Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: 30 December 2025

PB. Pierre, Ed.
Merklemap

28 June 2025

# Certificate Transparency Pages Extension
## draft-trans-pages-01

## Abstract

This document specifies an extension to RFC 6962 Certificate Transparency (CT) logs that enables efficient caching and batch retrieval through page-based access patterns. The extension introduces a binary format that eliminates base64 encoding overhead and certificate chain duplication while maintaining full backward compatibility with existing RFC 6962 implementations.

## 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 https://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 30 December 2025.

## Copyright Notice

Table of Contents



## 1. Introduction

Certificate Transparency (CT) [RFC6962] provides a framework for publicly logging the existence of Transport Layer Security (TLS) certificates as they are issued or observed. The current specification defines a "get-entries" endpoint that accepts arbitrary start and end parameters for retrieving log entries.

### 1.1. Motivation

The current RFC 6962 design presents several challenges:

* Arbitrary range requests make caching difficult or impossible, as responses for overlapping ranges cannot be efficiently cached or reused.

* Base64 encoding of certificates adds approximately 33% overhead to response sizes.

* Certificate chains are duplicated in full for each entry, even when many entries share the same intermediate certificates.

* Variable response sizes complicate client implementation and server resource planning.

This extension addresses these issues by introducing fixed-size pages with an efficient binary format, while maintaining full backward compatibility with existing CT infrastructure.

### 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 BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

## 2. Page-Based Entry Retrieval

This extension introduces a page-based mechanism for retrieving log entries in fixed-size batches.

### 2.1. Request Format

Clients request pages using the following HTTP GET request:

GET /ct-pages/v1/page/{page_number}

Where:

* page_number: A non-negative integer representing the zero-indexed page number.

Pages accessed by number ALWAYS contain exactly page_size entries. Partial pages (those with fewer than page_size entries) are not accessible via numbered endpoints until they become complete.

### 2.2. Response Format

#### 2.2.1. HTTP Headers

Successful responses include the following HTTP headers:

Content-Type: application/octet-stream
Cache-Control: public, max-age=31536000, immutable

For a partially filled page:

Cache-Control: no-store

Where:

* Cache-Control: Complete pages (those containing exactly page_size entries) are immutable and can be cached indefinitely. Partial pages (those containing fewer than page_size entries) must not be cached as they may receive additional entries.

#### 2.2.2. Binary Response Structure

The response body uses the following binary format, expressed using the TLS presentation language from [RFC8446]:

```
enum { v1(0), (255) } Version;

struct {
    Version format_version;
    uint64 entry_count;
    uint64 first_entry_index;
    PageEntry entries[entry_count];
} EntriesPage;

struct {
    TimestampedEntry timestamped_entry;
    uint16 chain_length;
    opaque issuer_hashes[chain_length][32];
} PageEntry;
```

Where:

* format_version: Set to v1(0) for this specification.

* entry_count: The number of entries in this page.

* first_entry_index: The log index of the first entry in this page.

* timestamped_entry: The TimestampedEntry structure as defined in Section 3.4 of [RFC6962].

* chain_length: The number of certificates in the chain (excluding the leaf certificate, which is included in timestamped_entry).

* issuer_hashes: SHA-256 hashes of the DER-encoded issuer certificates in the chain, ordered from the leaf’s issuer to the root.

### 2.3. Certificate Resolution

To retrieve the actual certificate data for entries in the chain, clients make separate requests:

GET /ct-pages/v1/certificate/{base64url_sha256_hash}

Where base64url_sha256_hash is the base64url encoding (without padding) of the SHA-256 hash of the certificate.

Successful responses return:

Content-Type: application/pkix-cert
Cache-Control: public, max-age=31536000, immutable

[binary certificate data]

This mechanism allows efficient deduplication of commonly used intermediate certificates across many log entries.

### 2.4. Latest Page Retrieval

To retrieve the highest-numbered page containing the most recent entries, clients make the following request:

GET /ct-pages/v1/latest

The response uses the same binary format as regular page requests (see Section 2.2.2), with the following HTTP headers:

Content-Type: application/octet-stream
Cache-Control: no-store

This endpoint returns the current page being filled with entries, which may be partial (containing fewer than page_size entries) or complete (containing exactly page_size entries). This page does not have a page number until it becomes complete.

Note: There may be temporary duplication between the /ct-pages/v1/latest endpoint and the highest numbered page endpoint in two scenarios:

* When a page has just been completed (exactly page_size entries)

* During the transition when a completed page is being assigned a number

Clients MUST handle potential duplication by using the first_entry_index and entry_count fields to identify unique pages. Clients MUST NOT cache responses from this endpoint as the content may change as new entries are added to the log.

### 2.5. Discovery Mechanism

Logs implementing this extension MUST provide a discovery endpoint:

GET /ct-pages/v1/discover

The response is a JSON object:

Content-Type: application/json

```
{
  "page_size": 1000,
  "static_endpoint": "https://static.example.com",
  "last_page_at_static": false
}
```

Where:

* page_size: The fixed number of entries per complete page.

* static_endpoint: (OPTIONAL) An alternative base URL for fetching pages and certificates. If provided, clients MUST use this endpoint for all complete page and certificate requests. The same path structure is used as with the main log server (e.g., if static_endpoint is "https://static.example.com", then page 0 would be fetched from "https://static.example.com/ct-pages/v1/page/0"). This field SHOULD be omitted if the static content is served from the same host as the log endpoints.

* last_page_at_static: (OPTIONAL) A boolean indicating whether the latest page should be fetched from the static_endpoint (true) or from the main log server (false). Defaults to false if not specified. This field MUST NOT be present if static_endpoint is not provided.

## 3. Backward Compatibility

This extension maintains full backward compatibility with RFC 6962:

* All original RFC 6962 endpoints remain unchanged and continue to function.

* The extension introduces new endpoints under the "/ct-pages/v1/" path prefix.

* Clients unaware of this extension continue to work with the original endpoints.

* Logs can implement this extension without breaking existing clients.

Unless a separate static_endpoint is specified in the discovery response, the pages endpoints MUST be served on the same host as the main log endpoints.

# 4. Operational Considerations

## 4.1. Page Size Stability

Once a log begins serving pages with a particular page size, it MUST NOT change this size. Changing the page size would:

* Invalidate all cached responses

* Break client assumptions about page boundaries

* Complicate client implementation significantly

Logs SHOULD choose a page size that balances response size with the number of requests needed to retrieve large ranges of entries. A page size of 1000 entries is RECOMMENDED as a reasonable default.

## 4.2. Static Deployment

Since pages become immutable once they contain page_size entries, logs can:

* Pre-generate page files for all complete pages

* Serve pages from static file hosting or CDN infrastructure

* Significantly reduce computational load on log servers

* Improve response times and reliability

The last_page_at_static flag in the discovery response provides deployment flexibility:

* When set to false (default), the latest page is always fetched from the main log server, ensuring real-time accuracy while complete pages are served from static infrastructure.

* When set to true, even the latest page is served from the static endpoint. This requires the static infrastructure to be updated frequently but allows for complete offloading of read traffic.

Clients retrieving the latest page using the /ct-pages/v1/latest endpoint MUST respect the last_page_at_static setting when a static_endpoint is configured. When last_page_at_static is true, the request would be sent to "{static_endpoint}/ct-pages/v1/latest".

## 5. Security Considerations

This extension does not alter the security properties of Certificate Transparency as defined in [RFC6962]. The cryptographic proofs and append-only properties of the log remain unchanged.

Clients MUST verify that certificate hashes match the actual certificates retrieved. This ensures that a compromised CDN or static hosting provider cannot substitute different certificates.

The use of SHA-256 for certificate identification is consistent with its use in RFC 6962 for Merkle tree operations.

## 6. IANA Considerations

This document has no IANA actions.

## 7. References

### 7.1. Normative References

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

[RFC6962] Laurie, B., Langley, A., and E. Kasper, "Certificate Transparency", RFC 6962, DOI 10.17487/RFC6962, June 2013, <https://www.rfc-editor.org/info/rfc6962>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

### 7.2. Informative References

[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, <https://www.rfc-editor.org/info/rfc7231>.

## Author’s Address

Pierre Barre (editor)
Merklemap
320 rue saint honor辿
75001 Paris
France
Email: pierre@barre.sh