ACE Working Group
Internet-Draft
Intended status: Standards Track
Expires: July 25, 2018

M. Jones
Microsoft
E. Wahlstroem

S. Erdtman
Spotify AB
H. Tschofenig
ARM Ltd.

January 21, 2018

# CBOR Web Token (CWT)
# draft-ietf-ace-cbor-web-token-11

## Abstract

CBOR Web Token (CWT) is a compact means of representing claims to be transferred between two parties. The claims in a CWT are encoded in the Concise Binary Object Representation (CBOR) and CBOR Object Signing and Encryption (COSE) is used for added application layer security protection. A claim is a piece of information asserted about a subject and is represented as a name/value pair consisting of a claim name and a claim value. CWT is derived from JSON Web Token (JWT) but uses CBOR rather than JSON.

## 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 July 25, 2018.

## Copyright Notice

Copyright (c) 2018 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.

Table of Contents

1. Introduction . . . . . 3
1.1. CBOR Related Terminology . . . . . 3
2. Terminology . . . . . 3
3. Claims . . . . . 4
3.1. Registered Claims . . . . . 5
3.1.1. iss (Issuer) Claim . . . . . 5
3.1.2. sub (Subject) Claim . . . . . 5
3.1.3. aud (Audience) Claim . . . . . 5
3.1.4. exp (Expiration Time) Claim . . . . . 5
3.1.5. nbf (Not Before) Claim . . . . . 5
3.1.6. iat (Issued At) Claim . . . . . 6
3.1.7. cti (CWT ID) Claim . . . . . 6
4. Summary of the claim names, keys, and value types . . . . . 6
5. CBOR Tags and Claim Values . . . . . 6
6. CWT CBOR Tag . . . . . 6
7. Creating and Validating CWTs . . . . . 7
7.1. Creating a CWT . . . . . 7
7.2. Validating a CWT . . . . . 8
8. Security Considerations . . . . . 9
9. IANA Considerations . . . . . 10
9.1. CBOR Web Token (CWT) Claims Registry . . . . . 10
9.1.1. Registration Template . . . . . 10
9.1.2. Initial Registry Contents . . . . . 11
9.2. Media Type Registration . . . . . 13
9.2.1. Registry Contents . . . . . 13
9.3. CoAP Content-Formats Registration . . . . . 13
9.3.1. Registry Contents . . . . . 14
9.4. CBOR Tag registration . . . . . 14
9.4.1. Registry Contents . . . . . 14
10. References . . . . . 14
10.1. Normative References . . . . . 14
10.2. Informative References . . . . . 15
Appendix A. Examples . . . . . 15
A.1. Example CWT Claims Set . . . . . 16
A.2. Example keys . . . . . 16
A.2.1. 128-bit Symmetric Key . . . . . 16

A.2.2. 256-bit Symmetric Key . . . . . 17
A.2.3. ECDSA P-256 256-bit COSE Key . . . . . 17
A.3. Example Signed CWT . . . . . 17
A.4. Example MACed CWT . . . . . 18
A.5. Example Encrypted CWT . . . . . 19
A.6. Example Nested CWT . . . . . 20
A.7. Example MACed CWT with a floating-point value . . . . . 21
Appendix B. Acknowledgements . . . . . 22
Appendix C. Document History . . . . . 22
Authors' Addresses . . . . . 24

## 1. Introduction

The JSON Web Token (JWT) [RFC7519] is a standardized security token format that has found use in OAuth 2.0 and OpenID Connect deployments, among other applications. JWT uses JSON Web Signature (JWS) [RFC7515] and JSON Web Encryption (JWE) [RFC7516] to secure the contents of the JWT, which is a set of claims represented in JSON. The use of JSON for encoding information is popular for Web and native applications, but it is considered inefficient for some Internet of Things (IoT) systems that use low power radio technologies.

An alternative encoding of claims is defined in this document. Instead of using JSON, as provided by JWTs, this specification uses CBOR [RFC7049] and calls this new structure "CBOR Web Token (CWT)", which is a compact means of representing secured claims to be transferred between two parties. CWT is closely related to JWT. It references the JWT claims and both its name and pronunciation are derived from JWT. To protect the claims contained in CWTs, the CBOR Object Signing and Encryption (COSE) [RFC8152] specification is used.

The suggested pronunciation of CWT is the same as the English word "cot".

### 1.1. CBOR Related Terminology

In JSON, maps are called objects and only have one kind of map key: a string. CBOR uses strings, negative integers, and unsigned integers as map keys. The integers are used for compactness of encoding and easy comparison. The inclusion of strings allows for an additional range of short encoded values to be used.

## 2. 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 BCP

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

This document reuses terminology from JWT [RFC7519] and COSE [RFC8152].

StringOrURI
: The "StringOrURI" term has the same meaning, syntax, and processing rules as the "StringOrURI" term defined in Section 2 of [RFC7519], except that it uses a CBOR text string instead of a JSON string value.

NumericDate
: The "NumericDate" term has the same meaning, syntax, and processing rules as the "NumericDate" term defined in Section 2 of [RFC7519], except that the CBOR numeric date representation (from Section 2.4.1 of [RFC7049]) is used. The encoding is modified so that the leading tag 1 (epoch-based date/time) MUST be omitted.

Claim Name
: The human-readable name used to identify a claim.

Claim Key
: The CBOR map key used to identify a claim.

Claim Value
: The CBOR map value representing the value of the claim.

CWT Claims Set
: The CBOR map that contains the claims conveyed by the CWT.

## 3. Claims

The set of claims that a CWT must contain to be considered valid is context dependent and is outside the scope of this specification. Specific applications of CWTs will require implementations to understand and process some claims in particular ways. However, in the absence of such requirements, all claims that are not understood by implementations MUST be ignored.

To keep CWTs as small as possible, the Claim Keys are represented using integers or text strings. Section 4 summarizes all keys used to identify the claims defined in this document.

### 3.1. Registered Claims

None of the claims defined below are intended to be mandatory to use or implement. They rather provide a starting point for a set of useful, interoperable claims. Applications using CWTs should define which specific claims they use and when they are required or optional.

#### 3.1.1. iss (Issuer) Claim

The "iss" (issuer) claim has the same meaning, syntax, and processing rules as the "iss" claim defined in Section 4.1.1 of [RFC7519], except that the value is of type StringOrURI. The Claim Key 1 is used to identify this claim.

#### 3.1.2. sub (Subject) Claim

The "sub" (subject) claim has the same meaning, syntax, and processing rules as the "sub" claim defined in Section 4.1.2 of [RFC7519], except that the value is of type StringOrURI. The Claim Key 2 is used to identify this claim.

#### 3.1.3. aud (Audience) Claim

The "aud" (audience) claim has the same meaning, syntax, and processing rules as the "aud" claim defined in Section 4.1.3 of [RFC7519], except that the value of the audience claim is of type StringOrURI when it is not an array or the values of the audience array elements are of type StringOrURI when the audience claim value is an array. The Claim Key 3 is used to identify this claim.

#### 3.1.4. exp (Expiration Time) Claim

The "exp" (expiration time) claim has the same meaning, syntax, and processing rules as the "exp" claim defined in Section 4.1.4 of [RFC7519], except that the value is of type NumericDate. The Claim Key 4 is used to identify this claim.

#### 3.1.5. nbf (Not Before) Claim

The "nbf" (not before) claim has the same meaning, syntax, and processing rules as the "nbf" claim defined in Section 4.1.5 of [RFC7519], except that the value is of type NumericDate. The Claim Key 5 is used to identify this claim.

### 3.1.6. iat (Issued At) Claim

The "iat" (issued at) claim has the same meaning, syntax, and processing rules as the "iat" claim defined in Section 4.1.6 of [RFC7519], except that the value is of type NumericDate. The Claim Key 6 is used to identify this claim.

### 3.1.7. cti (CWT ID) Claim

The "cti" (CWT ID) claim has the same meaning, syntax, and processing rules as the "jti" claim defined in Section 4.1.7 of [RFC7519], except that the value is of type byte string. The Claim Key 7 is used to identify this claim.

## 4. Summary of the claim names, keys, and value types

| Name | Key | Value type |
|---|---|---|
| iss | 1 | text string |
| sub | 2 | text string |
| aud | 3 | text string |
| exp | 4 | integer or floating-point number |
| nbf | 5 | integer or floating-point number |
| iat | 6 | integer or floating-point number |
| cti | 7 | byte string |

Table 1: Summary of the claim names, keys, and value types

## 5. CBOR Tags and Claim Values

The claim values defined in this specification MUST NOT be prefixed with any CBOR tag. For instance, while CBOR tag 1 (epoch-based date/time) could logically be prefixed to values of the "exp", "nbf", and "iat" claims, this is unnecessary, since the representation of the claim values is already specified by the claim definitions. Tagging claim values would only take up extra space without adding information. However, this does not prohibit future claim definitions from requiring the use of CBOR tags for those specific claims.

## 6. CWT CBOR Tag

How to determine that a CBOR data structure is a CWT is application-dependent. In some cases, this information is known from the application context, such as from the position of the CWT in a data structure at which the value must be a CWT. One method of indicating

that a CBOR object is a CWT is the use of the "application/cwt" content type by a transport protocol.

This section defines the CWT CBOR tag as another means for applications to declare that a CBOR data structure is a CWT. Its use is optional and is intended for use in cases in which this information would not otherwise be known.

If present, the CWT tag MUST prefix a tagged object using one of the COSE CBOR tags. In this example, the COSE_Mac0 tag is used. The actual COSE_Mac0 object has been excluded from this example.

```
/ CWT CBOR tag / 61(
  / COSE_Mac0 CBOR tag / 17(
    / COSE_Mac0 object /
  )
)
```

Figure 1: Example of a CWT tag usage

## 7. Creating and Validating CWTs

### 7.1. Creating a CWT

To create a CWT, the following steps are performed. The order of the steps is not significant in cases where there are no dependencies between the inputs and outputs of the steps.

1. Create a CWT Claims Set containing the desired claims.

2. Let the Message be the binary representation of the CWT Claims Set.

3. Create a COSE Header containing the desired set of Header Parameters. The COSE Header MUST be valid per the [RFC8152] specification.

4. Depending upon whether the CWT is signed, MACed, or encrypted, there are three cases:

   * If the CWT is signed, create a COSE_Sign/COSE_Sign1 object using the Message as the COSE_Sign/COSE_Sign1 Payload; all steps specified in [RFC8152] for creating a COSE_Sign/ COSE_Sign1 object MUST be followed.

   * Else, if the CWT is MACed, create a COSE_Mac/COSE_Mac0 object using the Message as the COSE_Mac/COSE_Mac0 Payload; all steps

specified in [RFC8152] for creating a COSE_Mac/COSE_Mac0 object MUST be followed.

* Else, if the CWT is a COSE_Encrypt/COSE_Encrypt0 object, create a COSE_Encrypt/COSE_Encrypt0 using the Message as the plaintext for the COSE_Encrypt/COSE_Encrypt0 object; all steps specified in [RFC8152] for creating a COSE_Encrypt/ COSE_Encrypt0 object MUST be followed.

5. If a nested signing, MACing, or encryption operation will be performed, let the Message be the tagged COSE_Sign/COSE_Sign1, COSE_Mac/COSE_Mac0, or COSE_Encrypt/COSE_Encrypt0, and return to Step 3.

6. If needed by the application, prepend the COSE object with the appropriate COSE CBOR tag to indicate the type of the COSE object. If needed by the application, prepend the COSE object with the CWT CBOR tag to indicate that the COSE object is a CWT.

### 7.2. Validating a CWT

When validating a CWT, the following steps are performed. The order of the steps is not significant in cases where there are no dependencies between the inputs and outputs of the steps. If any of the listed steps fail, then the CWT MUST be rejected -- that is, treated by the application as invalid input.

1. Verify that the CWT is a valid CBOR object.

2. If the object begins with the CWT CBOR tag, remove it and verify that one of the COSE CBOR tags follows it.

3. If the object is tagged with one of the COSE CBOR tags, remove it and use it to determine the type of the CWT, COSE_Sign/ COSE_Sign1, COSE_Mac/COSE_Mac0, or COSE_Encrypt/COSE_Encrypt0. If the object does not have a COSE CBOR tag, the COSE message type is determined from the application context.

4. Verify that the resulting COSE Header includes only parameters and values whose syntax and semantics are both understood and supported or that are specified as being ignored when not understood.

5. Depending upon whether the CWT is a signed, MACed, or encrypted, there are three cases:

   * If the CWT is a COSE_Sign/COSE_Sign1, follow the steps specified in [RFC8152] Section 4 (Signing Objects) for

validating a COSE_Sign/COSE_Sign1 object. Let the Message be the COSE_Sign/COSE_Sign1 payload.

* Else, if the CWT is a COSE_Mac/COSE_Mac0, follow the steps specified in [RFC8152] Section 6 (MAC Objects) for validating a COSE_Mac/COSE_Mac0 object. Let the Message be the COSE_Mac/COSE_Mac0 payload.

* Else, if the CWT is a COSE_Encrypt/COSE_Encrypt0 object, follow the steps specified in [RFC8152] Section 5 (Encryption Objects) for validating a COSE_Encrypt/COSE_Encrypt0 object. Let the Message be the resulting plaintext.

6. If the Message begins with a COSE CBOR tag, then the Message is a CWT that was the subject of nested signing, MACing, or encryption operations. In this case, return to Step 1, using the Message as the CWT.

7. Verify that the Message is a valid CBOR map; let the CWT Claims Set be this CBOR map.

## 8. Security Considerations

The security of the CWT relies upon on the protections offered by COSE. Unless the claims in a CWT are protected, an adversary can modify, add, or remove claims.

Since the claims conveyed in a CWT may be used to make authorization decisions, it is not only important to protect the CWT in transit but also to ensure that the recipient can authenticate the party that assembled the claims and created the CWT. Without trust of the recipient in the party that created the CWT, no sensible authorization decision can be made. Furthermore, the creator of the CWT needs to carefully evaluate each claim value prior to including it in the CWT so that the recipient can be assured of the validity of the information provided.

While syntactically, the signing and encryption operations for Nested CWTs may be applied in any order, if both signing and encryption are necessary, normally producers should sign the message and then encrypt the result (thus encrypting the signature). This prevents attacks in which the signature is stripped, leaving just an encrypted message, as well as providing privacy for the signer. Furthermore, signatures over encrypted text are not considered valid in many jurisdictions.

# 9. IANA Considerations

## 9.1. CBOR Web Token (CWT) Claims Registry

This section establishes the IANA "CBOR Web Token (CWT) Claims" registry.

Values are registered on a Specification Required [RFC5226] basis after a three-week review period on the cwt-reg-review@ietf.org mailing list, on the advice of one or more Designated Experts. However, to allow for the allocation of values prior to publication, the Designated Experts may approve registration once they are satisfied that such a specification will be published. [[ Note to the RFC Editor: The name of the mailing list should be determined in consultation with the IESG and IANA. Suggested name: cwt-reg-review@ietf.org. ]]

Registration requests sent to the mailing list for review should use an appropriate subject (e.g., "Request to register claim: example"). Registration requests that are undetermined for a period longer than 21 days can be brought to the IESG's attention (using the iesg@ietf.org mailing list) for resolution.

Criteria that should be applied by the Designated Experts includes determining whether the proposed registration duplicates existing functionality, whether it is likely to be of general applicability or whether it is useful only for a single application, and whether the registration description is clear.

It is suggested that multiple Designated Experts be appointed who are able to represent the perspectives of different applications using this specification in order to enable broadly informed review of registration decisions. In cases where a registration decision could be perceived as creating a conflict of interest for a particular Expert, that Expert should defer to the judgment of the other Experts.

### 9.1.1. Registration Template

Claim Name:
: The human-readable name requested (e.g., "iss").

Claim Description:
: Brief description of the claim (e.g., "Issuer").

JWT Claim Name:
: Claim Name of the equivalent JWT claim, as registered in [IANA.JWT.Claims]. CWT claims should normally have a

corresponding JWT claim. If a corresponding JWT claim would not make sense, the Designated Experts can choose to accept registrations for which the JWT Claim Name is listed as "N/A".

Claim Key:
CBOR map key for the claim. Integer values between -256 and 255 and strings of length 1 are designated as Standards Track Document required. Integer values from -65536 to 65535 and strings of length 2 are designated as Specification Required. Integer values of greater than 65535 and strings of length greater than 2 are designated as expert review. Integer values less than -65536 are marked as private use.

Claim Value Type(s):
CBOR types that can be used for the claim value.

Change Controller:
For Standards Track RFCs, list the "IESG". For others, give the name of the responsible party. Other details (e.g., postal address, email address, home page URI) may also be included.

Specification Document(s):
Reference to the document or documents that specify the parameter, preferably including URIs that can be used to retrieve copies of the documents. An indication of the relevant sections may also be included but is not required.

### 9.1.2. Initial Registry Contents

- Claim Name: (RESERVED)
- Claim Description: This registration reserves the key value 0.
- JWT Claim Name: N/A
- Claim Key: 0
- Claim Value Type(s): N/A
- Change Controller: IESG
- Specification Document(s): [[ this specification ]]

- Claim Name: "iss"
- Claim Description: Issuer
- JWT Claim Name: "iss"
- Claim Key: 1
- Claim Value Type(s): text string
- Change Controller: IESG
- Specification Document(s): Section 3.1.1 of [[ this specification ]]

- Claim Name: "sub"
- Claim Description: Subject

- JWT Claim Name: "sub"
- Claim Key: 2
- Claim Value Type(s): text string
- Change Controller: IESG
- Specification Document(s): Section 3.1.2 of [[ this specification ]]

- Claim Name: "aud"
- Claim Description: Audience
- JWT Claim Name: "aud"
- Claim Key: 3
- Claim Value Type(s): text string
- Change Controller: IESG
- Specification Document(s): Section 3.1.3 of [[ this specification ]]

- Claim Name: "exp"
- Claim Description: Expiration Time
- JWT Claim Name: "exp"
- Claim Key: 4
- Claim Value Type(s): integer or floating-point number
- Change Controller: IESG
- Specification Document(s): Section 3.1.4 of [[ this specification ]]

- Claim Name: "nbf"
- Claim Description: Not Before
- JWT Claim Name: "nbf"
- Claim Key: 5
- Claim Value Type(s): integer or floating-point number
- Change Controller: IESG
- Specification Document(s): Section 3.1.5 of [[ this specification ]]

- Claim Name: "iat"
- Claim Description: Issued At
- JWT Claim Name: "iat"
- Claim Key: 6
- Claim Value Type(s): integer or floating-point number
- Change Controller: IESG
- Specification Document(s): Section 3.1.6 of [[ this specification ]]

- Claim Name: "cti"
- Claim Description: CWT ID
- JWT Claim Name: "jti"
- Claim Key: 7
- Claim Value Type(s): byte string

- Change Controller: IESG
- Specification Document(s): Section 3.1.7 of [[ this specification ]]

### 9.2. Media Type Registration

This section registers the "application/cwt" media type in the "Media Types" registry [IANA.MediaTypes] in the manner described in RFC 6838 [RFC6838], which can be used to indicate that the content is a CWT.

#### 9.2.1. Registry Contents

- Type name: application
- Subtype name: cwt
- Required parameters: N/A
- Optional parameters: N/A
- Encoding considerations: binary
- Security considerations: See the Security Considerations section of [[ this specification ]]
- Interoperability considerations: N/A
- Published specification: [[ this specification ]]
- Applications that use this media type: IoT applications sending security tokens over HTTP(S), CoAP(S), and other transports.
- Fragment identifier considerations: N/A
- Additional information:

  Magic number(s): N/A
  File extension(s): N/A
  Macintosh file type code(s): N/A

- Person & email address to contact for further information: IESG, iesg@ietf.org
- Intended usage: COMMON
- Restrictions on usage: none
- Author: Michael B. Jones, mbj@microsoft.com
- Change controller: IESG
- Provisional registration? No

### 9.3. CoAP Content-Formats Registration

This section registers the CoAP Content-Format ID for the "application/cwt" media type in the "CoAP Content-Formats" registry [IANA.CoAP.Content-Formats].

#### 9.3.1. Registry Contents

- Media Type: application/cwt
- Encoding: -
- Id: TBD (maybe 61)
- Reference: [[ this specification ]]

### 9.4. CBOR Tag registration

This section registers the CWT CBOR tag in the "CBOR Tags" registry [IANA.CBOR.Tags].

#### 9.4.1. Registry Contents

- CBOR Tag: TBD (maybe 61 to use the same value as the Content-Format)
- Data Item: CBOR Web Token (CWT)
- Semantics: CBOR Web Token (CWT), as defined in [[ this specification ]]
- Reference: [[ this specification ]]
- Point of Contact: Michael B. Jones, mbj@microsoft.com

## 10. References

### 10.1. Normative References

[IANA.CBOR.Tags]
IANA, "Concise Binary Object Representation (CBOR) Tags", <http://www.iana.org/assignments/cbor-tags/cbor-tags.xhtml>.

[IANA.CoAP.Content-Formats]
IANA, "CoAP Content-Formats", <http://www.iana.org/assignments/core-parameters/core-parameters.xhtml#content-formats>.

[IANA.MediaTypes]
IANA, "Media Types", <http://www.iana.org/assignments/media-types>.

[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>.

[RFC7049] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", RFC 7049, DOI 10.17487/RFC7049, October 2013, <https://www.rfc-editor.org/info/rfc7049>.

[RFC7519] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Token (JWT)", RFC 7519, DOI 10.17487/RFC7519, May 2015, <https://www.rfc-editor.org/info/rfc7519>.

[RFC8152] Schaad, J., "CBOR Object Signing and Encryption (COSE)", RFC 8152, DOI 10.17487/RFC8152, July 2017, <https://www.rfc-editor.org/info/rfc8152>.

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

### 10.2. Informative References

[IANA.JWT.Claims] IANA, "JSON Web Token Claims", <http://www.iana.org/assignments/jwt>.

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

[RFC6838] Freed, N., Klensin, J., and T. Hansen, "Media Type Specifications and Registration Procedures", BCP 13, RFC 6838, DOI 10.17487/RFC6838, January 2013, <https://www.rfc-editor.org/info/rfc6838>.

[RFC7515] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Signature (JWS)", RFC 7515, DOI 10.17487/RFC7515, May 2015, <https://www.rfc-editor.org/info/rfc7515>.

[RFC7516] Jones, M. and J. Hildebrand, "JSON Web Encryption (JWE)", RFC 7516, DOI 10.17487/RFC7516, May 2015, <https://www.rfc-editor.org/info/rfc7516>.

## Appendix A. Examples

This appendix includes a set of CWT examples that show how the CWT Claims Set can be protected. There are examples that are signed, MACed, encrypted, and that use nested signing and encryption. To make the examples easier to read, they are presented both as hex strings and in the extended CBOR diagnostic notation described in Section 6 of [RFC7049].

Where a byte string is to carry an embedded CBOR-encoded item, the diagnostic notation for this CBOR data item can be enclosed in '<<'

and '>>' to notate the byte string resulting from encoding the data item, e.g., h'63666F6F' translates to <<"foo">>.

## A.1. Example CWT Claims Set

The CWT Claims Set used for the different examples displays usage of all the defined claims. For signed and MACed examples, the CWT Claims Set is the CBOR encoding as a byte string.

```
a70175636f61703a2f2f61732e6578616d706c652e636f6d02656572696b7703
7818636f61703a2f2f6c696768742e6578616d706c652e636f6d041a5612aeb0
051a5610d9f0061a5610d9f007420b71
```

Figure 2: Example CWT Claims Set as hex string

```
{
  / iss / 1: "coap://as.example.com",
  / sub / 2: "erikw",
  / aud / 3: "coap://light.example.com",
  / exp / 4: 1444064944,
  / nbf / 5: 1443944944,
  / iat / 6: 1443944944,
  / cti / 7: h'0b71'
}
```

Figure 3: Example CWT Claims Set in CBOR diagnostic notation

## A.2. Example keys

This section contains the keys used to sign, MAC, and encrypt the messages in this appendix. Line breaks are for display purposes only.

### A.2.1. 128-bit Symmetric Key

```
a42050231f4c4d4d3051fdc2ec0a3851d5b3830104024c53796d6d6574726963
313238030a
```

Figure 4: 128-bit symmetric COSE_Key as hex string

```
{
  / k /   -1: h'231f4c4d4d3051fdc2ec0a3851d5b383'
  / kty /  1: 4 / Symmetric /,
  / kid /  2: h'53796d6d6574726963313238' / 'Symmetric128' /,
  / alg /  3: 10 / AES-CCM-16-64-128 /
}
```

Figure 5: 128-bit symmetric COSE_Key in CBOR diagnostic notation

### A.2.2. 256-bit Symmetric Key

```
a4205820403697de87af64611c1d32a05dab0fe1fcb715a86ab435f1ec99192d
795693880104024c53796d6d6574726963323536030a
```

Figure 6: 256-bit symmetric COSE_Key as hex string

```
{
  / k /   -1: h'403697de87af64611c1d32a05dab0fe1fcb715a86ab435f1
               ec99192d79569388'
  / kty /  1: 4 / Symmetric /,
  / kid /  4: h'53796d6d6574726963323536' / 'Symmetric256' /,
  / alg /  3: 4 / HMAC 256/64 /
}
```

Figure 7: 256-bit symmetric COSE_Key in CBOR diagnostic notation

### A.2.3. ECDSA P-256 256-bit COSE Key

```
a72358206c1382765aec5358f117733d281c1c7bdc39884d04a45a1e6c67c858
bc206c1922582060f7f1a780d8a783bfb7a2dd6b2796e8128dbbcef9d3d168db
9529971a36e7b9215820143329cce7868e416927599cf65a34f3ce2ffda55a7e
ca69ed8919a394d42f0f200101020252417379 6d6d65747269634543445341 32
35360326
```

Figure 8: ECDSA 256-bit COSE Key as hex string

```
{
  / d /   -4: h'6c1382765aec5358f117733d281c1c7bdc39884d04a45a1e
               6c67c858bc206c19',
  / y /   -3: h'60f7f1a780d8a783bfb7a2dd6b2796e8128dbbcef9d3d168
               db9529971a36e7b9',
  / x /   -2: h'143329cce7868e416927599cf65a34f3ce2ffda55a7eca69
               ed8919a394d42f0f',
  / crv / -1: 1 / P-256 /,
  / kty /  1: 2 / EC2 /,
  / kid /  2: h'4173796d6d6574726963454344534132
               23536' / 'AsymmetricECDSA256' /,
  / alg /  3: -7 / ECDSA 256 /
}
```

Figure 9: ECDSA 256-bit COSE Key in CBOR diagnostic notation

## A.3. Example Signed CWT

This section shows a signed CWT with a single recipient and a full CWT Claims Set.

The signature is generated using the private key listed in Appendix A.2.3 and it can be validated using the public key from Appendix A.2.3. Line breaks are for display purposes only.

```
d28443a10126a104524173796d6d6574726963454344534132353658 50a701756
36f61703a2f2f61732e6578616d706c652e636f6d02656572696 b77037818636f
61703a2f2f6c696768742e6578616d706c652e636f6d041a5612aeb0051a5610d
9f0061a5610d9f007420b7158405427c1ff28d23fbad1f29c4c7c6a555e601d6f
a29f9179bc3d7438bacaca5acd08c8d4d4f96131680c429a01f85951ecee743a5
2b9b63632c57209120e1c9e30
```

Figure 10: Signed CWT as hex string

```
18(
  [
    / protected / << {
      / alg / 1: -7 / ECDSA 256 /
    } >>,
    / unprotected / {
      / kid / 4: h'4173796d6d6574726963454344534132
                   3536' / 'AsymmetricECDSA256' /
    },
    / payload / << {
      / iss / 1: "coap://as.example.com",
      / sub / 2: "erikw",
      / aud / 3: "coap://light.example.com",
      / exp / 4: 1444064944,
      / nbf / 5: 1443944944,
      / iat / 6: 1443944944,
      / cti / 7: h'0b71'
    } >>,
    / signature / h'5427c1ff28d23fbad1f29c4c7c6a555e601d6fa29f
                    9179bc3d7438bacaca5acd08c8d4d4f96131680c42
                    9a01f85951ecee743a52b9b63632c57209120e1c9e
                    30'
  ]
)
```

Figure 11: Signed CWT in CBOR diagnostic notation

## A.4. Example MACed CWT

This section shows a MACed CWT with a single recipient, a full CWT Claims Set, and a CWT tag.

The MAC is generated using the 256-bit symmetric key from Appendix A.2.2 with a 64-bit truncation. Line breaks are for display purposes only.

```
d83dd18443a10104a1044c53796d6d6574726963323536585 0a70175636f6170
3a2f2f61732e6578616d706c652e636f6d02656572696b77037818636f61703a
2f2f6c696768742e6578616d706c652e636f6d041a5612aeb0051a5610d9f006
1a5610d9f007420b7148093101ef6d789200
```

Figure 12: MACed CWT with CWT tag as hex string

```
61(
  17(
    [
      / protected / << {
        / alg / 1: 4 / HMAC-256-64 /
      } >>,
      / unprotected / {
        / kid / 4: h'53796d6d6574726963323536' / 'Symmetric256' /
      },
      / payload / << {
        / iss / 1: "coap://as.example.com",
        / sub / 2: "erikw",
        / aud / 3: "coap://light.example.com",
        / exp / 4: 1444064944,
        / nbf / 5: 1443944944,
        / iat / 6: 1443944944,
        / cti / 7: h'0b71'
      } >>,
      / tag / h'093101ef6d789200'
    ]
  )
)
```

Figure 13: MACed CWT with CWT tag in CBOR diagnostic notation

## A.5. Example Encrypted CWT

This section shows an encrypted CWT with a single recipient and a full CWT Claims Set.

The encryption is done with AES-CCM mode using the 128-bit symmetric key from Appendix A.2.1 with a 64-bit tag and 13-byte nonce, i.e., COSE AES-CCM-16-64-128. Line breaks are for display purposes only.

```
d08343a1010aa2044c53796d6d6574726963313238054d99a0d7846e762c49ff
e8a63e0b5858b918a11fd81e438b7f973d9e2e119bcb22424ba0f38a80f27562
f400ee1d0d6c0fdb559c02421fd384fc2ebe22d7071378b0ea7428fff157444d
45f7e6afcda1aae5f6495830c58627087fc5b4974f319a8707a635dd643b
```

Figure 14: Encrypted CWT as hex string

```
16(
  [
    / protected / << {
      / alg / 1: 10 / AES-CCM-16-64-128 /
    } >>,
    / unprotected / {
      / kid / 4: h'53796d6d6574726963313238' / 'Symmetric128' /,
      / iv /  5: h'99a0d7846e762c49ffe8a63e0b'
    },
    / ciphertext / h'b918a11fd81e438b7f973d9e2e119bcb22424ba0f38
                     a80f27562f400ee1d0d6c0fdb559c02421fd384fc2e
                     be22d7071378b0ea7428fff157444d45f7e6afcda1a
                     ae5f6495830c58627087fc5b4974f319a8707a635dd
                     643b'
  ]
)
```

Figure 15: Encrypted CWT in CBOR diagnostic notation

## A.6. Example Nested CWT

This section shows a Nested CWT, signed and then encrypted, with a single recipient and a full CWT Claims Set.

The signature is generated using the private ECDSA key from Appendix A.2.3 and it can be validated using the public ECDSA parts from Appendix A.2.3. The encryption is done with AES-CCM mode using the 128-bit symmetric key from Appendix A.2.1 with a 64-bit tag and 13-byte nonce, i.e., COSE AES-CCM-16-64-128. The content type is set to CWT to indicate that there are multiple layers of COSE protection before finding the CWT Claims Set. The decrypted ciphertext will be a COSE_sign1 structure. In this example, it is the same one as in Appendix A.3, i.e., a Signed CWT Claims Set. Note that there is no limitation to the number of layers; this is an example with two layers. Line breaks are for display purposes only.

```
d08343a1010aa2044c53796d6d6574726963313238054d4a0694c0e69ee6b595
6655c7b258b7f6b0914f993de822cc47e5e57a188d7960b528a747446fe12f0e
7de05650dec74724366763f167a29c002dfd15b34d8993391cf49bc91127f545
dba8703d66f5b7f1ae91237503d371e6333df9708d78c4fb8a8386c8ff09dc49
af768b23179deab78d96490a66d5724fb33900c60799d9872fac6da3bdb89043
d67c2a05414ce331b5b8f1ed8ff7138f45905db2c4d5bc8045ab372bff142631
610a7e0f677b7e9b0bc73adefdcee16d9d5d284c616abeab5d8c291ce0
```

Figure 16: Signed and Encrypted CWT as hex string

```
16(
  [
    / protected / << {
      / alg / 1: 10 / AES-CCM-16-64-128 /
    } >>,
    / unprotected / {
      / kid / 4: h'53796d6d6574726963313238' / 'Symmetric128' /,
      / iv /  5: h'4a0694c0e69ee6b5956655c7b2'
    },
    / ciphertext / h'f6b0914f993de822cc47e5e57a188d7960b528a7474
                     46fe12f0e7de05650dec74724366763f167a29c002d
                     fd15b34d8993391cf49bc91127f545dba8703d66f5b
                     7f1ae91237503d371e6333df9708d78c4fb8a8386c8
                     ff09dc49af768b23179deab78d96490a66d5724fb33
                     900c60799d9872fac6da3bdb89043d67c2a05414ce3
                     31b5b8f1ed8ff7138f45905db2c4d5bc8045ab372bf
                     f142631610a7e0f677b7e9b0bc73adefdcee16d9d5d
                     284c616abeab5d8c291ce0'
  ]
)
```

Figure 17: Signed and Encrypted CWT in CBOR diagnostic notation

## A.7. Example MACed CWT with a floating-point value

This section shows a MACed CWT with a single recipient and a simple CWT Claims Set. The CWT Claims Set with a floating-point 'iat' value.

The MAC is generated using the 256-bit symmetric key from Appendix A.2.2 with a 64-bit truncation. Line breaks are for display purposes only.

```
d18443a10104a1044c53796d6d65747269633235364ba106fb41d584367c2000
0048b8816f34c0542892
```

Figure 18: MACed CWT with a floating-point value as hex string

```
17(
  [
    / protected / << {
      / alg / 1: 4 / HMAC-256-64 /
    } >>,
    / unprotected / {
      / kid / 4: h'53796d6d6574726963323536' / 'Symmetric256' /,
    },
    / payload / << {
      / iat / 6: 1443944944.5
    } >>,
    / tag / h'b8816f34c0542892'
  ]
)
```

Figure 19: MACed CWT with a floating-point value in CBOR diagnostic notation

## Appendix B. Acknowledgements

This specification is based on JSON Web Token (JWT) [RFC7519], the authors of which also include Nat Sakimura and John Bradley. It also incorporates suggestions made by many people, including Carsten Bormann, Esko Dijk, Benjamin Kaduk, Jim Schaad, Ludwig Seitz, and Goeran Selander.

## Appendix C. Document History

[[ to be removed by the RFC Editor before publication as an RFC ]]

-11

- Corrected the "iv" value in the signed and encrypted CWT example.
- Mention CoAP in the "application/cwt" media type registration.
- Changed references of the form "Section 4.1.1 of JWT <xref target="RFC7519"/>" to "Section 4.1.1 of <xref target="RFC7519"/>" so that rfcmarkup will generate correct external section reference links.
- Updated Acknowledgements.

-10

- Clarified that the audience claim value can be a single audience value or an array of audience values, just as is the case for the JWT "aud" claim.

- Clarified the nested CWT description.

- Changed uses of "binary string" to "byte string".

-09

- Added key ID values to the examples.

- Key values for the examples are now represented in COSE_Key format using CBOR diagnostic notation.

-08

- Updated the diagnostic notation for embedded objects in the examples, addressing feedback by Carsten Bormann.

-07

- Updated examples for signing and encryption. Signatures are now deterministic as recommended by COSE specification.

-06

- Addressed review comments by Carsten Bormann and Jim Schaad. All changes were editorial in nature.

-05

- Addressed working group last call comments with the following changes:

- Say that CWT is derived from JWT, rather than CWT is a profile of JWT.

- Used CBOR type names in descriptions, rather than major/minor type numbers.

- Clarified the NumericDate and StringOrURI descriptions.

- Changed to allow CWT claim names to use values of any legal CBOR map key type.

- Changed to use the CWT tag to identify nested CWTs instead of the CWT content type.

- Added an example using a floating-point date value.

- Acknowledged reviewers.

-04

- Specified that the use of CBOR tags to prefix any of the claim values defined in this specification is NOT RECOMMENDED.

-03

- Reworked the examples to include signed, MACed, encrypted, and nested CWTs.
- Defined the CWT CBOR tag and explained its usage.

-02

- Added IANA registration for the application/cwt media type.
- Clarified the nested CWT language.
- Corrected nits identified by Ludwig Seitz.

-01

- Added IANA registration for CWT Claims.
- Added IANA registration for the application/cwt CoAP content-format type.
- Added Samuel Erdtman as an editor.
- Changed Erik's e-mail address.

-00

- Created the initial working group version based on draft-wahlstroem-ace-cbor-web-token-00.

## Authors' Addresses

Michael B. Jones
Microsoft

Email: mbj@microsoft.com
URI: http://self-issued.info/

Erik Wahlstroem
Sweden

Email: erik@wahlstromstekniska.se

Samuel Erdtman
Spotify AB
Birger Jarlsgatan 61, 4tr
Stockholm 113 56
Sweden

Phone: +46702691499
Email: erdtman@spotify.com

Hannes Tschofenig
ARM Ltd.
Hall in Tirol 6060
Austria

Email: Hannes.Tschofenig@arm.com