Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: December 10, 2016

F. Le Faucheur, Ed.

G. Bertrand, Ed.

I. Oprescu, Ed.

R. Peterkofsky
Google Inc.

June 8, 2016

CDNI Logging Interface
draft-ietf-cdni-logging-27

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 December 10, 2016.

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 [RFC7336] identify a Logging interface,
- Section 8 of [RFC7337] 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),
- 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 [RFC7336], and extend it with the additional terms defined below.

Intra-CDN Logging information: logging information generated and collected within a CDN. The format of the Intra-CDN Logging information may be different to the format of the CDNI Logging information.

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 Uniform Resource Identifier (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 or displaying content delivery information in a timely fashion with respect to the corresponding deliveries. Monitoring typically includes 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", "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 CDNI Logging fields are to be communicated to the uCDN for a given task performed by the dCDN or 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 customization by the uCDN of CDNI Logging information is outside the scope of this document and left for further study. Until such a mechanism is available, the uCDN and dCDN are expected to agree off-line on what exact set of CDNI Logging information is to be provided by the dCDN to the uCDN, and to rely on management plane actions to configure the CDNI Logging functions in the dCDN to generate this information set and in the uCDN to expect this information set.

- generation and collection by the dCDN of the intra-CDN 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. This document exclusively specifies non-real-time exchange of Logging information. Closer to real-time exchange of Logging information (say sub-minute or sub-second) is outside the scope of the present document and left for further study. This document exclusively specifies exchange of Logging information related to content delivery. Exchange of Logging information related to operational events (e.g., dCDN request routing function unavailable, content acquisition failure by dCDN) for audit or operational reactive adjustments by uCDN is outside the scope of the present document and left for further study.

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

- o generation and collection by the uCDN of Intra-CDN 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.

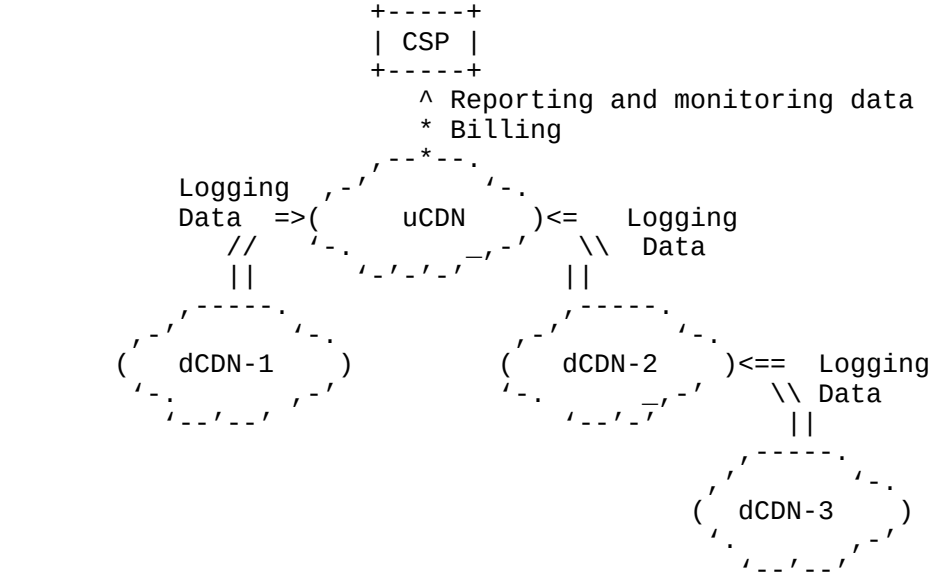
- o communication by the uCDN to the dCDN of the Logging information collected by the uCDN relevant to the dCDN. For example, the dCDN might potentially benefit from this information for security auditing or content acquisition troubleshooting. This is outside the scope of this document and left for 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 four 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, dCDN-2 has a CDNI interconnection with dCDN-3, where dCDN-2 is acting as an upstream CDN relative to dCDN-3. 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, the uCDN uses the Logging information:

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

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

For instance, the uCDN merges Logging information, 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 the CSP's behalf. The uCDN may also provide Logging information as raw log files to the CSP, so that the CSP can use its own logging analysis tools.



===> CDNI Logging Interface
***> outside the scope of CDNI

Figure 1: Interactions in CDNI Logging Reference Model

A downstream CDN relative to uCDN (e.g., dCDN-2) integrates the relevant Logging information obtained from its own downstream CDNs (i.e., 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. Such aggregation is further discussed in Section 3.7.

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 information 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 formats that are suitable for use inside CDNs and also are close to the intra-CDN Logging 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 an example and varies depending on the specific environments. For example, there may be more or fewer instantiations of each entity (e.g., 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.

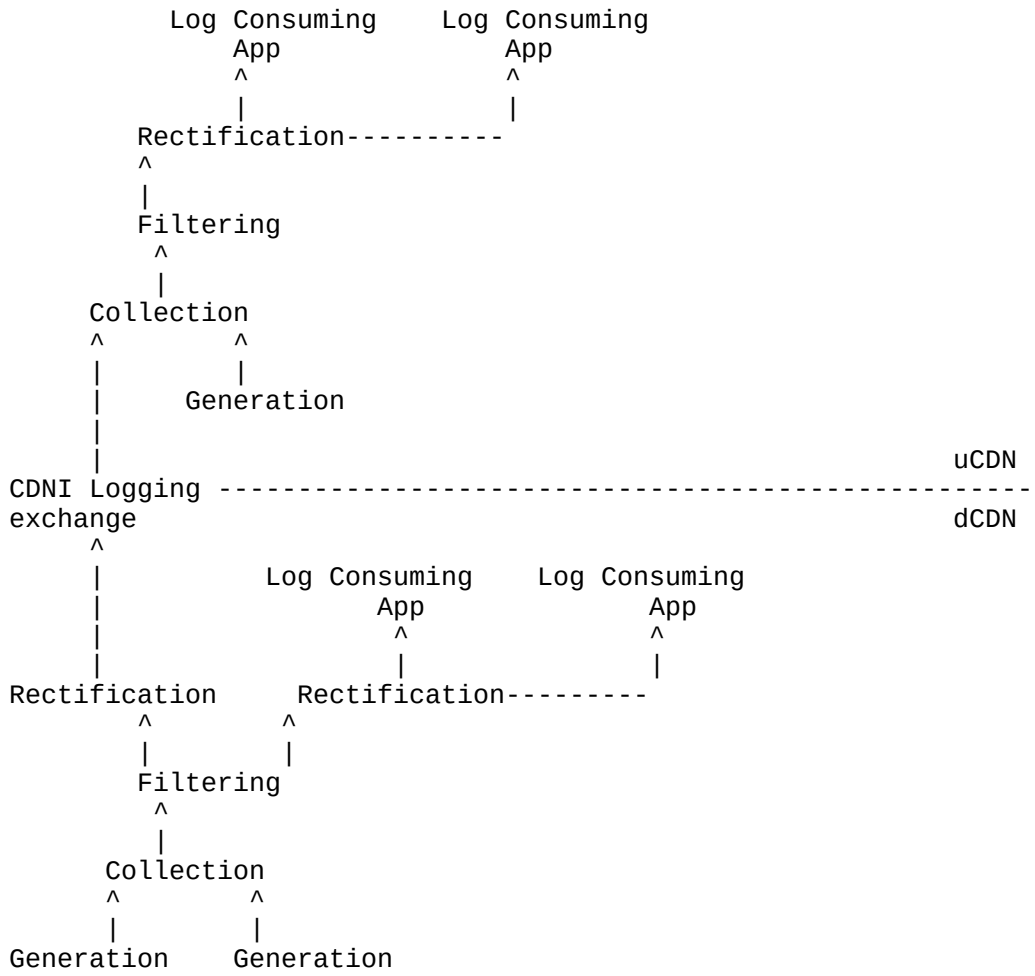


Figure 2: CDNI Logging in the overall Logging Chain

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. Logging information is 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 retention duration for Logging information, and/or potentially on a maximum volume of Logging information that the dCDN ought to keep. If this volume is exceeded, the dCDN is expected to alert the uCDN but may not keep more Logging information for the considered time period. In addition, CDNs may aggregate Logging information and transmit only summaries for some categories of operations instead of the full Logging information. Note that such aggregation leads to an information loss, which may be problematic for some usages of the Logging information (e.g., debugging).

[RFC6983] 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 [RFC6983], the Logging information for per-chunk 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 aggregate per-session logs for HAS delivery are for further study and outside the scope of this document.

2.2.2. Logging Collection

This is the process that continuously collects Logging information 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 from the dCDNs through the CDNI Logging interface.

2.2.3. Logging Filtering

A CDN may be required to only present different subsets 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 Logging 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 out 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 subset of Logging information 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 each Surrogate may be confidential. Therefore, the Logging information needs to be protected so that data such as Surrogates' hostnames are 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 information 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 information of this period will be available only after the end of the session, which delays the Logging information generation. A simple approach is to provide the complete Logging Record for a session in the Logging Period of the session end.

A Logging rectification/update mechanism could be useful to reach a good trade-off between the Logging information generation delay and the Logging information accuracy.

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 aggregate logs have been generated directly by the

log-generating entities, 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 [RFC6983], this document does not define exchange of such aggregate logs on the CDNI Logging interface. We note that this is for further study and outside the scope of this document.

2.2.5. Log-Consuming Applications

2.2.5.1. Maintenance/Debugging

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

To detect faults, Logging information needs to report success and failure of CDN delivery operations. The uCDN can summarize such information into KPIs. For instance, Logging information needs to allow the computation of the number of times, during a given time period, that content delivery related to a specific service succeeds/fails.

Logging information enables the CDN providers to identify and troubleshoot performance degradations. In particular, Logging information enables tracking 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.

Some of these maintenance and debugging applications only require aggregate logging information highly compatible with use of anonymization of IP addresses (as supported by the present document and specified in the definition of the c-groupid field under Section 3.4.1). However, in some situations, it may be useful, where compatible with privacy protection, to access some CDNI Logging Records containing full non-anonymized IP addresses. This is allowed in the definition of the c-groupid (under Section 3.4.1), with very significant privacy protection limitations that are discussed in the definition of the c-groupid field. For example, this may be useful for detailed fault tracking of a particular end user content delivery issue. Where there is a hard requirement by uCDN or CSP to associate a given enduser to individual CDNI Logging Records (e.g., to allow a-posteriori analysis of individual delivery for example in situations of performance-based penalties), instead of using

aggregates containing a single client as discussed in the c-groupid field definition, an alternate approach is to ensure that a client identifier is embedded in the request fields that can be logged in a CDNI Logging Record (for example by including the client identifier in the URI query string or in a HTTP Header). That latter approach offers two strong benefits: first, the aggregate inside the c-groupid can contain more than one client, thereby ensuring stronger privacy protection; second, it allows a reliable identification of the client while IP address does not in many situations (e.g., behind NAT, where dynamic IP addresses are used and reused,...). However, care SHOULD be taken that the client identifiers exposed in other fields of the CDNI Records cannot themselves be linked back to actual users.

2.2.5.2. Accounting

Logging information 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 goals of analytics include gathering any relevant information in order to be able to develop statistics on content download, analyze user behavior, and monitor the performance and quality of content delivery. For instance, Logging information 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. Content Protection

The goal of content protection is to prevent and monitor unauthorized access, misuse, modification, and denial of access to a content. A set of information is logged in a CDN 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. Notions common to multiple Log Consuming Applications

2.2.5.5.1. 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.

2.2.5.5.2. 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 the 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 most popularly requested contents (during a given day/week/month)

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

Additional KPIs can be computed from other sources of information than the Logging information, 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 document.

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 a timely manner. By contrast, some other KPIs, such as those 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

ALPHA = %x41-5A / %x61-7A ; A-Z / a-z

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

VCHAR = %x21-7E ; visible (printing) characters

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].

partial-time = as specified in [RFC3339].
```

The present document also defines the following additional rules:

```
ADDRESS = IPv4address / IPv6address

ALPHANUM = ALPHA / DIGIT

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

   ; Dates are encoded as "full-date" specified in [RFC3339].
```

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

NAMEFORMAT = ALPHANUM *(ALPHANUM / "_" / "-")

QSTRING = DQUOTE *(NDQUOTE / PCT-ENCODED) DQUOTE

NDQUOTE = %x20-21 / %x23-24 / %x26-7E / UTF8-2 / UTF8-3 / UTF8-4

   ; whereby a DQUOTE is conveyed inside a QSTRING unambiguously
   by escaping it with PCT-ENCODED.

PCT-ENCODED = "%" HEXDIG HEXDIG

   ; percent encoding is used for escaping octets that might be
   possible in HTTP headers such as bare CR, bare LF, CR LF, HTAB,
   SP or null.  These octets are rendered with percent encoding in
   ABNF as specified by [RFC3986] in order to avoid considering
   them as separators for the logging records.

NHTABSTRING = 1*(SP / VCHAR)

TIME = partial-time

USER-COMMENT = * (SP / VCHAR / UTF8-2 / UTF8-3 / UTF8-4)
```

3.2. CDNI Logging File Structure

As defined in Section 1.1: a CDNI Logging Field is as an atomic logging information element, a CDNI Logging Record is a collection of CDNI Logging fields containing all logging information corresponding to a single logging event, and a CDNI Logging File contains 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.

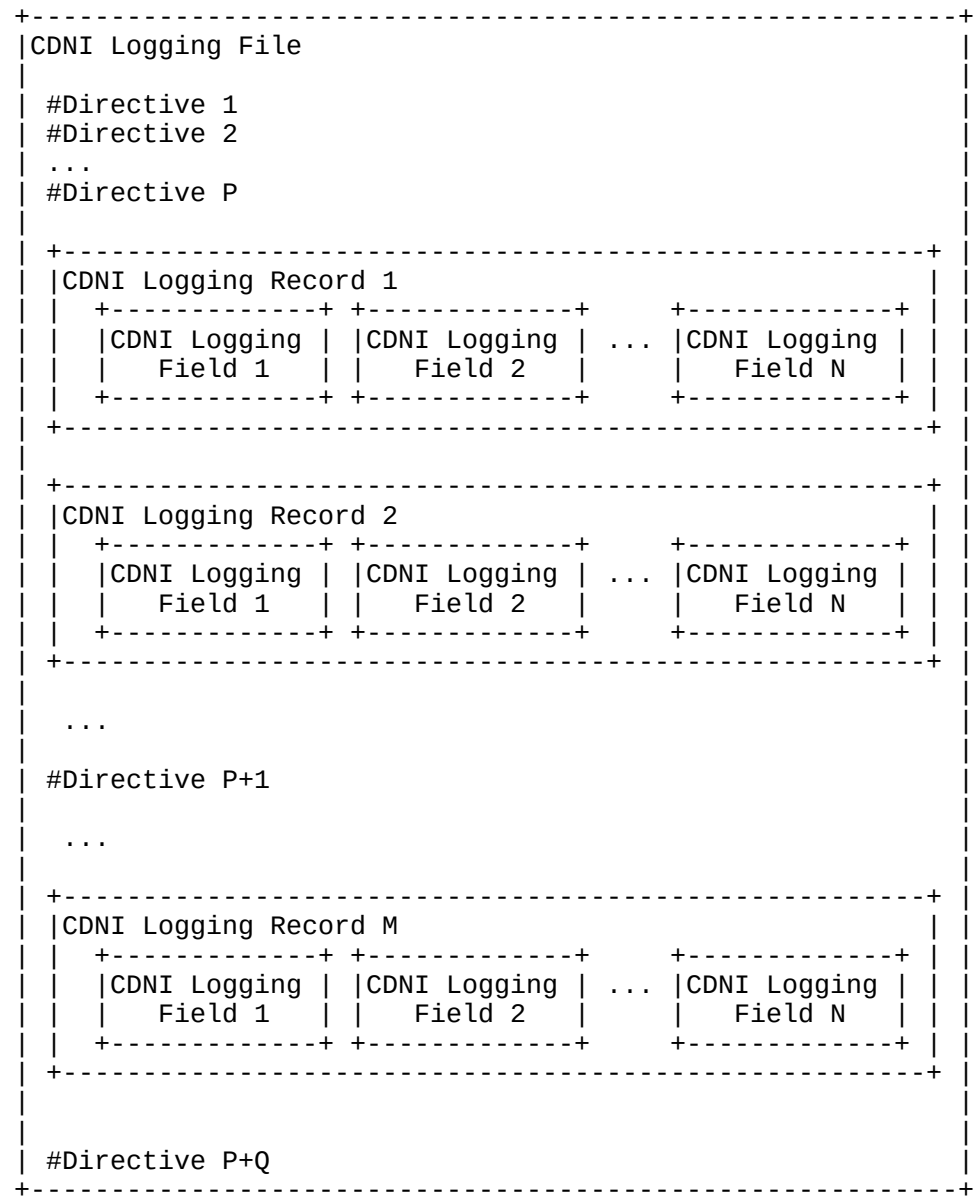


Figure 3: Structure of Logging Files

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 the CDNI Logging interface MUST comply with the present document. The W3C Extended Log File Format was used as a starting point, reused where possible and expanded when necessary.

Using a format that resembles the W3C Extended Log File Format is intended to keep CDNI logging format close to the intra-CDN Logging information format commonly used in CDNs today, thereby minimizing systematic translation at CDN/CDNI boundary.

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 Logging File has a specific structure. It always starts with a directive line and the first directive it contains MUST be the version.

The directive lines form together a group that contains at least one directive line. Each directives group is followed by a group of logging records. The records group contains zero or more actual logging record lines about the event being logged. A record line consists of the values corresponding to all or a subset of the possible Logging fields defined within the scope of the record-type directive. These values MUST appear in the order defined by the fields directive.

Note that future extensions MUST be compliant with the previous description. The following examples depict the structure of a CDNILOGFILE as defined currently by the record-type "cdni_http_request_v1."

```
DIRLINE = "#" directive CRLF

DIRGROUP = 1*DIRLINE

RECLINE = <any subset of record values that match what is expected
according to the fields directive within the immediately preceding
DIRGROUP>
```

```
RECGROUP = *RECLINE

CDNILOGFILE = 1*(DIRGROUP RECGROUP)
```

3.3. CDNI Logging Directives

A CDNI Logging directive line contains the directive name followed by ":" HTAB and the directive value.

Directive names MUST be of the format NAMEFORMAT. All directive names MUST be registered in the CDNI Logging Directives Names registry. Directive names are case-insensitive as per the basic ABNF([RFC5234]). Unknown directives MUST be ignored. Directive values can have various formats. All possible directive values for the record-type "cdni_http_request_v1" are further detailed in this section.

The following example shows the structure of a directive and enumerates strictly the directive values presently defined in the version "cdni/1.0" of the CDNI Logging File.

```
directive = DIRNAME ":" HTAB DIRVAL

DIRNAME = NAMEFORMAT

DIRVAL = NHTABSTRING / QSTRING / host / USER-COMMENT / FIENAME *
(HTAB FIENAME) / 64HEXDIG
```

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

- version:
  - format: NHTABSTRING
  - directive value: indicates the version of the CDNI Logging File format. The entity transmitting a CDNI Logging File as per the present document MUST set the value to "cdni/1.0". In the future, other versions of CDNI Logging File might be specified; those would use a value different to "cdni/1.0" allowing the entity receiving the CDNI Logging File to identify the corresponding version. CDNI Logging File versions are case-insensitive as per the basic ABNF([RFC5234]).
  - occurrence: there MUST be one and only one instance of this directive per CDNI Logging File. It MUST be the first line of the CDNI Logging File.

      * example: "version: HTAB cdni/1.0".

   o UUID:

      * format: NHTABSTRING

      * directive value: this a Uniform Resource Name (URN) from the Universally Unique IDentifier (UUID) URN namespace specified in [RFC4122]). The UUID contained in the URN uniquely identifies the CDNI Logging File.

      * occurrence: there MUST be one and only one instance of this directive per CDNI Logging File.

      * example: "UUID: HTAB NHTABSTRING".

   o 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 exactly one instance of this directive per CDNI Logging File. This directive MAY be included by the dCDN. It MUST NOT be included or modified by the uCDN.

      * example: "claimed-origin: HTAB host".

   o established-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 exactly one instance of this directive per CDNI Logging File. This directive MAY be added by the uCDN (e.g., before storing the CDNI Logging File). It MUST NOT be included by the dCDN. The mechanisms used by the

uCDN to establish and validate the entity responsible 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 transport layer of the CDNI Logging File pull mechanism (Section 4.2).

* ABNF example: "established-origin: HTAB host".

- remark:
  * format: USER-COMMENT
  * directive value: this contains comment information. Data contained in this field is to be ignored by analysis tools.
  * occurrence: there MAY be zero, one or any number of instance of this directive per CDNI Logging File.
  * example: "remark: HTAB USER-COMMENT".
- record-type:
  * format: NAMEFORMAT
  * 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 6.3). For example this may be "cdni_http_request_v1" as specified in Section 3.4.1. CDNI Logging record-types are case-insensitive as per the basic ABNF([RFC5234]).
  * occurrence: there MUST be at least one instance of this directive per CDNI Logging File. The first instance of this directive MUST precede a fields directive and MUST precede all CDNI Logging Records.
  * example: "record-type: HTAB cdni_http_request_v1".
- fields:
  * format: FIENAME *(HTAB FIENAME) ; where FIENAME can take any CDNI Logging field name registered in the CDNI Logging Field Names registry (Section 6.4) that is valid for the record type specified in the record-type directive.

  * directive value: this lists the names of all the fields for which a value is to appear in the CDNI Logging Records that follow the instance of this directive (until another instance of this directive). The names of the fields, as well as their occurrences, MUST comply with the corresponding rules specified in the document referenced in the CDNI Logging Record-types registry (Section 6.3) for the corresponding CDNI Logging record-type.

  * 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 appear before any CDNI Logging Record for this record-type. One situation where more than one instance of the fields directive can appear within a given CDNI Logging File, is when there is a change, in the middle of a fairly large logging period, in the agreement between the uCDN and the dCDN about the set of fields that are to be exchanged. The multiple occurrences allow records with the old set of fields and records with the new set of fields to be carried inside the same Logging File.

  * example: "fields: HTAB FIENAME * (HTAB FIENAME)".

- SHA256-hash:

  * format: 64HEXDIG

  * 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 valid SHA256-hash value is included in this directive by the entity that transmits the CDNI Logging File. It MUST be computed by applying the SHA-256 ([RFC6234]) cryptographic hash function on the CDNI Logging File, including all the directives and logging records, up to the SHA256-hash directive itself, excluding the SHA256-hash directive itself. The SHA256-hash value MUST be represented as a US-ASCII encoded hexadecimal number, 64 digits long (representing a 256 bit hash value). The entity receiving the CDNI Logging File also computes in a similar way the SHA-256 hash on the received CDNI Logging File and compares this hash to the value of the SHA256-hash directive. If the two values are equal, then the received CDNI Logging File is to be considered non-corrupted. If the two values are different, the received CDNI Logging File is to 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 a non-corrupted CDNI Logging File adds an established-origin directive, it MUST then recompute and update the SHA256-hash directive so it also protects the added established-origin directive.

* occurrence: there MUST be zero or exactly one instance of this directive. There SHOULD be exactly 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 Feed (Section 4.1) leveraging ATOM extensions such as those proposed in [I-D.snell-atompub-link-extensions]. When present, the SHA256-hash field MUST be the last line of the CDNI Logging File.

* example: "SHA256-hash: HTAB 64HEXDIG".

An uCDN-side implementation of the CDNI Logging interface MUST ignore a CDNI Logging File that does not comply with the occurrences specified above for each and every directive. For example, an uCDN-side implementation of the CDNI Logging interface receiving a CDNI Logging file with zero occurrence of the version directive, or with two occurrences of the SHA256-hash, MUST ignore this CDNI Logging File.

An entity receiving a CDNI Logging File with a value set to "cdni/1.0" MUST process the CDNI Logging File as per the present document. An entity receiving a CDNI Logging File with a value set to a different value MUST process the CDNI Logging File as per the specification referenced in the CDNI Logging File version registry (see Section 6.1) if the implementation supports this specification and MUST ignore the CDNI Logging File otherwise.

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 an 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 Request Logging Record as specified in Section 3.4.1.

A CDNI Logging Record contains the corresponding values for the fields that are enumerated in the last fields directive before the current log line. Note that the order in which the field values appear is dictated by the order of the fields names in the fields directive. There SHOULD be no dependency between the various fields values.

3.4.1. HTTP Request Logging Record

This section defines the CDNI Logging Record of record-type "cdni_http_request_v1". It is applicable to content delivery performed by the dCDN using HTTP/1.0([RFC1945]), HTTP/1.1([RFC7230],[RFC7231], [RFC7232], [RFC7233], [RFC7234], [RFC7235]) or HTTPS ([RFC2818], [RFC7230]). We observe that, in the case of HTTPS delivery, there may be value in logging additional information specific to the operation of HTTP over TLS and we note that this is outside the scope of the present document and may be addressed in a future document defining another CDNI Logging Record or another version of the HTTP Request Logging Record.

The "cdni_http_request_v1" record-type is also expected to be applicable to HTTP/2 [RFC7540] since a fundamental design tenet of HTTP/2 is to preserve the HTTP/1.1 semantics. We observe that, in the case of HTTP/2 delivery, there may be value in logging additional information specific to the additional functionality of HTTP/2 (e.g., information related to connection identification, to stream identification, to stream priority and to flow control). We note

that such additional information is outside the scope of the present document and may be addressed in a future document defining another CDNI Logging Record or another version of the HTTP Request Logging Record.

The "cdni_http_request_v1" record-type 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, which MUST be expressed in Coordinated Universal Time (UTC), 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-groupid:
  - format: NHTABSTRING
  - field value: an opaque identifier for an aggregate set of clients, derived from the client IPv4 or IPv6 address in the request received by the Surrogate and/or other network-level

identifying information. The c-groupid serves to group clients into aggregates. Example aggregates include civil geolocation information (the country, second-level administrative division, or postal code from which the client is presumed to make the request based on a geolocation database lookup) or network topological information (e.g., the BGP AS number announcing the prefix containing the address). The c-groupid MAY be structured e.g., US/TN/MEM/38138. Agreement between the dCDN and the uCDN on a mapping between IPv4 and IPv6 addresses and aggregates is presumed to occur out-of-band. The aggregation mapping SHOULD be chosen such that each aggregate contains more than one client.

+ When the aggregate is chosen so that it contains a single client (e.g., to allow more detailed analytics, or to allow a-posteriori analysis of individual delivery for example in situations of performance-based penalties) the c-groupid MAY be structured where some elements identify aggregates and one element identifies the client, e.g., US/TN/MEM/38138/43a5bdd6-95c4-4d62-be65-7410df0021e2. In the case where the aggregate is chosen so that it contains a single client:

  - the element identifying the client SHOULD be algorithmically generated (from the client IPv4 or IPv6 address in the request received by the Surrogate and/or other network-level identifying information) in a way that SHOULD NOT be linkable back to the global addressing context and that SHOULD vary over time (to offer protection against long term attacks).

  - It is RECOMMENDED that the mapping varies at least once every 24 hours.

  - The algorithmic mapping and variation over time can, in some cases, allow the uCDN (with the knowledge of the algorithm and time variation and associated attributes and keys) to reconstruct the actual client IPv4 or IPv6 address and/or other network-level identifying information when required (e.g., to allow a-posteriori analysis of individual delivery for example in situations of performance-based penalties). However, these enduser addresses SHOULD only be reconstructed on-demand and the CDNI Logging File SHOULD only be stored with the anonymised c-groupid value.

  - Allowing reconstruction of client address information carries with it grave risks to end-user privacy. Since

the c-groupid is in this case equivalent in identification power to a client IP address, its use may be restricted by regulation or law as personally identifiable information. For this reason, such use is NOT RECOMMENDED.

- One method for mapping that MAY be be supported by implementations relies on a symmetric key that is known only to the uCDN and dCDN and HMAC-based Extract-and-Expand Key Derivation Function (HKDF) key derivation ([RFC5869]), as will be used in TLS 1.3 ([I-D.ietf-tls-rfc5246-bis]). When that method is used:

  o The uCDN and dCDN need to agree on the "salt" and "input keying material", as described in Section 2.2 of [RFC5869] and the initial "info" parameter (which could be something like the business names of the two organizations in UTF-8, concatenated), as described in Section 2.3 of [RFC5869]. The hash SHOULD be either SHA-2 or SHA-3 [SHA-3] and the encryption algorithm SHOULD be 128-bit AES [AES] in Galois Counter Mode (GCM) [GCM] (AES-GCM) or better. The PRK SHOULD be chosen by both parties contributing alternate random bytes until sufficient length exists. After the initial setup, client-information can be encrypted using the key generated by the "expand" step of Section 2.3 of [RFC5869]. The encrypted value SHOULD be hex encoded or base64 encoded (as specified in section 4 of [RFC4648]). At the agreed-upon expiration time, a new key SHOULD be generated and used. New keys SHOULD be indicated by prefixing the key with a special character such as exclamation point. In this way, shorter lifetimes can be used as needed.

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

o 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 method of the request received by the Surrogate. In the case of HTTP delivery, this is the HTTP method in the request.
  - occurrence: There MUST be one and only one instance of this field.
- cs-uri:
  - format: NHTABSTRING
  - field value: this is the "effective request URI" of the request received by the Surrogate as specified in [RFC7230]. It complies with the "http" URI scheme or the "https" URI scheme as specified in [RFC7230]). Note that cs-uri can be privacy sensitive. In that case, and where appropriate, u-uri could be used instead of cs-uri.
  - occurrence: there MUST be zero or exactly one instance of this field.
- u-uri:
  - format: NHTABSTRING

  * field value: this is a complete URI, derived from the "effective request URI" ([RFC7230]) 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 [RFC7230] 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 Status-Code in the response from the Surrogate. In the case of HTTP delivery, this is the HTTP Status-Code in the HTTP response.

  * 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 response sent by the Surrogate in response to the request. In the case of HTTP delivery, this includes the bytes of the Status-Line,

the bytes of the HTTP headers and the bytes 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 or the bytes of the HTTP headers.
  * occurrence: there MUST be zero or exactly one instance of this field.
- cs(insert_HTTP_header_name_here):
  * format: QSTRING
  * field value: the value of the HTTP header (identified by the insert_HTTP_header_name_here in the CDNI Logging field name) as it appears in the request processed by the Surrogate, but prepended by a DQUOTE and appended by a DQUOTE. For example, when the CDNI Logging field name (FIENAME) listed in the preceding 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, but prepended by a DQUOTE and appended by a DQUOTE. If the HTTP header as it appeared in the request processed by the Surrogate contains one or more DQUOTE, each DQUOTE MUST be escaped with percent encoding. For example, if the HTTP header contains My_Header"value", then the field value of the cs(insert_HTTP_header_name_here) is "My_Header%x22value%x22". The entity transmitting the CDNI Logging File MUST ensure that the respective insert_HTTP_header_name_here of the cs(insert_HTTP_header_name_here) listed in the fields directive comply with HTTP specifications. In particular, this field name does not include any HTAB, since this would prevent proper parsing of the fields directive by the entity receiving the CDNI Logging File.
  * occurrence: there MAY be zero, one or any number of instance of this field.
- sc(insert_HTTP_header_name_here):

  * format: QSTRING

  * field value: the value of the HTTP header (identified by the insert_HTTP_header_name_here in the CDNI Logging field name) as it appears in the response issued by the Surrogate to serve the request, but prepended by a DQUOTE and appended by a DQUOTE. If the HTTP header as it appeared in the request processed by the Surrogate contains one or more DQUOTE, each DQUOTE MUST be escaped with percent encoding. For example, if the HTTP header contains My_Header"value", then the field value of the sc(insert_HTTP_header_name_here) is "My_Header%x22value%x22". The entity transmitting the CDNI Logging File MUST ensure that the respective insert_HTTP_header_name_here of the cs(insert_HTTP_header_name_here) listed in the fields directive comply with HTTP specifications. In particular, this field name does not include any HTAB, since this would prevent proper parsing of the fields directive by the entity receiving the CDNI Logging File.

  * occurrence: there MAY be zero, one or any number of instances of this field. For a given insert_HTTP_header_name_here, there MUST be zero or exactly one instance of this field.

- s-ccid:

  * format: QSTRING

  * field value: this contains the value of the Content Collection IDentifier (CCID) associated by the uCDN to the content served by the Surrogate via the CDNI Metadata interface ([I-D.ietf-cdni-metadata]), prepended by a DQUOTE and appended by a DQUOTE. If the CCID conveyed in the CDNI Metadata interface contains one or more DQUOTE, each DQUOTE MUST be escaped with percent encoding. For example, if the CCID conveyed in the CDNI Metadata interface is My_CCIDD"value", then the field value of the s-ccid is "My_CCID%x22value%X22".

  * occurrence: there MUST be zero or exactly one instance of this field. For a given insert_HTTP_header_name_here, 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 (SID) generated by the dCDN for a specific HTTP session, prepended by a DQUOTE and appended by a DQUOTE. In particular,

for HTTP Adaptive Streaming (HAS) session, the Session IDentifier 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 [RFC6983] for more discussion on the concept of Session IDentifier in the context of HAS. If the SID conveyed contains one or more DQUOTE, each DQUOTE MUST be escaped with percent encoding. For example, if the SID is My_SID"value", then the field value of the s-sid is "My_SID%x22value%x22".

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

o s-cached:

* format: 1DIGIT

* field value: this characterises whether the Surrogate served the request using content already stored on its local cache or not. The allowed values are "0" (for miss) and "1" (for hit). "1" MUST be used when the Surrogate did 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.

CDNI Logging field names are case-insensitive as per the basic ABNF([RFC5234]). The "fields" directive corresponding to a HTTP Request Logging Record MUST contain all the fields names 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 MAY be zero, one or any number of instances of this field". The set of such field names actually listed in the "fields" directive is selected by the CDN 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; if the value for the field is not available, this MUST be conveyed via a dash character ("-").

The fields names 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.

Logging some specific fields from HTTP requests and responses can introduce serious security and privacy risks. For example, cookies will often contain (months) long lived token values that can be used to log into a service as the relevant user. Similar values may be included in other header fields or within URLs or elsewhere in HTTP requests and responses. Centralising such values in a CDNI Logging File can therefore represent a significant increase in risk both for the user and the web service provider, but also for the CDNs involved. Implementations ought therefore to attempt to lower the probability of such bad outcomes e.g. by only allowing a configured set of headers to be added to CDNI Logging Records, or by not supporting wildcard selection of HTTP request/response fields to add. Such mechanisms can reduce the probability that security (or privacy) sensitive values are centralised in CDNI Logging Files. Also, when agreeing on which HTTP request/response fields are to be provided in CDNI Logging Files, the uCDN and dCDN administrators ought to consider these risks. Furthermore, CDNs making use of c-groupid to identify an aggregate of clients rather than individual clients ought to realize that by logging certain header fields they may create the possibility to re-identify individual clients. In these cases heeding the above advice, or not logging header fields at all, is particularly important if the goal is to provide logs that do not identify individual clients."

A dCDN-side implementation of the CDNI Logging interface MUST implement all the following Logging fields in a CDNI Logging Record of record-type "cdni_http_request_v1", and MUST support the ability to include valid values for each of them:

- date
- time
- time-taken
- c-groupid

- s-ip
- s-hostname
- s-port
- cs-method
- cs-uri
- u-uri
- protocol
- sc-status
- sc-total-bytes
- sc-entity-bytes
- cs(insert_HTTP_header_name_here)
- sc(insert_HTTP_header_name_here)
- s-cached

A dCDN-side implementation of the CDNI Logging interface MAY support the following Logging fields in a CDNI Logging Record of record-type "cdni_http_request_v1":

- s-ccid
- s-sid

If a dCDN-side implementation of the CDNI Logging interface supports these fields, it MUST support the ability to include valid values for them.

An uCDN-side implementation of the CDNI Logging interface MUST be able to accept CDNI Logging Files with CDNI Logging Records of record-type "cdni_http_request_v1" containing any CDNI Logging Field defined in Section 3.4.1 as long as the CDNI Logging Record and the CDNI Logging File are compliant with the present document.

In case an uCDN-side implementation of the CDNI Logging interface receives a CDNI Logging File with HTTP Request Logging Records that do not contain field values for exactly the set of field names actually listed in the preceding "fields" directive, the

implementation MUST ignore those HTTP Request Logging Records, and MUST accept the other HTTP Request Logging Records.

To ensure that the logging file is correct, the text MUST be sanitized before being logged. Null, bare CR, bare LF and HTAB have to be removed by escaping them through percent encoding to avoid confusion with the logging record separators.

3.5. CDNI Logging File Extension

The CDNI Logging File contains blocks of directives and blocks of corresponding records. The supported set of directives is defined relative to the CDNI Logging File Format version. The complete set of directives for version "cdni/1.0" are defined in Section 3.3. The directive list is not expected to require much extension, but when it does, the new directive MUST be defined and registered in the "CDNI Logging Directive Names" registry, as described in Figure 9, and a new version MUST be defined and registered in the "CDNI Logging File version" registry, as described in Section 6.2. For example, adding a new CDNI Logging Directive, e.g., "foo", to the set of directives defined for "cdni/1.0" in Section 3.3, would require registering both the new CDNI Logging Directive "foo" and a new CDNI Logging File version, e.g., "CDNI/2.0", which includes all of the existing CDNI Logging Directives of "cdni/1.0" plus "foo".

It is expected that as new logging requirements arise, the list of fields to log will change and expand. When adding new fields, the new fields MUST be defined and registered in the "CDNI Logging Field Names" registry, as described in Section 6.4, and a new record-type MUST be defined and registered in the "CDNI Logging record-types" registry, as described in Section 6.3. For example, adding a new CDNI Logging Field, e.g., "c-bar", to the set of fields defined for "cdni_http_request_v1" in Section 3.4.1, would require registering both the new CDNI Logging Field "c-bar" and a new CDNI record-type, e.g., "cdni_http_request_v2", which includes all of the existing CDNI Logging Fields of "cdni_http_request_v1" plus "c-bar".

3.6. CDNI Logging File Examples

Let us consider the upstream CDN and the downstream CDN labelled uCDN and dCDN-1 in Figure 1. When dCDN-1 acts as a downstream CDN for uCDN and performs content delivery on behalf of uCDN, dCDN-1 will include the CDNI Logging Records corresponding to the content deliveries performed on behalf of uCDN in the CDNI Logging Files for uCDN. An example CDNI Logging File communicated by dCDN-1 to uCDN is shown below in Figure 4.

```
#version:<HTAB>cdni/1.0<CRLF>

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

#claimed-origin:<HTAB>cdni-logging-entity.dcdn-1.example.com<CRLF>

#record-type:<HTAB>cdni_http_request_v1<CRLF>

#fields:<HTAB>date<HTAB>time<HTAB>time-taken<HTAB>c-groupid<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>US/TN/MEM/38138<HTAB>
GET<HTAB>
http://cdni-ucdn.dcdn-1.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>FR/PACA/NCE/06100<HTAB>
GET<HTAB>
http://cdni-ucdn.dcdn-1.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>US/TN/MEM/38138<HTAB>
GET<HTAB>
http://cdni-ucdn.dcdn-1.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>

#SHA256-hash:<HTAB> 64-hexadecimal-digit hash value <CRLF>
```

Figure 4: CDNI Logging File Example

If uCDN establishes by some means (e.g., via TLS authentication when pulling the CDNI Logging File) the identity of the entity from which it pulled the CDNI Logging File, uCDN can add to the CDNI Logging an established-origin directive as illustrated below:

```
#established-origin:<HTAB>cdni-logging-entity.dcdn-
1.example.com<CRLF>
```

As illustrated in Figure 2, uCDN will then ingest the corresponding CDNI Logging Records into its Collection process, alongside the Logging Records generated locally by the uCDN itself. This allows uCDN to aggregate Logging Records for deliveries performed by itself (through Records generated locally) as well as for deliveries performed by its downstream CDN(s). This aggregate information can then be used (after Filtering and Rectification, as illustrated in Figure 2) by Log Consuming Applications that take into account deliveries performed by uCDN as well as by all of its downstream CDNs.

We observe that the time between

1. when a delivery is completed in dCDN and

2. when the corresponding Logging Record is ingested by the Collection process in uCDN

depends on a number of parameters such as the Logging Period agreed to by uCDN and dCDN, how much time uCDN waits before pulling the CDNI Logging File once it is advertised in the CDNI Logging Feed, and the time to complete the pull of the CDNI Logging File. Therefore, if we consider the set of Logging Records aggregated by the Collection process in uCDN in a given time interval, there could be a permanent significant timing difference between the CDNI Logging Records received from the dCDN and the Logging Records generated locally. For example, in a given time interval, the Collection process in uCDN may be aggregating Logging Records generated locally by uCDN for deliveries performed in the last hour and CDNI Logging Records generated in the dCDN for deliveries in the hour before last.

Say, that for some reason (for example a Surrogate bug), dCDN-1 could not collect the total number of bytes of the responses sent by the Surrogate (in other words, the value for sc-total-bytes is not available). Then the corresponding CDNI Logging records would contain a dash character ("-") in lieu of the value for the sc-total-bytes field (as specified in Section 3.4.1). In that case, the CDNI Logging File that would be communicated by dCDN-1 to uCDN is shown below in Figure 5.

```
#version:<HTAB>cdni/1.0<CRLF>

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

#claimed-origin:<HTAB>cdni-logging-entity.dcdn-1.example.com<CRLF>

#record-type:<HTAB>cdni_http_request_v1<CRLF>

#fields:<HTAB>date<HTAB>time<HTAB>time-taken<HTAB>c-groupid<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>US/TN/MEM/38138<HTAB>
GET<HTAB>
http://cdni-ucdn.dcdn-1.example.com/video/movie100.mp4<HTAB>
HTTP/1.1<HTAB>200<HTAB>-<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>FR/PACA/NCE/06100<HTAB>
GET<HTAB>
http://cdni-ucdn.dcdn-1.example.com/video/movie118.mp4<HTAB>
HTTP/1.1<HTAB>200<HTAB>-<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>US/TN/MEM/38138<HTAB>
GET<HTAB>
http://cdni-ucdn.dcdn-1.example.com/video/picture11.mp4<HTAB>
HTTP/1.0<HTAB>200<HTAB>-<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>

#SHA256-hash:<HTAB> 64-hexadecimal-digit hash value <CRLF>
```

Figure 5: CDNI Logging File Example With A Missing Field Value

3.7. Cascaded CDNI Logging Files Example

Let us consider the cascaded CDN scenario of uCDN, dCDN-2 and dCDN-3 as depicted in Figure 1. After completion of a delivery by dCDN-3 on behalf of dCDN-2, dCDN-3 will include a corresponding Logging Record in a CDNI Logging File that will be pulled by dCDN-2 and that is illustrated below in Figure 6. In practice, a CDNI Logging File is

likely to contain a very high number of CDNI Logging Records. However, for readability, the example in Figure 6 contains a single CDNI Logging Record.

```
#version:<HTAB>cdni/1.0<CRLF>

#UUID:<HTAB>urn:uuid:65718ef-0123-9876-adce4321bcde<CRLF>

#claimed-origin:<HTAB>cdni-logging-entity.dcdn-3.example.com<CRLF>

#record-type:<HTAB>cdni_http_request_v1<CRLF>

#fields:<HTAB>date<HTAB>time<HTAB>time-taken<HTAB>c-groupid<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>00:39:09.119<HTAB>14.07<HTAB>US/CA/SFO/94114<HTAB>
GET<HTAB>
http://cdni-dcdn-2.dcdn-3.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>

#SHA256-hash:<HTAB> 64-hexadecimal-digit hash value <CRLF>
```

Figure 6: Cascaded CDNI Logging File Example (dCDN-3 to dCDN-2)

If dCDN-2 establishes by some means (e.g., via TLS authentication when pulling the CDNI Logging File) the identity of the entity from which it pulled the CDNI Logging File, dCDN-2 can add to the CDNI Logging an established-origin directive as illustrated below:

```
#established-origin:<HTAB>cdni-logging-entity.dcdn-
3.example.com<CRLF>
```

dCDN-2 (behaving as an upstream CDN from the viewpoint of dCDN-3) will then ingest the CDNI Logging Record for the considered dCDN-3 delivery into its Collection process (as illustrated in Figure 2). This Logging Record may be aggregated with Logging Records generated locally by dCDN-2 for deliveries performed by dCDN-2 itself. Say, for illustration, that the content delivery performed by dCDN-3 on behalf of dCDN-2 had actually been redirected to dCDN-2 by uCDN, and say that another content delivery has just been redirected by uCDN to dCDN-2 and that dCDN-2 elected to perform the corresponding delivery itself. Then after Filtering and Rectification (as illustrated in Figure 2), dCDN-2 will include the two Logging Records corresponding

respectively to the delivery performed by dCDN-3 and the delivery performed by dCDN-2, in the next CDNI Logging File that will be communicated to uCDN. An example of such CDNI Logging File is illustrated below in Figure 7.

```
#version:<HTAB>cdni/1.0<CRLF>

#UUID:<HTAB>urn:uuid:1234567-8fedc-abab-0987654321ff<CRLF>

#claimed-origin:<HTAB>cdni-logging-entity.dcdn-2.example.com<CRLF>

#record-type:<HTAB>cdni_http_request_v1<CRLF>

#fields:<HTAB>date<HTAB>time<HTAB>time-taken<HTAB>c-groupid<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>00:39:09.119<HTAB>14.07<HTAB>US/CA/SFO/94114<HTAB>
GET<HTAB>
http://cdni-ucdn.dcdn-2.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>01:42:53.437<HTAB>52.879<HTAB>FR/IDF/PAR/75001<HTAB>
GET<HTAB>
http://cdni-ucdn.dcdn-2.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>

#SHA256-hash:<HTAB> 64-hexadecimal-digit hash value <CRLF>
```

Figure 7: Cascaded CDNI Logging File Example (dCDN-2 to uCDN)

If uCDN establishes by some means (e.g., via TLS authentication when pulling the CDNI Logging File) the identity of the entity from which it pulled the CDNI Logging File, uCDN can add to the CDNI Logging an established-origin directive as illustrated below:

```
#established-origin:<HTAB>cdni-logging-entity.dcdn-
2.example.com<CRLF>
```

In the example of Figure 7, we observe that:

- the first Logging Record corresponds to the Logging Record communicated earlier to dCDN-2 by dCDN-3, which corresponds to a delivery redirected by uCDN to dCDN-2 and then redirected by dCDN-2 to dCDN-3. The fields values in this Logging Record are copied from the corresponding CDNI Logging REcord communicated to dCDN2 by dCDN-3, with the exception of the u-uri that now reflects the URI convention between uCDN and dCDN-2 and that presents the delivery to uCDN as if it was performed by dCDN-2 itself. This reflects the fact that dCDN-2 had taken the full responsibility of the corresponding delivery (even if in this case, dCDN-2 elected to redirect the delivery to dCDN-3 so it is actually performed by dCDN-3 on behalf of dCDN-2).

- the second Logging Record corresponds to a delivery redirected by uCDN to dCDN-2 and performed by dCDN-2 itself. The time of the delivery in this Logging Record may be significantly more recent than the first Logging Record since it was generated locally while the first Logging Record was generated by dCDN-3 and had to be advertised , and then pulled and then ingested into the dCDN-2 Collection process, before being aggregated with the second Logging Record.

4. Protocol for Exchange of CDNI Logging File After Full Collection

This section specifies a protocol for the exchange of CDNI Logging Files as specified in Section 3 after the CDNI Logging File is fully collected by the dCDN.

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 Files. 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’s convenience. The CDNI Logging File pull mechanisms is specified in Section 4.2.

An implementation of the CDNI Logging interface on the dCDN side (the entity generating the CDNI Logging file) MUST support the server side of the CDNI Logging feed (as specified in Section 4.1) and the server side of the CDNI Logging pull mechanism (as specified in Section 4.2).

An implementation of the CDNI Logging interface on the uCDN side (the entity consuming the CDNI Logging file) MUST support the client side

of the CDNI Logging feed (as specified in Section 4.1) and the client side of the CDNI Logging pull mechanism (as specified in Section 4.2).

4.1. CDNI Logging Feed

The server-side implementation of the CDNI Logging feed MUST produce 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 is a CDNI logging file. This MIME Media Type is defined as "application/cdni" (See [RFC7736]) with the Payload Type (ptype) parameter set to "logging-file".

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 is a CDNI Logging File using the "application/cdni" MIME Media Type with the Payload Type (ptype) parameter set to "logging-file"(See [RFC7736]).

The URI 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 are outside the scope of the present document and are expected to be agreed upon by uCDN and dCDN via other means (e.g., human agreement).

The server-side implementation MUST be able to set, and SHOULD set, HTTP cache control headers on the subscription feed to indicate the frequency at which the client-side is to poll for updates.

The client-side MAY use HTTP cache control headers (set by the server-side) on the subscription feed to determine the frequency at which to poll for updates. The client-side MAY instead, or in addition, use other information to determine when to poll for updates (e.g., a polling frequency that may have been negotiated between the uCDN and dCDN by mechanisms outside the scope of the present document and that is to override the indications provided in the HTTP cache control headers).

The potential retention limits (e.g., sliding time window) within which the dCDN is to retain and be ready to serve an archive document 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 MUST retain, and be ready to serve, any archive document within the agreed retention limits. Outside these agreed limits, the server-side implementation MAY indicate its inability to serve (e.g., with HTTP status code 404) an archive document or MAY refuse to serve it (e.g., with HTTP status code 403 or 410).

4.1.3. Redundant Feeds

The server-side implementation MAY present more than one CDNI Logging feed for redundancy. Each CDNI Logging File 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 can use the UUID of the CDNI Logging File, presented in the atom:id element of the Atom feed, to avoid unnecessarily pulling and storing a given CDNI Logging File more than once.

4.1.4. Example CDNI Logging Feed

Figure 8 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">
  <title type="text">CDNI Logging Feed</title>
  <updated>2013-03-23T14:46:11Z</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 14:15:00</title>
      <id>urn:uuid:12345678-1234-abcd-00aa-01234567abcd</id>
      <updated>2013-03-23T14:15:00Z</updated>
      <content src="https://dcdn.example/logs/ucdn/
         http-requests-20130323141500000000"
         type="application/cdni"
         ptype="logging-file"/>
      <summary>CDNI Logging File for uCDN at
      2013-03-23 14:15:00</summary>
  </entry>
  <entry>
    <title type="text">CDNI Logging File for uCDN at
      2013-03-23 14:30:00</title>
      <id>urn:uuid:87654321-4321-dcba-aa00-dcba7654321</id>
      <updated>2013-03-23T14:30:00Z</updated>
      <content src="https://dcdn.example/logs/ucdn/
         http-requests-20130323143000000000"
         type="application/cdni"
         ptype="logging-file"/>
      <summary>CDNI Logging File for uCDN at
      2013-03-23 14:30:00</summary>
  </entry>
  ...
  <entry>
    ...
  </entry>
</feed>
```

Figure 8: Example subscription document of a CDNI Logging Feed

4.2. CDNI Logging File Pull

A client-side implementation of the CDNI Logging interface MAY pull, at its convenience, a CDNI Logging File that is published by the server-side in the CDNI Logging Feed (in the subscription document or an archive document). To do so, the client-side:

- MUST implement HTTP/1.1 ([RFC7230],[RFC7231], [RFC7232], [RFC7233], [RFC7234], [RFC7235]), MAY also support other HTTP versions (e.g., HTTP/2 [RFC7540]) and MAY negotiate which HTTP version is actually used. This allows operators and implementers to choose to use later versions of HTTP to take advantage of new features, while still ensuring interoperability with systems that only support HTTP/1.1.

- MUST use the URI that was associated to the CDNI Logging File (within the "src" attribute of the corresponding atom:content element) in the CDNI Logging Feed;

- MUST support exchange of CDNI Logging Files with no content encoding applied to the representation;

- MUST support exchange of CDNI Logging Files with "gzip" content encoding (as defined in [RFC7230]) applied to the representation.

Note that a client-side implementation of the CDNI Logging interface MAY pull a CDNI Logging File that it has already pulled.

The server-side implementation MUST respond to valid pull request by a client-side implementation for a CDNI Logging File published by the server-side in the CDNI Logging Feed (in the subscription document or an archive document). The server-side implementation:

- MUST implement HTTP/1.1 to handle the client-side request and MAY also support other HTTP versions (e.g., HTTP/2);

- MUST include the CDNI Logging File identified by the request URI inside the body of the HTTP response;

- MUST support exchange of CDNI Logging Files with no content encoding applied to the representation;

- MUST support exchange of CDNI Logging Files with "gzip" content encoding (as defined in [RFC7231]) applied to the representation.

Content negotiation approaches defined in [RFC7231] (e.g., using Accept-Encoding request-header field or Content-Encoding entity-header field) MAY be used by the client-side and server-side

implementations to establish the content-coding to be used for a particular exchange of a CDNI Logging File.

Applying compression content encoding (such as "gzip") is expected to mitigate the impact of exchanging the large volumes of logging information expected across CDNs. This is expected to be particularly useful in the presence of HTTP Adaptive Streaming (HAS) which, as per the present version of the document, will result in a separate CDNI Log Record for each HAS segment delivery in the CDNI Logging File.

The potential retention limits (e.g., sliding time window, maximum aggregate file storage quotas) within which the dCDN is to retain and be ready to serve a CDNI Logging File previously advertised in the CDNI Logging Feed 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 MUST retain, and be ready to serve, any CDNI Logging File within the agreed retention limits. Outside these agreed limits, the server-side implementation MAY indicate its inability to serve (e.g., with HTTP status code 404) a CDNI Logging File or MAY refuse to serve it (e.g., with HTTP status code 403 or 410).

5. Protocol for Exchange of CDNI Logging File During Collection

We note that, in addition to the CDNI Logging File exchange protocol specified in Section 4, implementations of the CDNI Logging interface may also support other mechanisms to exchange CDNI Logging Files. In particular, such mechanisms might allow the exchange of the CDNI Logging File to start before the file is fully collected. This can allow CDNI Logging Records to be communicated by the dCDN to the uCDN as they are gathered by the dCDN without having to wait until all the CDNI Logging Records of the same logging period are collected in the corresponding CDNI Logging File. This approach is commonly referred to as "tailing" of the file.

Such an approach could be used, for example, to exchange logging information with a significantly reduced time-lag (e.g., sub-minute or sub-second) between when the event occurred in the dCDN and when the corresponding CDNI Logging Record is made available to the uCDN. This can satisfy log-consuming applications requiring extremely fresh logging information such as near-real-time content delivery monitoring. Such mechanisms are for further study and outside the scope of this document.

6. IANA Considerations

6.1. CDNI Logging Directive Names Registry

The IANA is requested to create a new "CDNI Logging Directive Names" subregistry under the "Content Delivery Networks Interconnection (CDNI) Parameters" registry.

The initial contents of the CDNI Logging 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
established-origin	RFC xxxx
remark	RFC xxxx
record-type	RFC xxxx
fields	RFC xxxx
SHA256-hash	RFC xxxx

Figure 9

[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]. Directive names are to be allocated by IANA with a format of NAMEFORMAT (see Section 3.1). All directive names defined in the logging file are case-insensitive as per the basic ABNF([RFC5234]).

Each specification that defines a new CDNI Logging directive needs to contain a description for the new directive with the same set of information as provided in Section 3.3 (i.e., format, directive value and occurrence).

6.2. CDNI Logging File version Registry

The IANA is requested to create a new "CDNI Logging File version" subregistry under the "Content Delivery Networks Interconnection (CDNI) Parameters" registry.

The initial contents of the CDNI Logging Logging File version registry comprise the value "cdni/1.0" specified in Section 3.3 of the present document, and are as follows:

version	Reference	Description
cdni/1.0	RFC xxxx	CDNI Logging File version 1.0 as specified in RFC xxxx

Figure 10

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

Within the registry, version values are to be allocated by IANA according to the "Specification Required" policy specified in [RFC5226]. Version values are to be allocated by IANA with a format of NAMEFORMAT (see Section 3.1). All version values defined in the logging file are case-insensitive as per the basic ABNF([RFC5234]).

6.3. CDNI Logging record-types Registry

The IANA is requested to create a new "CDNI Logging record-types" subregistry under the "Content Delivery Networks Interconnection (CDNI) Parameters" registry.

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	Description
cdni_http_request_v1	RFC xxxx	CDNI Logging Record version 1 for content delivery using HTTP

Figure 11

[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]. Record-types are to be allocated by IANA with a format of

NAMEFORMAT (see Section 3.1). All record-types defined in the logging file are case-insensitive as per the basic ABNF([RFC5234]).

Each specification that defines a new record-type needs to contain a description for the new record-type with the same set of information as provided in Section 3.4.1. This includes:

- a list of all the CDNI Logging fields that can appear in a CDNI Logging Record of the new record-type
- for all these fields: a specification of the occurrence for each Field in the new record-type
- for every newly defined Field, i.e., for every Field that results in a registration in the CDNI Logging Field Names Registry (Section 6.4): a specification of the field name, format and field value.

6.4. CDNI Logging Field Names Registry

The IANA is requested to create a new "CDNI Logging Field Names" subregistry under the "Content Delivery Networks Interconnection (CDNI) Parameters" registry.

This registry is intended to be shared across the currently defined record-type (i.e., cdni_http_request_v1) as well as potential other CDNI Logging record-types that may be defined in separate specifications. When a Field from this registry is used by another CDNI Logging record-type, it is to be used with the exact semantics and format specified in the document that registered this field and that is identified in the Reference column of the registry. If another CDNI Logging record-type requires a Field with semantics that are not strictly identical, or a format that is not strictly identical then this new Field is to be registered in the registry with a different Field name. When a Field from this registry is used by another CDNI Logging record-type, it can be used with different occurrence rules.

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-groupid	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(insert_HTTP_header_name_here)	RFC xxxx
sc(insert_HTTP_header_name_here)	RFC xxxx
s-ccid	RFC xxxx
s-sid	RFC xxxx
s-cached	RFC xxxx

Figure 12

[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]. Field names are to be allocated by IANA with a format of NHTABSTRING (see Section 3.1). All field names defined in the logging file are case-insensitive as per the basic ABNF([RFC5234]).

6.5. CDNI Logging MIME Media Type

The IANA is requested to register the following new Payload Type in the CDNI Payload Type registry for use with the application/cdni MIME media type.

[RFC Editor Note: Please replace the references to [RFCthis] below with this document’s RFC number before publication.]

Payload Type	Specification
logging-file	[RFCthis]

Figure 13: MIME Media Type payload

The purpose of the logging-file payload type is to distinguish between CDNI Logging Files and other CDNI messages.

Interface: LI

Encoding: see Section 3.2, Section 3.3 and Section 3.4

7. Security Considerations

7.1. Authentication, Authorization, Confidentiality, Integrity Protection

An implementation of the CDNI Logging interface MUST support TLS transport of the CDNI Logging feed (Section 4.1) and of the CDNI Logging File pull (Section 4.2) as per [RFC2818] and [RFC7230].

TLS MUST be used by the server-side and the client-side of the CDNI Logging feed, as well as the server-side and the client-side of the CDNI Logging File pull mechanism, including authentication of the remote end, unless alternate methods are used for ensuring the security of the information exchanged over the LI interface (such as setting up an IPsec tunnel between the two CDNs or using a physically secured internal network between two CDNs that are owned by the same corporate entity).

The use of TLS for transport of the CDNI Logging feed and CDNI Logging File pull allows:

- the dCDN and uCDN to authenticate each other using TLS client auth and TLS server auth.

and, once they have mutually authenticated each other, it allows:

- the dCDN and uCDN to authorize each other (to ensure they are transmitting/receiving CDNI Logging File to/from an authorized CDN)

- the CDNI Logging information to be transmitted with confidentiality

- o the integrity of the CDNI Logging information to be protected during the exchange.

When TLS is used, the general TLS usage guidance in [RFC7525] MUST be followed.

The SHA256-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, storage or exchange. This mechanism does not itself allow restoration of the corrupted CDNI Logging information, but it allows detection of such corruption and therefore triggering of appropriate corrective actions (e.g., discard of corrupted information, attempt to re-obtain the CDNI Logging information). Note that the SHA256-hash does not protect against tampering by a third party, since such a third party could have recomputed and updated the SHA256-hash after tampering. Protection against third party tampering, when the CDNI Logging File is communicated over the CDN Logging Interface, can be achieved as discussed above through the use of TLS.

7.2. Denial of Service

This document does not define specific mechanism to protect against Denial of Service (DoS) attacks on the Logging Interface. However, the CDNI Logging feed and CDNI Logging pull endpoints are typically to be accessed only by a very small number of valid remote endpoints and therefore can be easily protected against DoS attacks through the usual conventional DOS protection mechanisms such as firewalling or use of Virtual Private Networks (VPNs).

Protection of dCDN Surrogates against spoofed delivery requests is outside the scope of the CDNI Logging interface.

7.3. Privacy

CDNs have the opportunity to collect detailed information about the downloads performed by End Users. A dCDN is expected to collect such information into CDNI Logging Files, which are then communicated to an uCDN.

Having detailed CDNI logging information known by the dCDN in itself does not represent a particular privacy concern since the dCDN is obviously fully aware of all information logged since it generated the information in the first place.

Transporting detailed CDNI logging information over the HTTP based CDNI Logging Interface does not represent a particular privacy

concern because it is protected by usual IETF privacy-protection mechanism (e.g.,TLS).

When HTTP redirection is used between the uCDN and the dCDN, making detailed CDNI logging information known to the uCDN does not represent a particular privacy concern because the uCDN is already exposed at request redirection time to most of the information that shows up as CDNI logging information (e.g., enduser IP@, URL, HTTP headers). When DNS redirection is used between the uCDN and the dCDN, there are cases where there is no privacy concern in making detailed CDNI logging information known to the uCDN; this may be the case, for example, where (1) it is considered that because the uCDN has the authority (with respect to the CSP) and control on how the requests are delivered (including whether it is served by the uCDN itself or by a dCDN), the uCDN is entitled to access all detailed information related to the corresponding deliveries, and (2) there is no legal reasons to restrict access by the uCDN to all these detailed information. Conversely, still when DNS redirection is used between the uCDN and the dCDN, there are cases where there may be some privacy concern in making detailed CDNI logging information known to the uCDN; this may be the case, for example, because the uCDN is in a different jurisdiction to the dCDN resulting is some legal reasons to restrict access by the uCDN to all the detailed information related to the deliveries. In this latter case, the privacy concern can be taken into account when the uCDN and dCDN agree about which fields are to be conveyed inside the CDNI Logging Files and which privacy protection mechanism is to be used as discussed in the definition of the c-groupid field specified in Section 3.4.1.

Another privacy concern arises from the fact that large volumes of detailed information about content delivery to users, potentially traceable back to indvidual users, may be collected in CDNI Logging files. These CDNI Logging files represent high-value targets, likely concentrated in a fairly centralised system (although the CDNI Logging architecture does not mandate a particular level of centralisation/distribution) and at risk of potential data exfiltration. Note that the means of such data exfiltration are beyond the scope of the CDNI Logging interface itself (e.g., corrupted employee, corrupted logging storage system,...). This privacy concern calls for some protection.

The collection of large volumes of such information into CDNI Logging Files introduces potential End Users privacy protection concerns. Mechanisms to address these concerns are discussed in the definition of the c-groupid field specified in Section 3.4.1.

The use of mutually authenticated TLS to establish a secure session for the transport of the CDNI Logging feed and CDNI Logging pull as

discussed in Section 7.1 provides confidentiality while the logging information is in transit and prevents any party other than the authorised uCDN to gain access to the logging information.

We also note that the query string portion of the URL that may be conveyed inside the cs-uri and u-uri fields of CDNI Logging Files, or the HTTP cookies([RFC6265]) that may be conveyed as part of the cs(<HTTP-header-name>) field of CDNI Logging files, may contain personnal information or information that can be exploited to derive personal information. Where this is a concern, the CDNI Logging interface specification allows the dCDN to not include the cs-uri and to include a u-uri that removes (or hides) the sensitive part of the query string and allows the dCDN to not include the cs(<HTTP-header-name>) fields corresponding to HTTP headers associated with cookies.

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 thank Ben Niven-Jenkins, Kevin Ma, David Mandelberg and Ray van Brandenburg for their ongoing input.

Brian Trammel and Rich Salz made significant contributions into making this interface privacy-friendly.

Finally, we also thank Sebastien Cubaud, Pawel Grochocki, Christian Jacquenet, Yannick Le Louedec, Anne Marrec, Emile Stephan, Fabio Costa, Sara Oueslati, Yvan Massot, Renaud Edel, Joel Favier and the contributors of the EU FP7 OCEAN project for their input in the early versions of this document.

9. References

9.1. Normative References

[AES] NIST, "Advanced Encryption Standard (AES)", August 2015, <FIPS 197>.

[GCM] NIST, "Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC", November 2007, <SP 800-38D>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, DOI 10.17487/RFC3339, July 2002, <http://www.rfc-editor.org/info/rfc3339>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <http://www.rfc-editor.org/info/rfc3986>.

[RFC4122] Leach, P., Mealling, M., and R. Salz, "A Universally Unique IDentifier (UUID) URN Namespace", RFC 4122, DOI 10.17487/RFC4122, July 2005, <http://www.rfc-editor.org/info/rfc4122>.

[RFC4287] Nottingham, M., Ed. and R. Sayre, Ed., "The Atom Syndication Format", RFC 4287, DOI 10.17487/RFC4287, December 2005, <http://www.rfc-editor.org/info/rfc4287>.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <http://www.rfc-editor.org/info/rfc4648>.

[RFC5005] Nottingham, M., "Feed Paging and Archiving", RFC 5005, DOI 10.17487/RFC5005, September 2007, <http://www.rfc-editor.org/info/rfc5005>.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, DOI 10.17487/RFC5226, May 2008, <http://www.rfc-editor.org/info/rfc5226>.

[RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <http://www.rfc-editor.org/info/rfc5234>.

[RFC7230] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014, <http://www.rfc-editor.org/info/rfc7230>.

[RFC7231] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, DOI 10.17487/RFC7231, June 2014, <http://www.rfc-editor.org/info/rfc7231>.

[RFC7232] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Conditional Requests", RFC 7232, DOI 10.17487/RFC7232, June 2014, <http://www.rfc-editor.org/info/rfc7232>.

[RFC7233] Fielding, R., Ed., Lafon, Y., Ed., and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Range Requests", RFC 7233, DOI 10.17487/RFC7233, June 2014, <http://www.rfc-editor.org/info/rfc7233>.

[RFC7234] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Caching", RFC 7234, DOI 10.17487/RFC7234, June 2014, <http://www.rfc-editor.org/info/rfc7234>.

[RFC7235] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Authentication", RFC 7235, DOI 10.17487/RFC7235, June 2014, <http://www.rfc-editor.org/info/rfc7235>.

[RFC7525] Sheffer, Y., Holz, R., and P. Saint-Andre, "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", BCP 195, RFC 7525, DOI 10.17487/RFC7525, May 2015, <http://www.rfc-editor.org/info/rfc7525>.

[RFC7540] Belshe, M., Peon, R., and M. Thomson, Ed., "Hypertext Transfer Protocol Version 2 (HTTP/2)", RFC 7540, DOI 10.17487/RFC7540, May 2015, <http://www.rfc-editor.org/info/rfc7540>.

[SHA-3] NIST, "SHA-3 STANDARD: PERMUTATION-BASED HASH AND EXTENDABLE OUTPUT FUNCTIONS", November 2001, <http://dx.doi.org/10.6028/NIST.FIPS.202>.

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.ietf-cdni-metadata]
Niven-Jenkins, B., Murray, R., Caulfield, M., and K. Ma, "CDN Interconnection Metadata", draft-ietf-cdni-metadata-17 (work in progress), May 2016.

[I-D.ietf-tls-rfc5246-bis]
Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.3", draft-ietf-tls-rfc5246-bis-00 (work in progress), April 2014.

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

[RFC1945] Berners-Lee, T., Fielding, R., and H. Frystyk, "Hypertext Transfer Protocol -- HTTP/1.0", RFC 1945, DOI 10.17487/RFC1945, May 1996, <http://www.rfc-editor.org/info/rfc1945>.

[RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, DOI 10.17487/RFC2818, May 2000, <http://www.rfc-editor.org/info/rfc2818>.

[RFC5869] Krawczyk, H. and P. Eronen, "HMAC-based Extract-and-Expand Key Derivation Function (HKDF)", RFC 5869, DOI 10.17487/RFC5869, May 2010, <http://www.rfc-editor.org/info/rfc5869>.

[RFC6234] Eastlake 3rd, D. and T. Hansen, "US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)", RFC 6234, DOI 10.17487/RFC6234, May 2011, <http://www.rfc-editor.org/info/rfc6234>.

[RFC6265] Barth, A., "HTTP State Management Mechanism", RFC 6265, DOI 10.17487/RFC6265, April 2011, <http://www.rfc-editor.org/info/rfc6265>.

[RFC6707] Niven-Jenkins, B., Le Faucheur, F., and N. Bitar, "Content Distribution Network Interconnection (CDNI) Problem Statement", RFC 6707, DOI 10.17487/RFC6707, September 2012, <http://www.rfc-editor.org/info/rfc6707>.

[RFC6770] Bertrand, G., Ed., Stephan, E., Burbridge, T., Eardley, P., Ma, K., and G. Watson, "Use Cases for Content Delivery Network Interconnection", RFC 6770, DOI 10.17487/RFC6770, November 2012, <http://www.rfc-editor.org/info/rfc6770>.

[RFC6983] van Brandenburg, R., van Deventer, O., Le Faucheur, F., and K. Leung, "Models for HTTP-Adaptive-Streaming-Aware Content Distribution Network Interconnection (CDNI)", RFC 6983, DOI 10.17487/RFC6983, July 2013, <http://www.rfc-editor.org/info/rfc6983>.

[RFC7336] Peterson, L., Davie, B., and R. van Brandenburg, Ed., "Framework for Content Distribution Network Interconnection (CDNI)", RFC 7336, DOI 10.17487/RFC7336, August 2014, <http://www.rfc-editor.org/info/rfc7336>.

[RFC7337] Leung, K., Ed. and Y. Lee, Ed., "Content Distribution Network Interconnection (CDNI) Requirements", RFC 7337, DOI 10.17487/RFC7337, August 2014, <http://www.rfc-editor.org/info/rfc7337>.

[RFC7736] Ma, K., "Content Delivery Network Interconnection (CDNI) Media Type Registration", RFC 7736, DOI 10.17487/RFC7736, December 2015, <http://www.rfc-editor.org/info/rfc7736>.

Authors' Addresses

Francois Le Faucheur (editor)
FR

Phone: +33 6 19 98 50 90
Email: flefauch@gmail.com

Gilles Bertrand (editor)

Phone: +41 76 675 91 44
Email: gilbertrand@gmail.com

Iuniana Oprescu (editor)
FR

Email: iuniana.oprescu@gmail.com

Roy Peterkofsky
Google Inc.
345 Spear St, 4th Floor
San Francisco CA 94105
USA

Email: peterkofsky@google.com