

TLS  
Internet-Draft  
Updates: [6347](#) (if approved)  
Intended status: Standards Track  
Expires: April 23, 2020

E. Rescorla, Ed.  
RTFM, Inc.  
H. Tschofenig, Ed.  
T. Fossati  
Arm Limited  
October 21, 2019

**Connection Identifiers for DTLS 1.2**  
**draft-ietf-tls-dtls-connection-id-07**

Abstract

This document specifies the Connection ID (CID) construct for the Datagram Transport Layer Security (DTLS) protocol version 1.2.

A CID is an identifier carried in the record layer header that gives the recipient additional information for selecting the appropriate security association. In "classical" DTLS, selecting a security association of an incoming DTLS record is accomplished with the help of the 5-tuple. If the source IP address and/or source port changes during the lifetime of an ongoing DTLS session then the receiver will be unable to locate the correct security context.

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 April 23, 2020.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents

(<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

Table of Contents

- [1.](#) Introduction . . . . . [3](#)
- [2.](#) Conventions and Terminology . . . . . [3](#)
- [3.](#) The "connection\_id" Extension . . . . . [3](#)
- [4.](#) Record Layer Extensions . . . . . [5](#)
- [5.](#) Record Payload Protection . . . . . [7](#)
  - [5.1.](#) Block Ciphers . . . . . [7](#)
  - [5.2.](#) Block Ciphers with Encrypt-then-MAC processing . . . . . [8](#)
  - [5.3.](#) AEAD Ciphers . . . . . [8](#)
- [6.](#) Peer Address Update . . . . . [8](#)
- [7.](#) Examples . . . . . [9](#)
- [8.](#) Privacy Considerations . . . . . [11](#)
- [9.](#) Security Considerations . . . . . [11](#)
- [10.](#) IANA Considerations . . . . . [12](#)
- [11.](#) References . . . . . [12](#)
  - [11.1.](#) Normative References . . . . . [12](#)
  - [11.2.](#) Informative References . . . . . [13](#)
- [Appendix A.](#) History . . . . . [14](#)
- [Appendix B.](#) Working Group Information . . . . . [15](#)
- [Appendix C.](#) Contributors . . . . . [15](#)
- [Appendix D.](#) Acknowledgements . . . . . [16](#)
- Authors' Addresses . . . . . [16](#)



## 1. Introduction

The Datagram Transport Layer Security (DTLS) protocol was designed for securing connection-less transports, like UDP. DTLS, like TLS, starts with a handshake, which can be computationally demanding (particularly when public key cryptography is used). After a successful handshake, symmetric key cryptography is used to apply data origin authentication, integrity and confidentiality protection. This two-step approach allows endpoints to amortize the cost of the initial handshake across subsequent application data protection. Ideally, the second phase where application data is protected lasts over a longer period of time since the established keys will only need to be updated once the key lifetime expires.

In the current version of DTLS, the IP address and port of the peer are used to identify the DTLS association. Unfortunately, in some cases, such as NAT rebinding, these values are insufficient. This is a particular issue in the Internet of Things when devices enter extended sleep periods to increase their battery lifetime. The NAT rebinding leads to connection failure, with the resulting cost of a new handshake.

This document defines an extension to DTLS 1.2 to add a CID to the DTLS record layer. The presence of the CID is negotiated via a DTLS extension.

## 2. Conventions and Terminology

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].

This document assumes familiarity with DTLS 1.2 [RFC6347].

## 3. The "connection\_id" Extension

This document defines the "connection\_id" extension, which is used in ClientHello and ServerHello messages.

The extension type is specified as follows.

```
enum {  
    connection_id(TBD1), (65535)  
} ExtensionType;
```

The extension\_data field of this extension, when included in the ClientHello, MUST contain the ConnectionId structure. This structure



contains the CID value the client wishes the server to use when sending messages to the client. A zero-length CID value indicates that the client is prepared to send with a CID but does not wish the server to use one when sending. Alternatively, this can be interpreted as the client wishes the server to use a zero-length CID; the result is the same.

```
struct {  
    opaque cid<0..2^8-1>;  
} ConnectionId;
```

A server willing to use CIDs will respond with a "connection\_id" extension in the ServerHello, containing the CID it wishes the client to use when sending messages towards it. A zero-length value indicates that the server will send with the client's CID but does not wish the client to include a CID (or again, alternately, to use a zero-length CID).

Because each party sends the value in the "connection\_id" extension it wants to receive as a CID in encrypted records, it is possible for an endpoint to use a globally constant length for such connection identifiers. This can in turn ease parsing and connection lookup, for example by having the length in question be a compile-time constant. Implementations, which want to use variable-length CIDs, are responsible for constructing the CID in such a way that its length can be determined on reception. Such implementations must still be able to send CIDs of different length to other parties. Note that there is no CID length information included in the record itself.

In DTLS 1.2, CIDs are exchanged at the beginning of the DTLS session only. There is no dedicated "CID update" message that allows new CIDs to be established mid-session, because DTLS 1.2 in general does not allow TLS 1.3-style post-handshake messages that do not themselves begin other handshakes. When a DTLS session is resumed or renegotiated, the "connection\_id" extension is negotiated afresh.

If DTLS peers have not negotiated the use of CIDs then the [RFC 6347](#)-defined record format and content type MUST be used.

If DTLS peers have negotiated the use of a CIDs using the ClientHello and the ServerHello messages then the peers need to take the following steps.

The DTLS peers determine whether incoming and outgoing messages need to use the new record format, i.e., the record format containing the CID. The new record format with the the `tls12_cid` content type is



only used once encryption is enabled. Plaintext payloads never use the new record type and the CID content type.

For sending, if a zero-length CID has been negotiated then the [RFC 6347](#)-defined record format and content type MUST be used (see [Section 4.1 of \[RFC6347\]](#)) else the new record layer format with the `tls12_cid` content type defined in Figure 3 MUST be used.

When transmitting a datagram with the `tls12_cid` content type, the new MAC computation defined in [Section 5](#) MUST be used.

For receiving, if the `tls12_cid` content type is set, then the CID is used to look up the connection and the security association. If the `tls12_cid` content type is not set, then the connection and security association is looked up by the 5-tuple and a check MUST be made to determine whether the expected CID value is indeed zero length. If the check fails, then the datagram MUST be dropped.

When receiving a datagram with the `tls12_cid` content type, the new MAC computation defined in [Section 5](#) MUST be used. When receiving a datagram with the [RFC 6347](#)-defined record format the MAC calculation defined in [Section 4.1.2 of \[RFC6347\]](#) MUST be used.

#### 4. Record Layer Extensions

This specification defines the DTLS 1.2 record layer format and [\[I-D.ietf-tls-dtls13\]](#) specifies how to carry the CID in DTLS 1.3.

To allow a receiver to determine whether a record has a CID or not, connections which have negotiated this extension use a distinguished record type `tls12_cid(TBD2)`. Use of this content type has the following three implications:

- The CID field is present and contains one or more bytes.
- The MAC calculation follows the process described in [Section 5](#).
- The true content type is inside the encryption envelope, as described below.

Plaintext records are not impacted by this extension. Hence, the format of the DTLSPlaintext structure is left unchanged, as shown in Figure 1.





```

struct {
    ContentType type;
    ProtocolVersion version;
    uint16 epoch;
    uint48 sequence_number;
    uint16 length;
    opaque fragment[DTLSPlaintext.length];
} DTLSPlaintext;

```

Figure 1: DTLS 1.2 Plaintext Record Payload.

When CIDs are being used, the content to be sent is first wrapped along with its content type and optional padding into a `DTLSInnerPlaintext` structure. This newly introduced structure is shown in Figure 2. The `DTLSInnerPlaintext` byte sequence is then encrypted. To create the `DTLSCiphertext` structure shown in Figure 3 the CID is added.

```

struct {
    opaque content[length];
    ContentType real_type;
    uint8 zeros[length_of_padding];
} DTLSInnerPlaintext;

```

Figure 2: New `DTLSInnerPlaintext` Payload Structure.

`content` Corresponds to the fragment of a given length.

`real_type` The content type describing the payload.

`zeros` An arbitrary-length run of zero-valued bytes may appear in the cleartext after the type field. This provides an opportunity for senders to pad any DTLS record by a chosen amount as long as the total stays within record size limits. See [Section 5.4 of \[RFC8446\]](#) for more details. (Note that the term `TLSInnerPlaintext` in [RFC 8446](#) refers to `DTLSInnerPlaintext` in this specification.)

```

struct {
    ContentType special_type = tls12_cid;
    ProtocolVersion version;
    uint16 epoch;
    uint48 sequence_number;
    opaque cid[cid_length]; // New field
    uint16 length;
    opaque enc_content[DTLSCiphertext.length];
} DTLSCiphertext;

```

Figure 3: DTLS 1.2 CID-enhanced Ciphertext Record.



`special_type` The outer content type of a DTLSCiphertext record carrying a CID is always set to `tls12_cid(TBD2)`. The real content type of the record is found in `DTLSInnerPlaintext.real_type` after decryption.

`cid` The CID value, `cid_length` bytes long, as agreed at the time the extension has been negotiated.

`enc_content` The encrypted form of the serialized `DTLSInnerPlaintext` structure.

All other fields are as defined in [RFC 6347](#).

## 5. Record Payload Protection

Several types of ciphers have been defined for use with TLS and DTLS and the MAC calculation for those ciphers differs slightly.

This specification modifies the MAC calculation defined in [[RFC6347](#)] and [[RFC7366](#)] as well as the definition of the additional data used with AEAD ciphers provided in [[RFC6347](#)] for records with content type `tls12_cid`. The modified algorithm MUST NOT be applied to records that do not carry a CID, i.e., records with content type other than `tls12_cid`.

The following fields are defined in this document; all other fields are as defined in the cited documents.

`cid` Value of the negotiated CID.

`cid_length` 1 byte field indicating the length of the negotiated CID.

`length_of_DTLSInnerPlaintext` The length (in bytes) of the serialised `DTLSInnerPlaintext`.  
The length MUST NOT exceed  $2^{14}$ .

Note "+" denotes concatenation.

### 5.1. Block Ciphers

The following MAC algorithm applies to block ciphers that do not use the with Encrypt-then-MAC processing described in [[RFC7366](#)].



```
MAC(MAC_write_key, seq_num +
    tls12_cid +
    DTLSCiphertext.version +
    cid +
    cid_length +
    length_of_DTLSInnerPlaintext +
    DTLSInnerPlaintext.content +
    DTLSInnerPlaintext.real_type +
    DTLSInnerPlaintext.zeros
)
```

### 5.2. Block Ciphers with Encrypt-then-MAC processing

The following MAC algorithm applies to block ciphers that use the with Encrypt-then-MAC processing described in [[RFC7366](#)].

```
MAC(MAC_write_key, seq_num +
    tls12_cid +
    DTLSCipherText.version +
    cid +
    cid_length +
    length of (IV + DTLSCiphertext.enc_content) +
    IV +
    DTLSCiphertext.enc_content);
```

### 5.3. AEAD Ciphers

For ciphers utilizing authenticated encryption with additional data the following modification is made to the additional data calculation.

```
additional_data = seq_num +
    tls12_cid +
    DTLSCipherText.version +
    cid +
    cid_length +
    length_of_DTLSInnerPlaintext;
```

## 6. Peer Address Update

When a record with a CID is received that has a source address different than the one currently associated with the DTLS connection, the receiver MUST NOT replace the address it uses for sending records to its peer with the source address specified in the received datagram unless the following conditions are met:

- The received datagram has been cryptographically verified using the DTLS record layer processing procedures.



- The received datagram is "newer" (in terms of both epoch and sequence number) than the newest datagram received. Reordered datagrams that are sent prior to a change in a peer address might otherwise cause a valid address change to be reverted. This also limits the ability of an attacker to use replayed datagrams to force a spurious address change, which could result in denial of service. An attacker might be able to succeed in changing a peer address if they are able to rewrite source addresses and if replayed packets are able to arrive before any original.
- There is a strategy for ensuring that the new peer address is able to receive and process DTLS records. No such test is defined in this specification.

The above is necessary to protect against attacks that use datagrams with spoofed addresses or replayed datagrams to trigger attacks. Note that there is no requirement to use of the anti-replay window mechanism defined in [Section 4.1.2.6](#) of DTLS 1.2. Both solutions, the "anti-replay window" or "newer algorithm" will prevent address updates from replay attacks while the latter will only apply to peer address updates and the former applies to any application layer traffic.

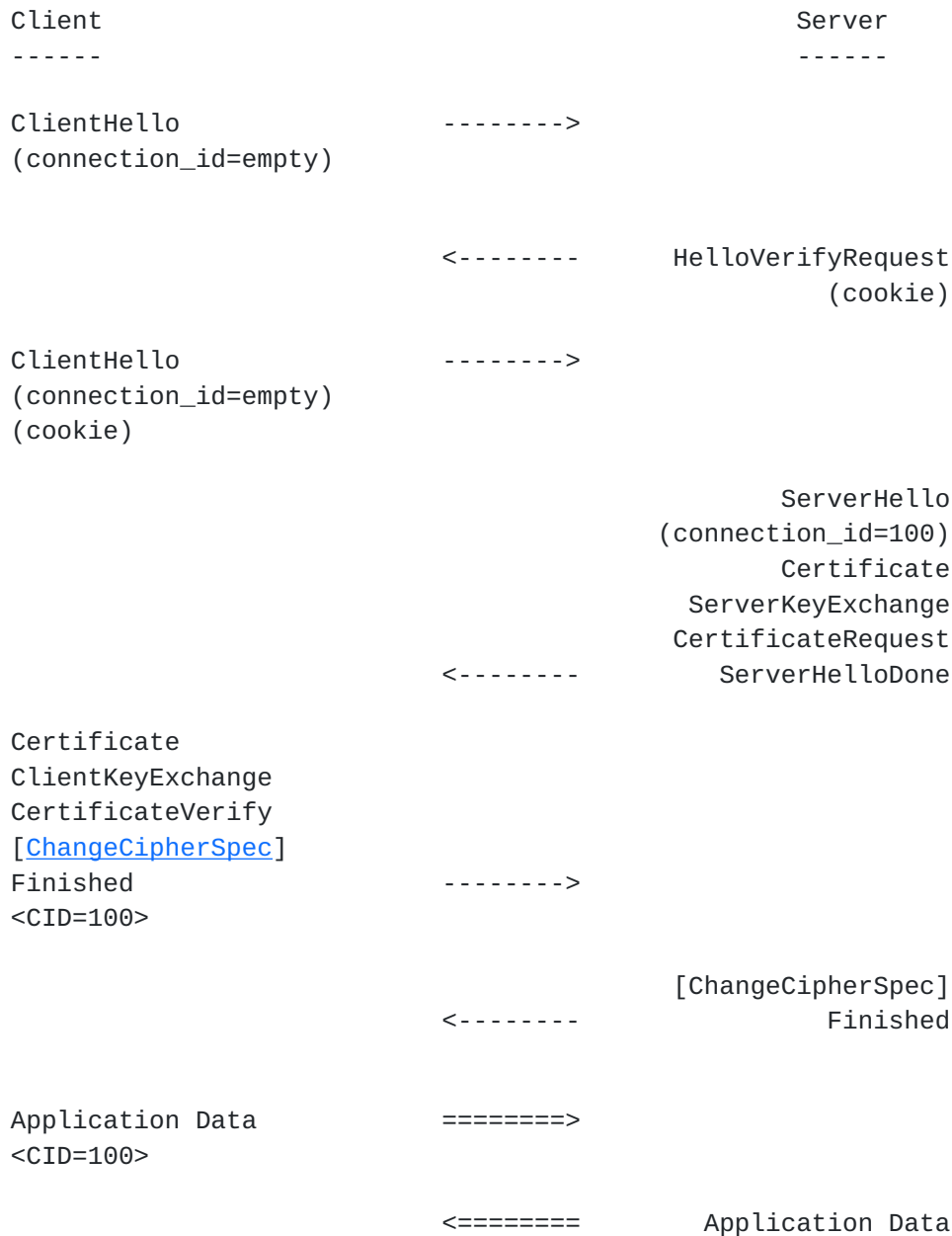
Application protocols that implement protection against these attacks depend on being aware of changes in peer addresses so that they can engage the necessary mechanisms. When delivered such an event, an application layer-specific address validation mechanism can be triggered, for example one that is based on successful exchange of minimal amount of ping-pong traffic with the peer. Alternatively, an DTLS-specific mechanism may be used, as described in [\[I-D.tschofenig-tls-dtls-rrc\]](#).

## 7. Examples

Figure 4 shows an example exchange where a CID is used unidirectionally from the client to the server. To indicate that a zero-length CID we use the term 'connection\_id=empty'.







Legend:

- <...> indicates that a connection id is used in the record layer
- (...) indicates an extension
- [...] indicates a payload other than a handshake message

Figure 4: Example DTLS 1.2 Exchange with CID

Note: In the example exchange the CID is included in the record layer once encryption is enabled. In DTLS 1.2 only one handshake message is encrypted, namely the Finished message. Since the example shows



how to use the CID for payloads sent from the client to the server only the record layer payloads containing the Finished messages include a CID. Application data payloads sent from the client to the server contain a CID in this example as well.

## **8. Privacy Considerations**

The CID replaces the previously used 5-tuple and, as such, introduces an identifier that remains persistent during the lifetime of a DTLS connection. Every identifier introduces the risk of linkability, as explained in [[RFC6973](#)].

An on-path adversary observing the DTLS protocol exchanges between the DTLS client and the DTLS server is able to link the observed payloads to all subsequent payloads carrying the same ID pair (for bi-directional communication). Without multi-homing or mobility, the use of the CID exposes the same information as the 5-tuple.

With multi-homing, a passive attacker is able to correlate the communication interaction over the two paths and the sequence number makes it possible to correlate packets across CID changes. The lack of a CID update mechanism in DTLS 1.2 makes this extension unsuitable for mobility scenarios where correlation must be considered. Deployments that use DTLS in multi-homing environments and are concerned about this aspects SHOULD refuse to use CIDs in DTLS 1.2 and switch to DTLS 1.3 where a CID update mechanism is provided and sequence number encryption is available.

The specification introduces record padding for the CID-enhanced record layer, which is a privacy feature not available with the original DTLS 1.2 specification. Padding allows to inflate the size of the ciphertext making traffic analysis more difficult. More details about record padding can be found in [Section 5.4](#) and [Appendix E.3 of RFC 8446](#).

Finally, endpoints can use the CID to attach arbitrary metadata to each record they receive. This may be used as a mechanism to communicate per-connection information to on-path observers. There is no straightforward way to address this concern with CIDs that contain arbitrary values. Implementations concerned about this aspects SHOULD refuse to use CIDs.

## **9. Security Considerations**

An on-path adversary can create reflection attacks against third parties because a DTLS peer has no means to distinguish a genuine address update event (for example, due to a NAT rebinding) from one



that is malicious. This attack is of concern when there is a large asymmetry of request/response message sizes.

Additionally, an attacker able to observe the data traffic exchanged between two DTLS peers is able to replay datagrams with modified IP address/port numbers.

The topic of peer address updates is discussed in [Section 6](#).

## 10. IANA Considerations

IANA is requested to allocate an entry to the existing TLS "ExtensionType Values" registry, defined in [\[RFC5246\]](#), for connection\_id(TBD1) as described in the table below. IANA is requested to add an extra column to the TLS ExtensionType Values registry to indicate whether an extension is only applicable to DTLS.

Value	Extension Name	TLS 1.3	DTLS Only	Recommended	Reference
TBD1	connection_id	-	Y	N	[[This doc]]

Note: The value "N" in the Recommended column is set because this extension is intended only for specific use cases. This document describes an extension for DTLS 1.2 only; it is not to TLS (1.3). The DTLS 1.3 functionality is described in [\[I-D.ietf-tls-dtls13\]](#).

IANA is requested to allocate tls12\_cid(TBD2) in the "TLS ContentType Registry". The tls12\_cid ContentType is only applicable to DTLS 1.2.

## 11. References

### 11.1. Normative References

- [I-D.tschofenig-tls-dtls-rrc]  
 Fossati, T. and H. Tschofenig, "Return Routability Check for DTLS 1.2 and DTLS 1.3", [draft-tschofenig-tls-dtls-rrc-00](#) (work in progress), July 2019.
- [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>>.
- [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", [RFC 5246](#), DOI 10.17487/RFC5246, August 2008, <<https://www.rfc-editor.org/info/rfc5246>>.



- [RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", [RFC 6347](#), DOI 10.17487/RFC6347, January 2012, <<https://www.rfc-editor.org/info/rfc6347>>.
- [RFC7366] Gutmann, P., "Encrypt-then-MAC for Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", [RFC 7366](#), DOI 10.17487/RFC7366, September 2014, <<https://www.rfc-editor.org/info/rfc7366>>.
- [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>>.

### **11.2. Informative References**

- [I-D.ietf-tls-dtls13]  
Rescorla, E., Tschofenig, H., and N. Modadugu, "The Datagram Transport Layer Security (DTLS) Protocol Version 1.3", [draft-ietf-tls-dtls13-33](#) (work in progress), October 2019.
- [RFC6973] Cooper, A., Tschofenig, H., Aboba, B., Peterson, J., Morris, J., Hansen, M., and R. Smith, "Privacy Considerations for Internet Protocols", [RFC 6973](#), DOI 10.17487/RFC6973, July 2013, <<https://www.rfc-editor.org/info/rfc6973>>.

### **11.3. URIs**

- [1] <mailto:tls@ietf.org>
- [2] <https://www1.ietf.org/mailman/listinfo/tls>
- [3] <https://www.ietf.org/mail-archive/web/tls/current/index.html>





## [Appendix A](#). History

RFC EDITOR: PLEASE REMOVE THE THIS SECTION

### [draft-ietf-tls-dtls-connection-id-07](#)

- Wording changes in the security and privacy consideration and the peer address update sections.

### [draft-ietf-tls-dtls-connection-id-06](#)

- Updated IANA considerations
- Enhanced security consideration section to describe a potential man-in-the-middle attack concerning address validation.

### [draft-ietf-tls-dtls-connection-id-05](#)

- Restructured [Section 5](#) "Record Payload Protection"

### [draft-ietf-tls-dtls-connection-id-04](#)

- Editorial simplifications to the 'Record Layer Extensions' and the 'Record Payload Protection' sections.
- Added MAC calculations for block ciphers with and without Encrypt-then-MAC processing.

### [draft-ietf-tls-dtls-connection-id-03](#)

- Updated list of contributors
- Updated list of contributors and acknowledgements
- Updated example
- Changed record layer design
- Changed record payload protection
- Updated introduction and security consideration section
- Author- and affiliation changes

### [draft-ietf-tls-dtls-connection-id-02](#)

- Move to internal content types a la DTLS 1.3.



[draft-ietf-tls-dtls-connection-id-01](#)

- Remove 1.3 based on the WG consensus at IETF 101

[draft-ietf-tls-dtls-connection-id-00](#)

- Initial working group version (containing a solution for DTLS 1.2 and 1.3)

[draft-rescorla-tls-dtls-connection-id-00](#)

- Initial version

**[Appendix B. Working Group Information](#)**

RFC EDITOR: PLEASE REMOVE THE THIS SECTION

The discussion list for the IETF TLS working group is located at the e-mail address [tls@ietf.org](mailto:tls@ietf.org) [1]. Information on the group and information on how to subscribe to the list is at <https://www1.ietf.org/mailman/listinfo/tls> [2]

Archives of the list can be found at: <https://www.ietf.org/mail-archive/web/tls/current/index.html> [3]

**[Appendix C. Contributors](#)**

Many people have contributed to this specification and we would like to thank the following individuals for their contributions:

- \* Yin Xinxing  
Huawei  
[yinxinxing@huawei.com](mailto:yinxinxing@huawei.com)
- \* Nikos Mavrogiannopoulos  
RedHat  
[nmav@redhat.com](mailto:nmav@redhat.com)
- \* Tobias Gondrom  
[tobias.gondrom@gondrom.org](mailto:tobias.gondrom@gondrom.org)

Additionally, we would like to thank the Connection ID task force team members:

- Martin Thomson (Mozilla)
- Christian Huitema (Private Octopus Inc.)



- Jana Iyengar (Google)
- Daniel Kahn Gillmor (ACLU)
- Patrick McManus (Mozilla)
- Ian Swett (Google)
- Mark Nottingham (Fastly)

The task force team discussed various design ideas, including cryptographically generated session ids using hash chains and public key encryption, but dismissed them due to their inefficiency. The approach described in this specification is the simplest possible design that works given the limitations of DTLS 1.2. DTLS 1.3 provides better privacy features and developers are encouraged to switch to the new version of DTLS.

Finally, we want to thank the IETF TLS working group chairs, Chris Wood, Joseph Salowey, and Sean Turner, for their patience, support and feedback.

#### [Appendix D](#). Acknowledgements

We would like to thank Achim Kraus for his review comments and implementation feedback.

#### Authors' Addresses

Eric Rescorla (editor)  
RTFM, Inc.

E-Mail: [ekr@rtfm.com](mailto:ekr@rtfm.com)

Hannes Tschofenig (editor)  
Arm Limited

E-Mail: [hannes.tschofenig@arm.com](mailto:hannes.tschofenig@arm.com)

Thomas Fossati  
Arm Limited

E-Mail: [thomas.fossati@arm.com](mailto:thomas.fossati@arm.com)

