openpgp
Internet-Draft
Intended status: Standards Track
Expires: 9 February 2026

D. Shaw
Jabberwocky Tech
A. Gallagher, Ed.
PGPKeys.EU

8 August 2025

OpenPGP HTTP Keyserver Protocol
draft-gallagher-openpgp-hkp-08

Abstract

This document specifies a series of conventions to implement an OpenPGP keyserver using the Hypertext Transfer Protocol (HTTP). As this document is a codification and extension of a protocol that is already in wide use, strict attention is paid to backward compatibility with these existing implementations.

About This Document

This note is to be removed before publishing as an RFC.

The latest revision of this draft can be found at https://andrewgdotcom.gitlab.io/draft-gallagher-openpgp-hkp. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-gallagher-openpgp-hkp/.

Discussion of this document takes place on the OpenPGP Working Group mailing list (mailto:openpgp@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/openpgp/. Subscribe at https://www.ietf.org/mailman/listinfo/openpgp/.

Source for this draft and an issue tracker can be found at https://gitlab.com/andrewgdotcom/draft-gallagher-openpgp-hkp.

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/.

Copyright Notice

Table of Contents

1. Introduction

For ease of use, public key cryptography requires a key distribution system. For many years, the most commonly used system has been a keyserver - a server that stores public keys and/or certificates, with a searchable interface. The HTTP Keyserver Protocol is a OpenPGP keyserver implemented using HTTP.

2. Conventions and Definitions

The term "OpenPGP Certificate" is used in this document interchangeably with "OpenPGP Transferable Public Key", as defined in Section 10.1 of [RFC9580].

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.

3. Keyserver Use Cases

A keyserver is typically used for the following (non-exhaustive) use cases:

3.1. Certificate Discovery

When initiating secure communication with a new correspondent, a client will typically attempt to discover the encryption key(s) that it should use. This is a subtle issue with many security considerations, however many discovery methods involve looking up a certificate on a server using a human-readable identifier such as an email address.

3.2. Certificate Refresh

Certificates in OpenPGP are dynamic objects, therefore it is important to refresh known certificates in order to pick up the latest changes. These changes can include new subkeys and User IDs, updated self-signatures and third-party certifications, and revocations. In some cases it may no longer be possible to search by User ID, therefore it is RECOMMENDED that clients refresh known certificates by fingerprint search.

3.3. Reference Resolution

The OpenPGP wire format includes fields that reference primary keys or subkeys by either Key ID or fingerprint. A client may therefore wish to search for previously unknown certificates based on such a reference.

4. HKP and HTTP

As HKP is implemented over HTTP, everything in [RFC1945] applies to HKP as well, and HKP error codes are the same as the ones used in HTTP.

Due the very large deployment of HKP clients based on HTTP version 1.0, HKP keyservers MUST support HTTP 1.0. HKP keyservers MAY additionally support other HTTP versions.

((dshaw : I expect this to be controversial, but we’ve got tons of deployed code that only works with 1.0. I’d be willing to discuss removing this MUST or make it a SHOULD and add a "implementation notes" section pointing out the problem instead. See issue #5.))

When used over HTTPS, HKP is commonly referred to as "HKPS".

HKP(S) are distinguished from generic use of HTTP(S) by using the URI schemes "hkp" and "hkps" [RFC7595]. HKP is assigned port number 11371 and HKPS is assigned 11372 (although this is rarely used in practice). For reasons of maximum compatibility with firewalls and filtering HTTP proxies, HKP(S) are often served over the standard HTTP(S) port(s) (TCP ports 80 and 443).

By convention and history, HKP defaults to HTTP on TCP port 11371, and HKPS defaults to HTTPS on TCP port 443.

((andrewg : if we assign hkps, we appear to be required to specify a dedicated port, even though nobody uses it. See issue #14.))

A keyserver SHOULD support both HKP and HKPS. A client SHOULD use HKPS, or a transport method with equivalent security properties, such as Tor hidden services [TOR].

4.1. Request Paths

HKP defines three paths, namely "/pks/lookup" for legacy lookups (see Section 5), "/pks/add" for legacy submission (see Section 6), and "/pks/v2" for lookups and submission in the v2 API. Paths beginning with "/pks/v<?>" are reserved for future versions of HKP.

A keyserver MAY support requests to other paths under "/pks", but these are outside the scope of this document. These alternative paths have historically been used to provide human-readable interfaces such as HTML forms, and functionality extensions such as [SKS].

4.2. HTTP Status Codes

When a status or error code needs to be returned by a keyserver, the most appropriate HTTP code from [RFC9110] should be used. It is good practice to return the most specific error code possible: for example, returning 404 ("Not Found") rather than 400 ("Bad Request") when a certificate is not found.

This document gives suggested HTTP error codes for several common situations. Note that these are only suggestions, and implementations may have good reasons (such as not revealing the reason why a request failed) for using other error codes.

Clients SHOULD understand the following codes:

Status Code	Description
200 OK	Request succeeded
202 Accepted	Submitted certificate was altered to match keyserver policy
403 Forbidden	The requested operation is not permitted
404 Not found	The search returned no results, or path not found
410 Gone	Certificate has been permanently deleted, e.g. due to RTBF
413 Content too large	The search returned too many responses
422 Unprocessable content	Submitted certificate was rejected as per keyserver policy
501 Not implemented	The requested operation is not supported

Table 1: Status Codes

In addition, a client SHOULD understand 3xx redirect codes.

((andrewg : In draft-shaw-00 it was suggested that a novel header be used for statuses that could not be represented by the HTTP response codes of the time. This was only partially specified, and it is unclear if any implementations of this header existed. In the meantime many new HTTP response codes have been defined, so I am using them instead - even if their semantics does not exactly match that of [RFC9110]. NB therefore that codes 202, 410, 413, 422 may not have been implemented anywhere yet.))

((andrewg : note also that 413 and 202 may be incorrect usage, see issues #25 and #26 respectively.))

5. Looking up Data from a Keyserver

Certificate lookups are done via an HTTP GET request. Specifically, the abs_path (Section 3.2 of [RFC1945]) is built up of the base path "/pks/lookup" or "/pks/v2", followed by request-specific URL components. These components differ slightly between the Legacy and v2 lookup formats (see below).

Most HKP lookups contain both the "op" (operation) and "search" fields. The "op" field determines what operation the keyserver will execute, and the "search" field determines which certificates are operated on.

There may also be modifier variables, as specified in Section 7 below. Variables are passed using HTTP query strings as specified in Section 8.2.2 of [RFC1866]. HTTP query strings MAY be given in any order. Keyservers MUST ignore any unknown query strings.

5.1. Legacy and v2 Lookup Formats

For backwards compatibility with the existing installed client base, a Legacy lookup format is defined. New implementations SHOULD use the v2 lookup format.

5.1.1. Legacy Lookup Format

In the Legacy lookup format, the "op" and "search" fields are supplied as HTTP query strings, in the form "<field-name>=<value>":

/pks/lookup?op=<op>&search=<search>[&...]

They are treated in the same way as modifier variables, including arbitrary ordering, however the "op" field MUST be supplied, and the "search" field MUST be supplied unless the "stats" operation is being requested (see Section 5.2.6). No URL path components under "/pks/lookup" are used.

5.1.2. v2 Lookup Format

In the v2 lookup format, the values of the "op" and "search" fields are supplied as URL path components. They are appended to "/pks/v2" as follows:

/pks/v2/<op>/<search>[?...]

The "op" and "search" fields MUST be supplied. Modifier variables are supplied as HTTP query strings, in the form "<field-name>=<value>".

If the v2 lookup format is being used, v2 output format (Section 8) MUST be returned.

The v2 lookup format is designed so that a basic HKP service can be implemented using static files.

5.2. The Operation Lookup Field

The operation ("op") field specifies the lookup operation to be performed on the keyserver. The "op" field is generally accompanied by a "search" field to specify the certificates that should be looked up.

If a particular operation is not supported, the keyserver SHOULD return an appropriate HTTP error code such as 501 ("Not Implemented"). The server SHOULD NOT return an error code (such as 404 ("Not Found")) that could be interpreted by the client as an explicit statement of non-existence.

5.2.1. The "get" Operation

A keyserver MAY support the "get" operation.

The "get" operation requests certificates from the keyserver by textual search. A string that specifies which certificate(s) to return is provided in the "search" field.

The response to a successful "get" operation is a HTTP document containing a certificate bundle as specified in Section 10.

A keyserver SHOULD limit the returned certificate bundle to a reasonable length. Results from a User ID search SHOULD be sorted in order of decreasing confidence in that User ID (Section 9), and then by creation date (most recent first). A keyserver MAY choose to only return results where the User ID being searched for has a nonzero confidence value. If no certificates match the request, the keyserver SHOULD return an appropriate HTTP error code such as 404 ("Not Found").

5.2.2. The "authget" (authoritative get) Operation

A keyserver SHOULD support the "authget" operation.

The "authget" operation is similar to the "get" operation, but is used for authoritative lookups by User ID only, in particular during certificate discovery. The keyserver SHOULD only return results where the User ID being searched for has a nonzero confidence value (Section 9). If a keyserver is being used for certificate discovery,

it MUST return only results for which it is has complete confidence. In addition, exact textual matching MUST be used even if the "exact" variable is set to "off" (Section 7.5).

5.2.3. The "prefixlog" Operation

A keyserver MAY support the "prefixlog" operation.

"prefixlog" requests a list of fingerprint prefixes that indicate which certificates have been modified since 00:00:00 UTC on a specific date. The date is provided in the "search" field in ISO format with no time component, i.e. "2025-12-01".

The returned data is a list of CRLF-separated, hexadecimal-encoded fingerprint prefixes, and each prefix is truncated at a hex-digit boundary. The keyserver SHOULD calibrate the prefix length so that it is long enough to provide collision resistance, but short enough to maintain a useful anonymity cohort. A client MUST NOT make any assumptions about the length of the prefixes returned.

A client that wishes to update its local keystore from a keyserver MAY first make a "prefixlog" request with the date of the last successful refresh. It can then compare the returned list of prefixes to see if any of them are present in its local keystore, and make subsequent "vfpget" requests as appropriate (Section 5.2.7). In this way, it can avoid making unnecessary requests that will return no updates, but will still leak information to the keyserver.

Note that prefixes are always used, regardless of fingerprint version. This contrasts with Key IDs, which are version-dependent.

A keyserver MUST NOT support indexing or downloading certificates by prefix.

5.2.4. The "index" Operation

A keyserver MAY support the "index" operation.

The "index" operation requests a list of certificates on the keyserver that match the text in the "search" field. Historically, the "index" operation returned a human-readable HTML document containing links for each found certificate, but this is not required.

A keyserver SHOULD limit the returned index to a reasonable length. Results from a User ID search SHOULD be sorted in order of decreasing confidence in that User ID (Section 9), and then by creation date (most recent first). If no certificates match the request, the keyserver SHOULD return an appropriate HTTP error code such as 404 ("Not Found").

5.2.5. The "vindex" (verbose index) Operation (Deprecated)

A keyserver MAY support the "vindex" operation for Legacy lookups. The "vindex" operation is deprecated and MUST NOT be used in a v2 lookup.

Historically, a "vindex" response was the same as "index" with the addition of showing the signatures on each certificate, but this is not required. A server that supports "vindex" SHOULD treat it as a synonym for "index".

5.2.6. The "stats" (statistics/status) Operation (Deprecated)

A keyserver MAY support the "stats" operation in Legacy lookup format. The "stats" operation is deprecated, and MUST NOT be used in a v2 lookup. It is RECOMMENDED to use a URL outside the standard HKP paths (such as "/pks/stats") instead.

The output of the "stats" operation is implementation-dependent, but may include diagnostic output, configuration state, or other metadata. The "search" field SHOULD NOT be supplied, and SHOULD be ignored if received.

5.2.7. The "vfpget" (versioned fingerprint get) Operation

A keyserver MAY support the "vfpget" operation.

"vfpget" requests a certificate from a keyserver by specifying its versioned fingerprint. The versioned fingerprint is provided in the "search" field in hexadecimal encoding, without a preceding "0x". The hexadecimal digits are not case sensitive.

A versioned fingerprint consists of one octet of fingerprint version number and N octets of fingerprint. This is the same octet sequence used in the Issuer Fingerprint and Intended Recipient Fingerprint subpackets (Section 5.2.3 of [RFC9580]).

If no certificates match the request, the keyserver SHOULD return an appropriate HTTP error code such as 404 ("Not Found").

5.2.8. The "kidget" (key id get) Operation

A keyserver MAY support the "kidget" operation.

"kidget" requests a certificate from a keyserver by specifying its Key ID (Section 5.5.4 of [RFC9580]). The Key ID is provided in the "search" field as 16 hexadecimal digits, without a preceding "0x". The hexadecimal digits are not case sensitive.

If no certificates match the request, the keyserver SHOULD return an appropriate HTTP error code such as 404 ("Not Found").

"kidget" is only required for locating a signing key that made either a V3 signature, or a V4 signature with an Issuer Key ID subpacket and no Issuer Fingerprint subpacket. Issuer Key ID subpackets are not specified for use in later signature versions (Section 5.2.3.12 of [RFC9580]), and so certificates with versions greater than 4 MUST NOT be returned in response to a "kidget" operation.

5.2.9. The "hget" (hash get) Operation

A keyserver MAY support the "hget" operation.

"hget" requests a certificate from a keyserver by specifying its [SKS] digest. The digest is provided in the "search" field in hexadecimal encoding, without a preceding "0x". The hexadecimal digits are not case sensitive.

If no certificates match the request, the keyserver SHOULD return an appropriate HTTP error code such as 404 ("Not Found").

5.3. The "search" Lookup Field

The "search" field contains arbitrary text encoded as usual for a HTTP URL. This text may represent the Key ID, or fingerprint, or some text from a User ID on the certificate being sought, depending on the operation.

5.3.1. Key ID and Fingerprint Searches

To search for a certificate by its Key ID or fingerprint, a client SHOULD use a v2 lookup and either the "kidget" (Section 5.2.8) or "vfpget" (Section 5.2.7) operation (as appropriate).

If making a Legacy lookup, a client SHOULD use the "get" operation and prefix the "search" string with "0x" to indicate a hexadecimal number. Key ID strings are 16 hexadecimal digits (64 bits). Fingerprint strings are either 32 (version 3), 40 (version 4), or 64 (version 6) hexadecimal digits. The hexadecimal digits are not case sensitive.

A keyserver:

* SHOULD accept fingerprints and MAY accept 64-bit Key IDs in the Legacy lookup "search" field.

* MUST NOT return results for 32-bit "short Key ID" searches, as these do not provide sufficient collision resistance.

* MUST NOT return certificates with versions later than 4 for Key ID searches.

* MUST NOT return version 6 (or later) certificates in the results for Legacy machine-readable requests, but MAY do so for Legacy human-readable requests (see Section 8.3).

V3 certificates are no longer considered secure, but MAY be distributed for historical reference.

5.3.2. Text Searches

To search for a certificate by the text of a User ID, a client SHOULD use the "get", "authget", or "index" operation (as appropriate) and SHOULD NOT prefix the "search" string with "0x".

A keyserver MUST NOT return version 6 (or later) certificates in the results for Legacy machine-readable requests, but MAY do so for Legacy human-readable requests (see also Section 8.3.1). Otherwise, how a keyserver handles textual search is implementation defined. See also the definition of the "exact" variable (Section 7.5) for a method to give additional instructions to the server on how the search is to be executed.

5.4. Lookup Examples

Search for all certificates containing the string "dshaw", using plaintext HTTP:

* Legacy lookup format:

  http://keys.example.com:11371/pks/lookup?search=dshaw&op=index

* v2 lookup format:

  http://keys.example.com:11371/pks/v2/index/dshaw

Get certificate 0xDEADBEEFDECAFBAD (64-bit Key ID), using HTTPS:

* Legacy lookup format:

  https://keys.example.com/pks/lookup?op=get&search=0xDEADBEEFDECAFBAD

* v2 lookup format:

  https://keys.example.com/pks/v2/kidget/DEADBEEFDECAFBAD

6. Submitting Certificates To A Keyserver

A keyserver MAY accept submissions via an HTTP POST request, as specified in Section 8.3 of [RFC1945], and Section 8.2.3 of [RFC1866]. Specifically, the abs_path (Section 3.2 of [RFC1945]) is built up of the base path "/pks/add" or "/pks/v2", followed by request-specific URL components. These components, and the form of the POST body, differ between the Legacy and v2 submission formats (see below).

There may also be modifier variables, as specified in Section 7 below. Modifiers are passed using HTTP query strings as specified in Section 8.2.2 of [RFC1866]. HTTP query strings MAY be given in any order. Keyservers MUST ignore any unknown query strings.

6.1. Legacy and v2 Submission Formats

For backwards compatibility with the existing installed client base, a Legacy submission format is defined. New implementations SHOULD use the v2 submission format. Note that more than one certificate may be submitted in a single transaction.

If a keyserver does not support adding certificates via HTTP, then requests to do so should return an appropriate HTTP error code, such as 403 ("Forbidden") if certificate submission has been disallowed, or 404 ("Not Found") if the server does not support the requested submission format.

6.1.1. Legacy Submission Format

In the Legacy submission format, the abs_path (Section 3.2 of [RFC1945]) is always "/pks/add":

/pks/add[?...]

No URL path components under "/pks/add" are used, and there are no mandatory query strings.

The body of the POST message has a content-type of "application/x-www-form-urlencoded". It contains a "keytext" field whose value is an ASCII-armored certificate bundle as specified in Section 10. The ASCII armored certificate bundle should also be urlencoded as specified in Section 8.2.1 of [RFC1866].

6.1.2. V2 Submission Format

In the v2 submission format, the value of the operation field is supplied as a URL path component, and is appended to "/pks/v2" as follows:

/pks/v2/<op>[?...]

The "op" field MUST be supplied. Unlike lookup requests, there is no "search" field defined.

6.2. The Operation Submission Field

The operation field specifies the submission operation to be performed on the keyserver. If a particular submission operation is not supported, the keyserver SHOULD return an appropriate HTTP error code such as 501 ("Not Implemented").

6.2.1. The "add" Operation

A keyserver MAY support the "add" operation.

The body of the POST message contains a certificate bundle as specified in Section 10. It MUST have a content-type of "application/pgp-keys; encoding=binary" and MUST NOT be ASCII-armored.

((TBC: this aligns with the current thinking of the KOO board, but it is a weak preference. We may support additional submission methods; these would be detected by Accept responses to OPTIONS requests (see issue #30).))

The URL MAY additionally contain a "tokens" query parameter (Section 7.2). The "add" operation MAY have limited effect or fail entirely if a token corresponding to the email address in any of the submitted User IDs is not provided.

(Note that a keyserver MAY accept other forms of User ID verification, and the lack of a "tokens" query parameter will not necessarily result in submission failure.)

6.2.2. The "tokensend" Operation

A keyserver MAY support the "tokensend" operation.

The body of the POST message is a CRLF-separated list of email addresses. It SHOULD have a content-type of "text/plain". A keyserver that supports the "tokensend" operation SHOULD attempt to verify each address by sending a time-limited auth token via email. A keyserver MAY limit the number and frequency of verification requests.

((TBC: this follows a prove-then-submit model, which is the inverse of KOO's current submit-then-prove process. The rationale is that submit-then-prove often silently degrades to "submit-then-forget-to-prove". Failed advance proofs are less likely to be mis-reported as a success. In addition, prove-then-submit more easily generalises to other forms of verification.))

6.3. Submission Examples

((to be completed))

7. Modifier Variables

These variables are used to modify basic requests.

7.1. The "options" Variable

This variable takes one or more option values, separated by commas. These are used to modify the behavior of the keyserver on a per-request basis. Each value indicates a boolean flag, where the presence of the value indicates "true" and the absence "false".

7.1.1. The "mr" (Machine-Readable) Option (Legacy)

The machine-readable option instructs the server that a program (rather than a person) is making a Legacy request, so the output SHOULD be in Legacy machine-readable format. If a v2 request format is being used, this option has no effect. See Section 8.3 for the specific details of Legacy machine-readable output.

An implementation that does not wish to provide a human-readable interface MAY choose to behave as if this option is always present. Implementations SHOULD NOT provide a Legacy interface without supporting machine-readable output.

"mr" is meaningful for Legacy requests only.

7.1.2. The "nm" (No Modification) Option

As keyservers may modify submitted certificates to suit a particular policy, this option is used to inform the keyserver that the submitter would rather have the submission fail completely than have the submitted certificate(s) modified. An example of this would be a keyserver that does not accept User IDs with an email address outside of the local domain. If such a certificate was submitted, the keyserver MAY trim any noncompliant User IDs before accepting the certificate. If this option was set, then such a certificate submission SHOULD fail with an appropriate error code such as 422 (Unprocessable content).

"nm" is meaningful for submissions only.

7.1.3. The "cb" (Canonical Bundle) Option

The canonical bundle option instructs the server that for each proof of User ID (such as a verification token, see Section 6.2.2) contained in the submission:

* The certificates contained in the submission that have the corresponding User ID are regarded by the owner as canonical for that User ID,

* The owner wishes for these certificates to be served in the same order that they appear in the submission, and:

* Any certificates with that User ID that are not contained in the submission are not (or no longer) canonical for that User ID.

When a keyserver verifies the proof of each User ID, it updates its internal confidence value accordingly (see Section 9).

Canonicality only affects the identity and order of certificate(s) returned by User ID lookups, and not to the particular form of the certificate(s) returned. Other methods such as [I-D.dkg-openpgp-1pa3pc] are more appropriate for controlling the form of certificates. In particular, a keyserver MUST allow a valid revocation certificate for any key to be uploaded without identity verification.

"cb" is meaningful for submissions only.

7.2. The "tokens" Variable

The "tokens" variable contains a comma-separated list of one or more tokens obtained via the "tokensend" operation (Section 6.2.2). These tokens SHOULD correspond to one or more User IDs present in the certificate bundle being submitted. A keyserver MAY refuse to serve the certificate(s) when queried by User ID if a valid token was not supplied.

"tokens" is meaningful for submissions only.

7.3. The "fingerprint" Variable (Legacy)

This variable takes one argument: "on" or "off". If present and on, it instructs the server to provide the primary key fingerprint for each certificate in a Legacy "index" or "vindex" operation. This variable has no effect on any other operation. The exact format of the displayed fingerprint, like the "index" and "vindex" operations themselves, is implementation defined in Legacy human-readable output. In Legacy machine-readable indexes, a value of "on" indicates that the "keyid" field SHOULD contain the fingerprint, except for v3 certificates (see Section 8.3.1). An implementation SHOULD treat this variable as being "on" for all Legacy machine-readable indexes. An implementation MAY decide to ignore this variable and/or set the default behaviour to "on" for Legacy human-readable indexes.

"fingerprint" is meaningful for Legacy lookups only.

7.4. The "hash" Variable (Legacy)

This variable takes one argument: "on" or "off". If present and on, it instructs the server to provide the [SKS] digest of each certificate in an "index" or "vindex" operation in the Legacy human-readable output format. This variable has no effect on any other operation, or on Legacy machine-readable output. The exact format of the displayed digest, like the "index" and "vindex" operations themselves, is implementation defined. An implementation MAY decide to ignore this variable and/or set the default behaviour to "on".

"hash" is meaningful for Legacy lookups only.

7.5. The "exact" Variable (Legacy)

This variable takes one argument: "on" or "off". If set to "on", it instructs the server to search for an exact match for the contents of the "search" field. A keyserver implementation SHOULD set the default behaviour to "on" and MAY ignore this variable for Legacy lookups. A keyserver implementation MUST ignore this variable and treat all searches as exact in v2 lookups.

When "exact" is set to "on", a keyserver SHOULD only return results if the "search" field exactly matches one of the following:

* The full text string of a User ID.

* The portion between angle brackets ("<...>") in an email-address style User ID.

In either case, the string matching SHOULD NOT be case sensitive.

"exact" is meaningful for Legacy lookups only.

8. Output Formats

HKP was originally intended for both human and programmatic use. In general, the Legacy human-readable output is implementation specific. The "machine-readable" option is used to tailor the output of Legacy requests for automated use. For interoperability, the Legacy machine-readable output MUST carefully follow the guidelines given here. A client implementation SHOULD NOT attempt to parse Legacy human-readable output.

The v2 API always returns either non-armored certificate bundles or JSON [RFC8259], depending on the request.

8.1. v2 Output Format

Clients making v2 requests:

* MUST silently ignore any primary keys with unknown versions or algorithms.

* MUST silently ignore any unknown JSON fields in "index" and "add" responses.

In response to a v2 request, a keyserver:

* MUST set the HTTP header "Access-Control-Allow-Origin: *", as specified in [CORS].

* MUST use the format specified in Section 8.1.1 when responding to "index" operations.

* MUST use the format specified in Section 8.2 when responding to "add" operations.

* MUST return non-armored (binary) certificate bundles in response to lookup requests.

* MUST set "Content-Type: application/json" for "index" and "add" responses.

* MUST set "Content-Type: application/pgp-keys; encoding=binary" when returning non-armored certificate bundles (see Section 5 of [I-D.gallagher-openpgp-media-types])

8.1.1. v2 Indexes

A v2 "index" operation SHOULD return a JSON list of certificates. If the search was for a User ID, it SHOULD be sorted in decreasing order of confidence (Section 9). Each certificate object contains some or all of the following fields:

Field	Type	Description
version	integer	version of the primary key (REQUIRED)
fingerprint	string	fingerprint of the primary key (REQUIRED)
creation	string	creation date of the key
expiration	string	expiration date of the key
isExpired	boolean	
isRevoked	boolean	
algorithm	algorithm	(Table 3)
userIDs	userID array	(Table 4)
subkeys	subkey array	(Table 5)

Table 2: v2 Index Fields

Field	Type	Description
code	integer	algorithm ID (REQUIRED)
name	string	a human-readable identifier for the algorithm
bitLength	integer	key length in bits (DSA/RSA/ElGamal keys only)

Table 3: v2 Index Algorithm Fields

Field	Type	Description
uidString	string	User ID string contents (REQUIRED)
creation	string	creation date of (the first signature over) the User ID
expiration	string	expiration date of the User ID
isExpired	boolean	
isRevoked	boolean	
confidence	integer	(Section 9)

Table 4: v2 Index UserID Fields

Field	Type	Description
version	integer	version of the subkey (REQUIRED)
fingerprint	string	fingerprint of the subkey (REQUIRED)
creation	string	creation date of the subkey
expiration	string	expiration date of the subkey
isExpired	boolean	
isRevoked	boolean	
algorithm	algorithm	(Table 3)

Table 5: v2 Index Subkey Fields

Fingerprints and long Key IDs are given in hexadecimal notation, without any "0x" prefix. Dates are given in ISO 8601 format. Algorithm IDs are as specified in Section 9.1 of [RFC9580].

The only required fields are the version and fingerprint of any key material, and the uidString of any User IDs. Implementations MAY omit algorithms, subkeys and User IDs from indexes; however if they are present they MUST contain the required fields.

8.2. Submission Responses

An "add" operation MAY return an empty response, or it MAY return a JSON object summarising the changes. The JSON object MAY contain any or all of the following fields, each of which is an array of certificate objects:

Field	Type	Description
inserted	certificate array	newly added certificates
updated	certificate array	updated certificates
deleted	certificate array	deleted certificates
ignored	certificate array	certificates that were discarded

Table 6: Submission Response Fields

Each certificate object MUST contain "version" and "fingerprint" fields, as in Table 2.

8.3. Legacy machine-readable Output

Clients requesting machine-readable output in Legacy requests:

* SHOULD supply "options=mr" (Section 7.1.1).

* MUST silently ignore any content preceding or following a returned armored key block.

* MUST silently ignore any primary keys with unknown versions or algorithms.

Keyservers returning Legacy machine-readable output:

* MUST set the HTTP header "Access-Control-Allow-Origin: *", as specified in [CORS].

* MUST return ASCII-armored certificate bundles.

* MUST NOT return version 6 (or later) certificates.

* MUST set "Content-Type: application/pgp-keys" when returning ASCII-armored certificate bundles (the "get", "vfpget", "kidget", and "hget" operations), as specified in Section 7 of [RFC3156].

* MUST use the format specified in Section 8.3.1 when returning indexes (the "index" and "vindex" operations).

* MAY return statistics in JSON format [RFC8259], the schema of which is otherwise implementation-dependent.

ASCII-armored responses MAY be wrapped in any HTML or other text desired, except that the actual certificate data consisting of an initial line break, the "-----BEGIN PGP PUBLIC KEY BLOCK-----" header, the armored certificate data itself, the "-----END PGP PUBLIC KEY BLOCK-----" footer, and a final line break MUST NOT be modified from the form specified in [RFC9580].

8.3.1. Legacy machine-readable Indexes

The Legacy machine-readable index format is a list of newline-separated records, consisting of colon-separated fields. The document is 7-bit clean, and as such is sent with no encoding and Content-Type: text/plain.

The machine-readable response MAY be prefixed by an information record:

info:<version>:<count>

Field	Description
version	the version of this output format
count	the number of certificates returned

Table 7: Legacy Information Record Fields

If this line is not included, or the version information is not supplied, the version number is assumed to be 1. Currently, only version 1 is defined.

Note that "count" is the number of certificates, and not the number of lines returned. That is, it SHOULD match the number of "pub:" lines returned.

The certificate listings themselves are made up of several records per certificate. The first record specifies the primary key:

pub:<keyID>:<code>:<bitLength>:<creation>:<expiration>:<flags>

Field	Description
keyID	fingerprint or long Key ID
code	algorithm ID
bitLength	key length in bits
creation	creation date of the key
expiration	expiration date of the key
flags	letter codes to indicate details of the key

Table 8: Legacy Public Key Record Fields

Since it is not possible to calculate the Key ID from a V3 fingerprint, for V3 primary keys the "keyID" field SHOULD contain the 16-digit long Key ID only. Otherwise, a keyserver SHOULD return a fingerprint if available (see Section 7.3).

Algorithm IDs are as specified in Section 9.1 of [RFC9580], i.e. 1==RSA, 17==DSA, etc.

Following the "pub" record are one or more "uid" records to indicate User IDs on the certificate:

uid:<uidString>:<creation>:<expiration>:<flags>

Field	Description
uidString	User ID string contents
creation	creation date of (the first self-signature over) the User ID
expiration	expiration date of the User ID
flags	letter codes to indicate details of the User ID

Table 9: Legacy User ID Record Fields

The User ID string MUST use urlencoding for anything that isn't 7-bit safe as well as for the ":" and "%" characters. Any other characters MAY be urlencoded, as desired.

The information for the "creation", "expiration", and "flags" fields is taken from the User ID self-signature, if any, and applies to the User ID in question, not to the certificate as a whole.

Primary key and User ID records may contain a "flags" field containing a sequence of alphabetical characters, one per flag. Flags MAY be given in any order. The meaning of "disabled" is implementation-specific. Note that individual flags may be unimplemented, so the absence of a given flag does not necessarily mean the absence of the detail.

Flag	Description
r	revoked
d	disabled
e	expired

Table 10: Legacy Record Flags

Note that empty fields are allowed. For example, a primary key with no expiration date would have the "expirationdate" field empty. Also, a keyserver that does not track a particular piece of information may leave that field empty as well. Colons for empty fields on the end of each line MAY be left off, if desired. All dates are given in the number of seconds since 1970-01-01T00:00:00 UTC.

For backwards compatibility with the installed client base, Legacy machine-readable lookup requests MUST omit version 6 (and later) certificates from the returned indexes.

9. Confidence

Traditionally, keyservers did not perform any checks against uploaded content other than simple parseability. This left them open to abuse such as flooding and third-party signature spam. Most modern keyservers are now able to perform cryptographic validity tests, and automated content moderation is generally possible.

It is RECOMMENDED that a keyserver assigns a "confidence" value to each User ID in its database. The exact definition of "confidence" is implementation-dependent, but MAY include checks such as email verification or third-party certifications.

A keyserver MAY represent confidence as a number between 0 and 255, where values of 120 or greater indicate "complete confidence", in a v2 Index User ID object (see Table 4). This is numerically compatible with the "trust amount" field specified in Section 5.2.3.21 of [RFC9580], but it is not required to calculate it in the same way or represent it internally as an integer value. The confidence field in a v2 Index User ID object is intended as a heuristic for sorting and filtering responses to lookup requests, and is not a replacement for cryptographic verification. A client SHOULD NOT rely on this confidence field, and SHOULD perform its own checks. A keyserver that wishes to publish a cryptographically-verifiable statement about its internal confidence value MAY do so using a certification signature.

10. Certificate Bundle Format

HKP uses a "certificate bundle" as its primary data representation for both input and output.

A certificate bundle is a sequence of one or more OpenPGP certificates (Transferable Public Keys), concatenated directly, as specified in Sections 10.1 and 3.6 of [RFC9580]. An ASCII-armored certificate bundle is a certificate bundle that has been encoded as a single armored block, as specified in Section 6.2 of [RFC9580].

Certificate bundles are often called "keyrings", however the term "keyring" is used to refer to several related but distinct concepts:

* A sequence of one or more Transferable Public Keys ("public keyring")

* A sequence of one or more Transferable Secret Keys ("private/secret keyring")

* A sequence of packets forming a single Transferable Public Key

* A sequence of packets forming a single Transferable Secret Key

It is therefore RECOMMENDED that implementations avoid using the term "keyring" without qualification.

10.1. Detached Revocations

For OpenPGP certificates prior to version 6, revocation signatures have customarily been distributed as a detached "revocation certificate", as per Section 10.1.3 of [RFC9580]. An HKP server SHOULD allow submission of these detached revocations.

An HKP implementation MAY accept or serve an ASCII-armored "mixed certificate bundle" where one or more revoked certificates have been replaced by their detached revocation certificate(s). Such a "mixed certificate bundle" MUST be sorted so that all detached revocation certificates appear first. This ensures that detached revocations cannot be mistaken for signatures over another primary key.

Mixed certificate bundles MUST NOT be served in responses to v2 lookup requests.

11. Security Considerations

As described here, a keyserver is a searchable database of OpenPGP Certificates accessed over the network. While there may be security considerations arising from distributing arbitrary certificates in this manner, this does not impact the security of OpenPGP itself.

Without some sort of trust relationship between the client and server, information returned from a keyserver in arbitrary search results cannot be trusted by the client until the OpenPGP client actually retrieves and checks the certificate for itself. This is important and must be stressed: without a specific reason to treat information otherwise, all search results SHOULD be regarded as untrustworthy and informational only.

12. IANA Considerations

This document allocates the ports 11371 and 11372, and the URI schemes "hkp" and "hkps".

12.1. Updated Registry Entries

IANA is requested to update the contact details for port 11371 in the "Service Name and Transport Protocol Port Number" registry to "Daphne Shaw".

12.2. New Registry Entries

IANA is requested to add the following entry to the "Service Name and Transport Protocol Port Number" registry as per [RFC6335]:

Service Name	Port	Transport Protocol	Description	Contact
hkps	11372	tcp and udp	OpenPGP HTTP Keyserver (Secure)	Daphne Shaw

Table 11: Service Name and Transport Protocol Port Number Registry

IANA is requested to add the following entries to the "URI Schemes" registry as per [RFC7595]:

URI Scheme	Description	Status	Reference
hkp	OpenPGP HTTP Keyserver	Permanent	This document
hkps	OpenPGP HTTP Keyserver (Secure)	Permanent	This document

Table 12: Uniform Resource Identifier (URI) Schemes Registry

13. References

13.1. Normative References

[CORS] "Cross Origin Resource Sharing", n.d., <https://fetch.spec.whatwg.org/#cors-protocol>.

[I-D.gallagher-openpgp-media-types] Gallagher, A., "Media Types for OpenPGP", Work in Progress, Internet-Draft, draft-gallagher-openpgp-media-types-00, 8 August 2025, <https://datatracker.ietf.org/doc/html/draft-gallagher-openpgp-media-types-00>.

[RFC1866] Berners-Lee, T. and D. Connolly, "Hypertext Markup Language - 2.0", RFC 1866, DOI 10.17487/RFC1866, November 1995, <https://www.rfc-editor.org/rfc/rfc1866>.

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

[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/rfc/rfc2119>.

[RFC3156] Elkins, M., Del Torto, D., Levien, R., and T. Roessler, "MIME Security with OpenPGP", RFC 3156, DOI 10.17487/RFC3156, August 2001, <https://www.rfc-editor.org/rfc/rfc3156>.

[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/rfc/rfc8174>.

[RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <https://www.rfc-editor.org/rfc/rfc8259>.

[RFC9110] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "HTTP Semantics", STD 97, RFC 9110, DOI 10.17487/RFC9110, June 2022, <https://www.rfc-editor.org/rfc/rfc9110>.

[RFC9580] Wouters, P., Ed., Huigens, D., Winter, J., and Y. Niibe, "OpenPGP", RFC 9580, DOI 10.17487/RFC9580, July 2024, <https://www.rfc-editor.org/rfc/rfc9580>.

13.2. Informative References

[I-D.dkg-openpgp-1pa3pc] Gillmor, D. K., "First-Party Approved Third-Party Certifications in OpenPGP", Work in Progress, Internet-Draft, draft-dkg-openpgp-1pa3pc-02, 6 September 2024, <https://datatracker.ietf.org/doc/html/draft-dkg-openpgp-1pa3pc-02>.

[RFC6335] Cotton, M., Eggert, L., Touch, J., Westerlund, M., and S. Cheshire, "Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry", BCP 165, RFC 6335, DOI 10.17487/RFC6335, August 2011, <https://www.rfc-editor.org/rfc/rfc6335>.

[RFC7595] Thaler, D., Ed., Hansen, T., and T. Hardie, "Guidelines and Registration Procedures for URI Schemes", BCP 35, RFC 7595, DOI 10.17487/RFC7595, June 2015, <https://www.rfc-editor.org/rfc/rfc7595>.

[SKS] "Synchronising Key Server Wiki", n.d., <https://github.com/sks-keyserver/sks-keyserver/wiki>.

[TOR] "Tor Specifications", n.d., <https://spec.torproject.org/>.

Appendix A. Acknowledgments

This document is a formalization and extension of HKP, originally implemented in the PKS keyserver by Marc Horowitz, which in turn was based on earlier work by Brian LaMacchia and Michael Graff. The "prefixlog" operation is based on an earlier proposal by Daniel Kahn Gillmor and Vincent Breitmoser.

The authors would like to thank Peter Gutmann for his work on the Certstore protocol, some of which was applicable here, and the members of the pgp-keyserver-folk mailing list who contributed valuable comments and suggestions. They would also like to thank Bart Butler, Daniel Huigens, Daniel Kahn Gillmor, Heiko Sch辰fer, Justus Winter and Vincent Breitmoser for help with the v2 request format.

Appendix B. Document History

Note to RFC Editor: this section should be removed before publication.

B.1. Changes Between draft-gallagher-openpgp-hkp-07 and draft-gallagher-openpgp-hkp-08

* Verified uploads now use prove-then-submit workflow.
* Removed SRV and discovery discussion (temporarily?).
* Added definitions of "discovery", "refresh" and "reference resolution".
* Normalised "certificate" terminology and warned about unqualified use of "keyring".
* Renumbered HKPv1 to HKPv2 for avoidance of confusion, and reordered path components.
* HKPv2 is now explicitly a binary protocol, and uses simplified paths.
* Defined v2 submission requests and "cb" option.

* Explicitly forbade mixed certificate bundle responses to v2 lookups.

* "since" is now "prefixlog".

* "get" operation is now MAY.

* "x-*" parameters are no longer specified (as per RFC6648).

* Fixed several errata.

* Expanded commentary and guidance.

B.2. Changes Between draft-gallagher-openpgp-hkp-06 and draft-gallagher-openpgp-hkp-07

* Added "authget" and "since" operations.

* Defined confidence.

* Added more explicit guidance re sorting, filtering and error codes.

* Key version MR output field is now "keyversion" to distinguish from the output format version.

B.3. Changes Between draft-gallagher-openpgp-hkp-05 and draft-gallagher-openpgp-hkp-06

* Updated references.

B.4. Changes Between draft-gallagher-openpgp-hkp-04 and draft-gallagher-openpgp-hkp-05

* Allow detached revocations in keyrings.

* Redesigned v2 request format to use path components for required fields.

* Added openpgpkey discovery file.

* Added "vfpget" and "kidget" operations.

* Added meaningful "exact" semantics.

* HKPS is now SHOULD.

* Defined port 11372.

* IANA registry tables.

* Deprecated op=stats.

B.5. Changes Between draft-gallagher-openpgp-hkp-03 and draft-gallagher-openpgp-hkp-04

* Reworded Section 5 for clarity.

* Separate section for keyring format.

* Specify detached revocations.

* Updated references.

B.6. Changes Between draft-gallagher-openpgp-hkp-02 and draft-gallagher-openpgp-hkp-03

* Clients SHOULD supply the v=1 api-versioning variable.

* machine-readable output includes key version field.

* Clients MUST silently ignore leading and trailing cruft, trailing unknown fields, and unknown flags.

* Clients MUST silently ignore keys with unknown versions or algorithms.

* All other m-r index specs (CORS, Content-Type etc.) are now MUST.

* Included the hash variable from SKS.

B.7. Changes Between draft-gallagher-openpgp-hkp-01 and draft-gallagher-openpgp-hkp-02

* Tightened up BCP-14 language.

* Included op=hget from SKS.

* Options now strictly boolean with default false, variables less strict.

* More detail about HTTP status code usage.

B.8. Changes Between draft-shaw-openpgp-hkp-00 and draft-gallagher-openpgp-hkp-01

* Improved text structure.
* Added references to HTTPS/HKPS, and hkp:/hkps: URL schemes.
* Forbade short IDs and deprecated V3 keys.
* Included op=stats from SKS.
* Mentioned CORS.
* Made use of terminology more consistent.
* Replaced custom status codes with standard HTTP status codes.

Authors' Addresses

Daphne Shaw
Jabberwocky Tech
Email: dshaw@jabberwocky.com

Andrew Gallagher (editor)
PGPKeys.EU
Email: andrewg@andrewg.com