Network Working Group
Internet-Draft
Obsoletes: 4767 (if approved)
Intended status: Standards Track
Expires: 28 September 2026

G. Lehmann
Telecom Sud Paris

27 March 2026

Transport of Incident Detection Message Exchange Format version 2 (IDMEFv2) Messages over HTTPS

draft-lehmann-idmefv2-https-transport-06

Abstract

The Incident Detection Message Exchange Format version 2 (IDMEFv2) defines a data representation for security incidents detected on cyber and/or physical infrastructures.

This draft is maintained by the IDMEFv2 Task Force. Please consult our website for more information. https://www.idmefv2.org

The format is agnostic so it can be used in standalone or combined cyber (SIEM), physical (PSIM) and availability (NMS) monitoring systems. IDMEFv2 can also be used to represent man made or natural hazards threats.

IDMEFv2 improves situational awareness by facilitating correlation of multiple types of events using the same base format thus enabling efficient detection of complex and combined cyber and physical attacks and incidents.

This document defines a way to transport IDMEFv2 Alerts over HTTPs.

If approved this document would obsolete RFC4767.

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 28 September 2026.

Table of Contents

1. Introduction

[RFC-DEV-IDMEFv2] defines a model for the purpose of describing security alerts and incidents as IDMEF version 2 (IDMEFv2) messages. It also defines serialization methods so that IDMEFv2 messages can be shared between IDMEFv2 Analyser detecting incidents and IDMEFv2 Manager, the management systems that may need to interact with them.

1.1. About the transmission protocol

IDMEFv2 defines a message format, not a protocol, as the sharing of messages is assumed to be out of scope in order to allow Analyzers and Managers to exchange and store alerts in a way most suited to their established incident-detection processes. However, a protocol specification is required so that actual exchanges can take place between the involved programs. Defining a common protocol also ensures interoperability between otherwise concurrent implementations.

This document specifies the transport of IDMEFv2 messages within [RFC9112] or [RFC9113] Request and Response messages over Transport Layer Security, a.k.a. [RFC9110].

This document describes a serialization method for IDMEF messages based on [RFC8259].

2. Terminology

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

3. Normative sections

Implementations of IDMEFv2 willing to exchange messages with other implementations are REQUIRED to fully implement:

* The transmission protocol defined in Section 4

* The JavaScript Object Notation (JSON) serialization method defined in Section 5

4. Transmission of IDMEFv2 Messages over HTTPS

This section specifies the details of the transport of IDMEFv2 messages [RFC-DEV-IDMEFv2] over HTTPS.

[BCP56] contains a number of important considerations when using HTTP for application protocols. These include the size of the payload for the application, whether the application will use a web browser, whether the protocol should be defined on a port other than the well-known port for HTTP/HTTPS, and if the security provided through HTTPS suits the needs of the new application.

It is acknowledged within the scope of these concerns that HTTPS is not ideally suited for IDMEFv2 transport, as the former is a client-server protocol and the latter a message-exchange protocol; however, the ease of implementation over HTTPS outweighs these concerns.

4.1. Architecture overview

In the context of this document, it is expected that IDMEFv2 Analyzers will act as HTTP clients, while IDMEFv2 Managers act as HTTP servers.

However, due to external constraints, implementation-specific design decisions, etc. these roles may sometimes be switched around. This can be especially true when dealing with segmented networks such as demilitarized zones, where the Analyzer may be unable to contact its Manager, or for communications which do not rely on an Analyzer sending IDMEFv2 messages to a Manager (e.g. for Manager-to-Manager communications). Such specific cases are outside the scope of this specification.

4.2. Listening port

Consistent with [BCP56], compatible system participating in a consortium and hosting a server for the purpose of receiving IDMEFv2 message using this specification SHOULD listen for TCP connections on port TBD (see in Section 7 for more information).

The systems MAY be configurable so as to allow listening on ports other than the well-known port; this configuration is out of scope for this specification.

4.3. Compatibility with HTTP(S)

The implementations MUST implement all REQUIRED functionality for [RFC9112]. The implementations SHOULD implement all REQUIRED functionality for [RFC9113] in a TLS context. In particular, the implementations SHOULD support HTTP/2 protocol negotiation using the [RFC7301] as defined in [RFC9113] ch 3.3. The "h2c" protocol (HTTP/2.0 over cleartext) MUST NOT be used. HTTP over TLS (also known as HTTPS) is specified in [RFC9110] ch 2.

Each IDMEFv2 message MUST be sent as a separate HTTP Request. However, an implementation MAY send multiple HTTP Requests in parallel to achieve higher message throughput. This can be done by creating a separate connection to the HTTP server for each request, or by using any multiplexing mechanism provided by the underlying protocol (see for example [RFC9113] ch 5).

All IDMEFv2 messages sent in HTTP Requests MUST be sent using the POST method. In addition, the Request-URI of such requests SHOULD be set to '/' for interoperability, but an implementation MUST also be able to handle alternative paths to cope with situations where URI rewriting may be used (e.g. by a reverse proxy server).

As IDMEFv2 messages MUST be sent using the POST method, a compatible implementation SHOULD answer with an appropriate HTTP code (405 Method Not Allowed) to requests made using any other method.

If a request is made to a Request-URI other than the one dedicated to the processing of IDMEFv2 messages, the server SHOULD return an appropriate HTTP code (e.g. 404 Not Found).

The HTTP Request's body MUST be set to the serialized form of the IDMEFv2 message. Since an IDMEFv2 message can be serialized in a variety of ways, the 'Content-Type' Request header [RFC2045] MUST be set to a media type [RFC2046] that reflects the mechanism used to perform the serialization. If the server does not support the media type sent by the client, it SHOULD respond with an appropriate HTTP code (415 Unsupported Media Type).

A conforming IDMEFv2 system acting as an HTTP server MUST support content negotiation based on the value of the 'Accept' Request header sent by the client. If the client does not advertise supported media types (i.e. omits the 'Accept' header), the server MUST default to 'application/json' and use the format defined in [RFC8259] to build the HTTP response. If the server does not support any of the media types advertised by the client, it SHOULD respond with an appropriate HTTP code (406 Not Acceptable).

If the Response does not contain a body, the appropriate HTTP code SHOULD be returned to the client (204 No Content). In this case, the 'Content-Type' header can be omitted from the HTTP Response entirely.

To improve compatibility between implementations, it is RECOMMENDED that all IDMEFv2 systems conforming to this document support the JSON serialization format defined in [RFC-DEV-IDMEFv2], with a media type set to 'application/json'.

If an IDMEFv2 system receives an improper IDMEFv2 message in an HTTP Request, it MUST return an appropriate 4xx Client Error result code to the requesting system (e.g. 400 Bad Request).

The HTTP code serves as an acknowledgment of the message. The receiving IDMEFv2 system SHOULD NOT confirm acceptance of a message (2xx) unless it managed to safely deal with it, e.g. by saving the message to disk / in a database, or by relaying it to another system that successfully acknowledged it for further processing.

If an IDMEFv2 implementation cannot process an IDMEFv2 message received in an HTTP Request due to an error on its own side, it MUST return an appropriate 5xx Server Error result code to the requesting system.

Note that HTTP provides no mechanism for signaling to a server that a Response body is not a valid IDMEFv2 response. If an IDMEFv2 system receives an improper HTTP Response, or cannot process an HTTP Response due to an error on its own side, it MUST log the error and present it to the IDMEFv2 system administrator for handling; the error logging format is considered an implementation detail and is out of scope for this specification.

Appendix A provides informative guidance on how to map various error conditions to an appropriate HTTP result code.

IDMEFv2 systems relying on HTTP/1.1 MUST support and SHOULD use HTTP/1.1 persistent connections as described in [RFC9112]. IDMEFv2 systems MUST support chunked transfer encoding on the HTTP server side to allow the implementation of clients that do not need to pre-calculate message sizes before constructing HTTP headers.

4.4. Compatibility with DNS

The use of stable DNS names to address IDMEFv2 systems is RECOMMENDED; this facilitates connection to IDMEFv2 systems within a consortium. The protocol provides no in-band method for handling a change of address of an IDMEFv2 system.

4.5. Compatibility with TLS

4.5.1. Acceptable TLS versions

IDMEFv2 systems SHOULD use TLS version 1.3 [RFC8446] or higher for confidentiality, identification, and authentication, when sending IDMEFv2 messages over HTTPS.

IDMEFv2 systems MUST NOT request, offer, or use any version of SSL, or any version of TLS prior to 1.3, due to known weaknesses in these protocols; see [RFC8446] ch E for more information.

4.5.2. Acceptable ciphersuites

The TLS session MUST use non-NULL ciphersuites for authentication, integrity, and confidentiality. IDMEFv2 systems MUST NOT use ciphersuites with known vulnerabilities and MUST use ciphersuites that provide Perfect Forward Secrecy to protect the messages' confidentiality in case a vulnerability is later discovered in the ciphersuites.

Administrators can find guidance regarding the selection of acceptable ciphersuites for TLS version 1.2 in [BCP195].

4.5.3. Authentication

IDMEFv2 systems MUST use mutual authentication; that is, both IDMEFv2 systems acting as HTTPS clients and IDMEFv2 systems acting as HTTPS servers MUST be identified by an X.509 certificate [RFC5280]. Mutual authentication requires full path validation on each certificate, as defined in [RFC5280].

All IDMEFv2 systems SHOULD be identified by a certificate containing a DNS-ID identifier as in [RFC6125] ch 6.4; Common Names (CN-IDs) MUST NOT be included in certificates identifying IDMEFv2 systems.

4.5.4. Certificate validation

IDMEFv2 systems MUST verify the reference identifiers of their peers against those stored in the certificates presented using the methods defined in [RFC6125]. In addition, the following IDMEFv2-specific considerations apply:

* Wildcards MUST NOT appear in the DNS-ID of a certificate identifying an IDMEFv2 system. If such a certificate is received by an IDMEFv2 system, the verification MUST fail and the receiving IDMEFv2 system MUST close the connection.

* An IDMEFv2 system MAY match the DNS-ID with a reverse DNS lookup of the connecting IDMEFv2 peer; this support SHOULD allow the lookup to be cached and/or done in advance in order to ensure verifiability during instability or compromise of DNS itself.

* IDMEFv2 systems MUST support the verification of certificates against an explicit list of approved peer certificates.

* IDMEFv2 systems MAY apply additional security measures such as IP filtering to further restrict the list of authorized peers. Such additional defenses are outside of the scope of this specification.

4.5.5. PKI integration and certificate issuance

This document makes no provision on whether the security consortium implementing IDMEFv2 is responsible or not for issuing the certificates used by the systems.

In a PKIX certificate to be presented by an IDMEFv2 system, the certificate MUST include one or more identifiers associated with the IDMEFv2 system.

The following IDMEFv2-specific considerations apply:

* Support for the DNS-ID identifier type [RFC5280] is REQUIRED in both IDMEFv2 Analyzers and Managers. Certification authorities that issue IDMEFv2-specific certificates MUST support the DNS-ID identifier type. IDMEFv2 service providers SHOULD include the DNS-ID identifier type in certificate requests.

* Support for the SRV-ID identifier type [RFC4985] is REQUIRED in both IDMEFv2 Analyzers and Managers implementations using the "_idmef" application service type. Certificate authorities that issue IDMEFv2-specific certificates SHOULD support the SRV-ID identifier type. IDMEFv2 service providers SHOULD include the SRV-ID identifier type in certificate requests.

* Support for the URI-ID identifier type ([RFC5280] and [RFC3986]) is OPTIONAL for both IDMEFv2 Analyzers and Managers. Certification authorities that issue IDMEFv2-specific certificates SHOULD support the URI-ID identifier type. IDMEFv2 service providers MAY include the URI-ID identifier type in certificate requests.

* Support for the CN-ID identifier type [RFC5280] is discouraged. Certification authorities that issue IDMEFv2-specific certificates SHOULD NOT support the CN-ID identifier type. IDMEFv2 service providers MUST NOT include the CN-ID identifier type in certificate requests.

* DNS domain names in IDMEFv2-specific certificates MUST NOT contain the wildcard character '*'. Certification authorities that issue IDMEFv2-specific certificates MUST refuse to issue certificates containing a wildcard character.

Great care must be taken by administrators when selecting the PKI to use, as trusting a rogue PKI could have dramatic results on the integrity, confidentiality and authentication mechanisms provided by TLS; and hence, result in the compromise of the protocol defined in this specification.

4.6. Locating the HTTPS endpoint using DNS

Information about an organization’s IDMEFv2 over HTTPS endpoint’s location may be stored using DNS Service Location Records (SRV) [RFC2782]. The data in a SRV record contains the DNS name of the server that provides the IDMEFv2 over HTTPS server, the corresponding Port number, and parameters that enable the client to choose an appropriate server from multiple servers according to the algorithm described in [RFC2782]. The name of this record has the following format:

<Service>.<Proto>.<Domain>

where <Service> is always "idmef", and <Proto> is always "tcp". <Domain> is the domain name of the entity exposing the endpoint. Note that "idmef" is the symbolic name for the IDMEFv2 over HTTPS service in Assigned Numbers [RFC3232], as required by [RFC2782].

Presence of such records enables clients to find the IDMEFv2 over HTTPS endpoints using standard DNS query. An IDMEFv2 system seeking the endpoint for a particular entity, does a SRV record query using the DNS name formed as described in the preceding paragraph, and interprets the response as described in [RFC2782] to determine a host (or set of hosts) to contact. As an example, a client that searches for the IDMEFv2 over HTTPS endpoint for "example.net" will submit a DNS query for a set of SRV records with owner name:

_idmef._tcp.example.net.

The requesting system will receive the list of SRV records published in DNS that satisfy the requested criteria. The following is an example of such a record:

_idmef._tcp.example.net. IN SRV 0 0 TBD soc.example.net.

The set of returned records may contain multiple records in the case where multiple servers can act as endpoints for the same domain. If there are no matching SRV records available for the domain name, the client SHOULD NOT attempt to ’walk the tree’ by removing the least significant portion of the constructed fully qualified domain name.

5. JavaScript Object Notation Serialization Method

This serialization method aims to convert IDMEFv2 messages to a format that is easy to parse and process, both by software/hardware processors, as well as humans.

It relies on the the JavaScript Object Notation (JSON) Data Interchange Format defined in [RFC8259].

Conforming implementations MUST implement all the requirements specified in [RFC8259].

In addition, the following rules MUST be observed when serializing an IDMEFv2 message:

* The top-level Alert class ([RFC-DEV-IDMEFv2] ch 4.2) is represented as a JSON object ([RFC8259] ch 4). This JSON object is returned to the calling process at the end of the serialization process.

* Aggregate classes are represented as JSON objects and stored as members of the top-level JSON object, using the same name as in the IDMEF data model. E.g. the Analyzer aggregate class ([RFC-DEV-IDMEFv2] ch 4.3) appears under the name "Analyzer" inside the top-level JSON object.

* Attributes are stored as members of the JSON object representing the class they belong to, using the same name as in the IDMEF data model. E.g. the "Version" attribute of the Alert class is stored under the name "Version" inside the top-level JSON object.

* Lists from the IDMEF data model are represented as JSON arrays ([RFC8259] ch 5). This also applies to aggregate classes where a list is expected. E.g. the "Sensor" member inside the top-level JSON object contains a list of objects, where each object represents an instance of the Sensor aggregate class ([RFC-DEV-IDMEFv2] ch 4.4).

* The various string-based data types listed in [RFC-DEV-IDMEFv2] ch 3.3 are represented as JSON strings ([RFC8259] ch 7). Please note that the issues outlined in [RFC8259] ch 8 regarding strings processing also apply here.

* Since JSON does not provide a way to distinguish between an integer value and a floating-point (decimal) value, IDMEF attributes with the "INT" and "FLOAT" type annotations are both represented as JSON numbers ([RFC8259] ch 6).

6. Security Considerations

Most of the content in Section 4 deals with security considerations. Great care has been taken to protect the confidentiality, authentication and integrity of IDMEFv2 messages while in transit.

In addition, to the requirements set in Section 4, all security considerations of related documents apply, especially the Incident Detection Message Exchange Format version 2 [RFC-DEV-IDMEFv2]. Implementations should also be aware of known attacks against Transport Layer Security (TLS) and Datagram TLS (DTLS), and how to mitigate them [RFC7457].

Specific considerations have been taken into account regarding certificates usage:

* Use of the wildcard character inside certificates lacks clear specifications and consistency across application technologies. Considering the issues outlined in [RFC6125] ch 7.2 regarding such certificates, the use of wildcard characters inside certificates is explicitly prohibited in this document.

* Use of the Common Name field (CN-ID) has been deprecated for a long time ([RFC9110] ch 3.1). Moreover, this field does not cope well with multiple identifiers ([RFC6125] ch 7.4). As such, this document explicitly prohibits the use of CN-IDs altogether.

IDMEFv2 systems MAY implement additional security measures (e.g. confidentiality and integrity of specific data inside the message at the field level). Such additions SHOULD be designed with interoperability in mind, so that implementations that do not recognize these extensions can still process IDMEFv2 messages in a reasonable manner without degrading the security provided by these additions (e.g. by omitting the additions entirely when passing the message to another system, or by handling them as opaque data). The design of such measures is outside the scope of this document.

7. IANA Considerations

Consistent with [BCP56], since IDMEFv2 over HTTP/TLS is a substantially new service, and should be controlled at the consortium member network’s border differently than HTTP/TLS, it requires a new port number.

IANA will assigned port TBD/tcp to IDMEFv2 over HTTP/TLS with service name "IDMEFv2".

Registration template:

* Service Name: IDMEFv2

* Transport Protocol: TCP

* Assignee: Telecom SudParis

* Contact: Telecom SudParis (Herve.Debar @telecom-sudparis.eu)

* Description: Transport of Incident Detection Message Exchange Format version 2 (IDMEFv2) Messages over HTTPS

* Reference: TBD

* Port Number: TBD

8. Acknowledgement

The following groups and individuals contributed to the creation of this document and should be recognized for their efforts.

* Thomas Andrejak & Fran巽ois Poirotte (Co-authors of the first version of this document)

9. References

9.1. 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>.

[RFC-DEV-IDMEFv2] Lehmann, G., "The Incident Detection Message Exchange Format (IDMEF) version 2", Work in Progress, Internet-Draft, draft-lehmann-idmefv2-01, 27 March 2026, <https://github.com/IDMEFv2/>.

[RFC9112] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "HTTP/1.1", STD 99, RFC 9112, DOI 10.17487/RFC9112, June 2022, <https://www.rfc-editor.org/info/rfc9112>.

[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, <https://www.rfc-editor.org/info/rfc3986>.

[RFC9113] Thomson, M., Ed. and C. Benfield, Ed., "HTTP/2", RFC 9113, DOI 10.17487/RFC9113, June 2022, <https://www.rfc-editor.org/info/rfc9113>.

[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/info/rfc9110>.

[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>.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <https://www.rfc-editor.org/info/rfc5280>.

[RFC4985] Santesson, S., "Internet X.509 Public Key Infrastructure Subject Alternative Name for Expression of Service Name", RFC 4985, DOI 10.17487/RFC4985, August 2007, <https://www.rfc-editor.org/info/rfc4985>.

[RFC6125] Saint-Andre, P. and J. Hodges, "Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS)", RFC 6125, DOI 10.17487/RFC6125, March 2011, <https://www.rfc-editor.org/info/rfc6125>.

[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/info/rfc8259>.

9.2. Informative References

[RFC3232] Reynolds, J., Ed., "Assigned Numbers: RFC 1700 is Replaced by an On-line Database", RFC 3232, DOI 10.17487/RFC3232, January 2002, <https://www.rfc-editor.org/info/rfc3232>.

[RFC2045] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, DOI 10.17487/RFC2045, November 1996, <https://www.rfc-editor.org/info/rfc2045>.

[RFC2046] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", RFC 2046, DOI 10.17487/RFC2046, November 1996, <https://www.rfc-editor.org/info/rfc2046>.

[RFC2782] Gulbrandsen, A., Vixie, P., and L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)", RFC 2782, DOI 10.17487/RFC2782, February 2000, <https://www.rfc-editor.org/info/rfc2782>.

[RFC7301] Friedl, S., Popov, A., Langley, A., and E. Stephan, "Transport Layer Security (TLS) Application-Layer Protocol Negotiation Extension", RFC 7301, DOI 10.17487/RFC7301, July 2014, <https://www.rfc-editor.org/info/rfc7301>.

[RFC7457] Sheffer, Y., Holz, R., and P. Saint-Andre, "Summarizing Known Attacks on Transport Layer Security (TLS) and Datagram TLS (DTLS)", RFC 7457, DOI 10.17487/RFC7457, February 2015, <https://www.rfc-editor.org/info/rfc7457>.

[BCP56] Best Current Practice 56, <https://www.rfc-editor.org/info/bcp56>. At the time of writing, this BCP comprises the following:

Nottingham, M., "Building Protocols with HTTP", BCP 56, RFC 9205, DOI 10.17487/RFC9205, June 2022, <https://www.rfc-editor.org/info/rfc9205>.

[BCP195] Best Current Practice 195, <https://www.rfc-editor.org/info/bcp195>. At the time of writing, this BCP comprises the following:

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, <https://www.rfc-editor.org/info/rfc7525>.

Moriarty, K. and S. Farrell, "Deprecating TLS 1.0 and TLS 1.1", BCP 195, RFC 8996, DOI 10.17487/RFC8996, March 2021, <https://www.rfc-editor.org/info/rfc8996>.

Appendix A. HTTP Result Codes

The following table MAY be used as general guidance by IDMEFv2 systems acting as HTTP servers when mapping various error conditions to an appropriate HTTP result code.

Result codes other than the ones listed below MAY still be used, so long as their semantics match the requirements set in Section 4.

Error condition	HTTP result code
No error	2xx, usually 200 (OK) or 204 (No content)
Malformed HTTP request (e.g. the HTTP request's body is not a valid IDMEFv2 message)	400 (Bad request)
Unauthorized request (e.g. a request from an unauthorized IP address)	403 (Forbidden)
The HTTP request uses a method other than 'POST'	405 (Method Not Allowed). In addition, the "Allow" HTTP response header MUST be set accordingly when this code is used.
Failure to negotiate the response's media type based on the "Accept" HTTP request header.	406 (Not Acceptable)
Missing "Content-Length" request header in case the implementation does not support the chunked transfer encoding	411 (Length required), but please note that conforming implementations are REQUIRED to support the chunked transfer encoding
Exceedingly long IDMEFv2 message (e.g. the message is considered to be too big due to size restrictions applied by an administrator)	413 (Request entity too large)
Unsupported serialization format	415 (Unsupported media type)
Internal server error	500 (Internal server error)
Unavailable service (e.g. the IDMEFv2 system is overloaded or the next	503 (Service unavailable)

| processing service is | |
| unavailable) | |

Table 1: Mapping error conditions to HTTP result codes

Appendix B. Transmission examples

In the examples below, ">" denotes a request sent by the IDMEFv2 system acting as an HTTP client to an IDMEFv2 system acting as an HTTP server, while "<" denotes a response to such requests.

B.1. Acknowledged IDMEFv2 message

In this example, the IDMEFv2 system acting as an HTTP server acknowledges the message sent by an IDMEFv2 system acting as an HTTP client. Because the HTTP server does not wish to send any content back to the client, it replies with HTTP result code 204 (No content), and both the "Content-Type" and "Content-Length" headers are omitted from the response.

```
> POST / HTTP/1.1\r\n
> Host: idmefv2.example.com:TBD\r\n
> Content-Type: application/json\r\n
> Content-Length: 1234\r\n
> Accept: application/json
> \r\n
> {"Version": "..."}

< 204 No content\r\n
< Connection: close\r\n
< \r\n
```

B.2. Unsupported serialization format

The following example illustrates the situation where an IDMEFv2 system acting as an HTTP client tries to send a message using a serialization format (media type) that is either not recognized or not supported by the IDMEFv2 system acting as an HTTP server.

```
> POST / HTTP/1.1\r\n
> Host: idmefv2.example.com:TBD\r\n
> Content-Type: application/x-idmefv2\r\n
> Content-Length: 1234\r\n
> Accept: application/json
> \r\n
> ...

< 415 Unsupported media type\r\n
< Connection: close\r\n
< Content-Type: application/json\r\n
< Content-Length: 61\r\n
< \r\n
< {"error": "Unsupported or unrecognized serialization format"}
```

B.3. Content negotiation failure

In this example, the IDMEFv2 system acting as an HTTP client sends an IDMEFv2 message and requests that the server reponds using the "application/x-example-type" media type.

Since the server wishes to include details in the HTTP response but does not know how to generate a response using that media type, it sends back a page using HTTP result code 406 (Not acceptable).

The response’s body may contain additional information about the media types the server is willing to use and the client may try to resend the request using one of the alternative media types presented inside the new HTTP request’s "Accept" header.

```
> POST / HTTP/1.1\r\n
> Host: idmefv2.example.com:TBD\r\n
> Content-Type: application/json\r\n
> Content-Length: 1234\r\n
> Accept: application/x-example-type
> \r\n
> ...

< 406 Not Acceptable\r\n
< Connection: close\r\n
< Content-Type: application/json\r\n
< Content-Length: 111\r\n
< \r\n
< {"error": "Cannot reply using ’application/x-example-type’
  media type",\n"alternatives": ["application/json"]}\n
```

Author’s Address

Gilles Lehmann
Telecom Sud Paris
France
Email: gilles.lehmann@telecom-sudparis.eu