Replace "RFC xxxx" above by the RFC number of the present document]

Within the registry, names are to be allocated by IANA according to the "Specification Required" policy specified in [RFC5226].

6.4. CDNI Logging MIME Media Type

The IANA is requested to allocate the "application/cdni.LoggingFile" MIME Media Type (whose use is specified in Section 4.1.1 of the present document) in the MIME Media Types registry.

7. Security Considerations

7.1. Authentication, Confidentiality, Integrity Protection

The use of TLS for transport of the CDNI Logging feed mechanism (Section 4.1) and CDNI Logging File pull mechanism (Section 4.2) allows:

- the dCDN and uCDN to authenticate each other (to ensure they are transmitting/receiving CDNI Logging File from an authenticated CDN)
- the CDNI Logging information to be transmitted with confidentiality
- the integrity of the CDNI Logging information to be protected during the exchange

In an environment where any such protection is required, TLS SHOULD be used for transport of the CDNI Logging feed and the CDNI Logging File pull.

A CDNI Logging implementation MUST support TLS transport of the CDNI Logging feed and the CDNI Logging File pull.

The Integrity-Hash directive inside the CDNI Logging File provides additional integrity protection, this time targeting potential corruption of the CDNI logging information during the CDNI Logging File generation. This mechanism does not allow restoration of the corrupted CDNI Logging information, but it allows detection of such corruption and therefore triggering of appropraite correcting actions (e.g. discard of corrupted information, attempt to re-obtain the CDNI Logging information).

7.2. Non Repudiation

The Non-Repudiation-Signature directive in the CDNI Logging File allows support of non-repudiation of the CDNI Logging File by the dCDN. The optional Non-Repudiation-Hash can be used on the CDNI Logging interface where needed.

7.3. Privacy

CDNs have the opportunity to collect detailed information about the downloads performed by End-Users. The provision of this information to another CDN introduces potential End-Users privacy protection concerns. We observe that when CDNI interconnection is realised as per [I-D.ietf-cdni-framework], the uCDN handles the initial End-User requests (before it is redirected to the dCDN) so, regardless of which information is, or is not, communicated to the uCDN through the

CDNI Logging interface, the uCDN has visibility on significant information such as the IP address of the End-User request and the URL of the request. Nonetheless, if the dCDN and uCDN agree that anonymization is required to avoid making some detailed information available to the uCDN (such as how much bytes of the content has been watched by an enduser and/or at what time) or is required to meet some legal obligations, then the uCDN and dCDN can agree to exchange anonymized End-User IP addresses in CDNI Logging files and the c-ip-anonymization field can be used to convey the number of bits that have been anonymized so that the meaningful information can still be easily extracted from the anonymized addressses (e.g. for geolocation aware analytics).

8. Acknowledgments

This document borrows from the W3C Extended Log Format [ELF].

Rob Murray significantly contributed into the text of Section 4.1 .

The authors would like to thank Sebastien Cubaud, Pawel Grochocki, Christian Jacquenet, Yannick Le Louedec, Anne Marrec and Emile Stephan for their contributions on early versions of this document. The authors would like also to thank Fabio Costa, Sara Oueslati, Yvan Massot, Renaud Edel, and Joel Favier for their input and comments. Finally, they thank the contributors of the EU FP7 OCEAN project for valuable inputs.

9. References

9.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.

[RFC4122] Leach, P., Mealling, M., and R. Salz, "A Universally Unique IDentifier (UUID) URN Namespace", RFC 4122, July 2005.

[RFC4287] Nottingham, M., Ed. and R. Sayre, Ed., "The Atom Syndication Format", RFC 4287, December 2005.

[RFC5005] Nottingham, M., "Feed Paging and Archiving", RFC 5005, September 2007.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

[RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.

9.2. Informative References

[CHAR_SET] , "IANA Character Sets registry", , <http://www.iana.org/assignments/character-sets/character-sets.xml>.

[ELF] Phillip M. Hallam-Baker, . and . Brian Behlendorf, "Extended Log File Format, W3C (work in progress), WD-logfile-960323", , <http://www.w3.org/TR/WD-logfile.html>.

[I-D.brandenburg-cdni-has] Brandenburg, R., Deventer, O., Faucheur, F., and K. Leung, "Models for adaptive-streaming-aware CDN Interconnection", draft-brandenburg-cdni-has-05 (work in progress), April 2013.

[I-D.ietf-cdni-framework] Peterson, L. and B. Davie, "Framework for CDN Interconnection", draft-ietf-cdni-framework-03 (work in progress), February 2013.

[I-D.ietf-cdni-metadata] Niven-Jenkins, B., Murray, R., Watson, G., Caulfield, M., Leung, K., and K. Ma, "CDN Interconnect Metadata", draft-ietf-cdni-metadata-01 (work in progress), February 2013.

[I-D.ietf-cdni-requirements] Leung, K. and Y. Lee, "Content Distribution Network Interconnection (CDNI) Requirements", draft-ietf-cdni-requirements-09 (work in progress), July 2013.

[I-D.snell-atompub-link-extensions] Snell, J., "Atom Link Extensions", draft-snell-atompub-link-extensions-09 (work in progress), June 2012.

[RFC1321] Rivest, R., "The MD5 Message-Digest Algorithm", RFC 1321, April 1992.

[RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, May 2000.

[RFC6707] Niven-Jenkins, B., Le Faucheur, F., and N. Bitar, "Content Distribution Network Interconnection (CDNI) Problem Statement", RFC 6707, September 2012.

[RFC6770] Bertrand, G., Stephan, E., Burbridge, T., Eardley, P., Ma, K., and G. Watson, "Use Cases for Content Delivery Network Interconnection", RFC 6770, November 2012.

Appendix A. Requirements

A.1. Compliance with cdni-requirements

This section discusses compliance of the present specification against all the relevant requirements of [I-D.ietf-cdni-requirements].

[Editor's Note: we may want to re-structure this into a table that would more clearly show compliance level]

A.1.1. General requirements

Some of the general CDNI requirements defined in [I-D.ietf-cdni-requirements] are not applicable to the CDNI Logging Interface [GEN-5, GEN-6, GEN-7, GEN-8, GEN-9, GEN-12].

The Logging Interface does not define any new protocols [GEN-1], does not require any change or upgrade on the user agent [GEN-2] or on the Content Service Provider side [GEN-3]. Also, no intra-CDN information is necessary [GEN-4] and the CDNI Logging Interface supports any interconnection topology [GEN-10]. However, The CDNI Logging Interface does not define a specific loop avoidance mechanism [GEN-11], but the exchange of logs is usually done in a point to point manner between two well identified entities situated respectively in the uCDN and the dCDN.

The CDNI Loggin Interface supports specific logging for the HTTP Adaptive Streaming content. [I-D.brandenburg-cdni-has] offers more details about particular logging fields required for HTTP Adaptive Streaming.

A.1.2. Logging Interface requirements

Reliable transfer is achieved by the transport protocol: the logging information is transmitted over HTTP running over TCP [LI-1].

The CDNI Logging Interface supports logs for all content deliveries both complete and incomplete performed by the dCDN on behalf of the uCDN [LI-2].

The CDNI Logging Interface does not impose any restrictions related to the transmission of logs generated by intermediary CDNs. The dCDN formats internally all the final logging files, including those received from intermediary CDNs and the files locally generated. The dCDN then sends all required logging files to the uCDN [LI-3].

The ATOM feed allows the uCDN to trigger the download of logging files whenever needed [LI-4].

The uCDN can pull logging files from the dCDN whenever a new file is available. The timing constraints for the generation of the logging files are to be defined offline, since the CDNI Logging Interface does not include a negotiation mechanism for the frequency of logging file generation. Note that the current version of this document refers strictly to non real-time logging [LI-5].

Section Section 3.4 describes the CDNI Logging Records and the possible fields that can be included in a record [LI-6].

As a transport mechanism, the CDNI Logging Interface uses the ATOM protocol over HTTP (or HTTPS) [LI-7].

A CDN can query another CDN for relevant current logging files by using the ATOM feed that allows to check for newly published content. Note that the current version of this document refers strictly to non real-time logging [LI-8].

The current version of the document does not specify any mechanisms for producing aggregate / summarized logs, but the exchanged logging files provide all information that is necessary to the uCDN in order to obtain aggregated logs. Future versions might include such mechanisms [LI-9].

No logging of performance data or consumed resources for the dCDN itself or any other cascaded CDN is included in the current version of the document. Future versions of this document might define such information [LI-10, LI-11, LI-12].

The current version of the document specifies the logging information related strictly to the delivery process. Logging files for any other functionalities (e.g., content purge, request routing events etc.) might be taken into account in future versions [LI-13].

Extensibility, the logging and exchange of proprietary information fields are detailed in Section 6 IANA Considerations [LI-14, LI-15].

The ATOM protocol allows the dCDN to publish the list of available resources (i.e. logging files) [LI-16].

Section 3.4 provides details about the fields of the HTTP Adaptive Streaming specific logging records, including the Content Collection Identifier (s-ccid) and Session Identifier (s-sid) [LI-17].

A.1.3. Security requirements

[SEC-3, SEC-5] are not applicable to the CDNI Loggin Interface, all remaining security requirements are addressed as discussed in Section 7.

A.2. Considerations on CDNI Logging Applicability

This section discusses a number of considerations related to the applicability of the CDNI Logging interface as specified in the present document.

[Editor's note: How do we incorporate this info into the I-D: in appendix? in main body? does it remain after publication or is temporary?]

A.2.1. Timeliness

Some applications consuming CDNI Logging information, such as accounting or trend analytics, only require logging information to be available with a timeliness of the order of a day or the hour. This document focuses on addressing this requirement.

Some applications consuming CDNI Logging information, such as real-time analytics, require logging information to be available in real-time (i.e. of the order of a second after the corresponding event). This document leaves this requirement out of scope.

A.2.2. Reliability

CDNI logging information must be transmitted reliably. The transport protocol should contain an anti-replay mechanism.

A.2.3. Security

CDNI logging information exchange must allow authentication, integrity protection, and confidentiality protection. Also, a non-repudiation mechanism is mandatory, the transport protocol should support it.

A.2.4. Scalability

CDNI logging information exchange must support large scale information exchange, particularly so in the presence of HTTP Adaptive Streaming.

For example, if we consider a client pulling HTTP Progressive Download content with an average duration of 10 minutes, this represents 1/600 CDNI delivery Logging Records per second. If we assume the dCDN is simultaneously serving 100,000 such clients on behalf of the uCDN, the dCDN will be generating 167 Logging Records per second to be communicated to the uCDN over the CDNI Logging interface. Or equivalently, if we assume an average delivery rate of 2Mb/s, the dCDN generates 0.83 CDNI Logging Records per second for every Gb/s of streaming on behalf of the uCDN.

For example, if we consider a client pulling HAS content and receiving a video chunk every 2 seconds, a separate audio chunck every 2 seconds and a refreshed manifest every 10 seconds, this represents 1.1 delivery Logging Record per second. If we assume the dCDN is simultaneously serving 100,000 such clients on behalf of the uCDN, the dCDN will be generating 110,000 Logging Records per second to be communicated to the uCDN over the CDNI Logging interface. Or equivalently, if we assume an average delivery rate of 2Mb/s, the dCDN generates 550 CDNI Logging Records per second for every Gb/s of streaming on behalf of the uCDN.

A.2.5. Consistency between CDNI Logging and CDN Logging

There are benefits in using a CDNI logging format as close as possible to intra-CDN logging format commonly used in CDNs today in order to minimize systematic translation at CDN/CDNI boundary.

A.2.6. Dispatching/Filtering

When a CDN is acting as a dCDN for multiple uCDNs, the dCDN needs to dispatch each CDNI Logging Record to the uCDN that redirected the corresponding request. The CDNI Logging format need to allow, and possibly facilitate, such a dispatching.

Authors' Addresses

Francois Le Faucheur (editor)
Cisco Systems
E.Space Park - Batiment D
6254 Allee des Ormes - BP 1200
Mougins cedex 06254
FR

Phone: +33 4 97 23 26 19
Email: flefauch@cisco.com

Gilles Bertrand (editor)
Orange
38-40 rue du General Leclerc
Issy les Moulineaux 92130
FR

Phone: +33 1 45 29 89 46
Email: gilles.bertrand@orange.com

Iuniana Oprescu (editor)
Orange
38-40 rue du General Leclerc
Issy les Moulineaux 92130
FR

Phone: +33 6 89 06 92 72
Email: iuniana.oprescu@orange.com

Roy Peterkofsky
Skytide, Inc.
One Kaiser Plaza, Suite 785
Oakland CA 94612
USA

Phone: +01 510 250 4284
Email: roy@skytide.com