CBOR Web Token (CWT)

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 is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc8392.
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 .................................................. 4
  1.1. CBOR-Related Terminology .................................. 4
2. Terminology ................................................... 4
3. Claims .......................................................... 5
  3.1. Registered Claims .......................................... 5
    3.1.1. iss (Issuer) Claim ................................... 5
    3.1.2. sub (Subject) Claim .................................. 6
    3.1.3. aud (Audience) Claim .................................. 6
    3.1.4. exp (Expiration Time) Claim ........................... 6
    3.1.5. nbf (Not Before) Claim ................................ 6
    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 .................. 7
5. CBOR Tags and Claim Values ..................................... 7
6. CWT CBOR Tag .................................................. 7
7. Creating and Validating CWTs .................................... 8
  7.1. Creating a CWT .............................................. 8
  7.2. Validating a CWT ............................................ 9
8. Security Considerations .......................................... 10
9. IANA Considerations ............................................ 11
  9.1. CBOR Web Token (CWT) Claims Registry ..................... 11
    9.1.1. Registration Template ................................ 12
    9.1.2. Initial Registry Contents ............................ 12
  9.2. Media Type Registration .................................... 14
    9.2.1. Registry Contents .................................... 14
  9.3. CoAP Content-Formats Registration .......................... 14
    9.3.1. Registry Contents .................................... 15
  9.4. CBOR Tag registration ...................................... 15
    9.4.1. Registry Contents .................................... 15
10. References .................................................... 15
  10.1. Normative References ..................................... 15
  10.2. Informative References ................................... 16
Appendix A. Examples ............................................... 17
  A.1. Example CWT Claims Set .................................... 17
  A.2. Example Keys .............................................. 17
    A.2.1. 128-Bit Symmetric Key ................................ 18
    A.2.2. 256-Bit Symmetric Key ................................ 18
    A.2.3. Elliptic Curve Digital Signature Algorithm (ECDSA)
           P-256 256-Bit COSE Key ................................ 19
  A.3. Example Signed CWT ......................................... 19
  A.4. Example MACed CWT ......................................... 20
  A.5. Example Encrypted CWT ..................................... 21
  A.6. Example Nested CWT ......................................... 22
  A.7. Example MACed CWT with a Floating-Point Value ............. 23
Acknowledgements .................................................. 24
Authors’ Addresses ............................................... 25
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 (the suggested pronunciation of CWT is the same as the English word "cot"). To protect the claims contained in CWTs, the CBOR Object Signing and Encryption (COSE) [RFC8152] specification is used.

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 in this specification has the same meaning and processing rules as the JWT "StringOrURI" term defined in Section 2 of [RFC7519], except that it is represented as a CBOR text string instead of a JSON text string.
NumericDate
The "NumericDate" term in this specification has the same meaning and processing rules as the JWT "NumericDate" term defined in Section 2 of [RFC7519], except that it is represented as a CBOR numeric date (from Section 2.4.1 of [RFC7049]) instead of a JSON number. 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. Rather, they 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 and processing rules as the "iss" claim defined in Section 4.1.1 of [RFC7519], except that the value is a StringOrURI, as defined in Section 2 of this specification. The Claim Key 1 is used to identify this claim.
3.1.2.  sub (Subject) Claim

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

3.1.3.  aud (Audience) Claim

   The "aud" (audience) claim has the same meaning and processing rules as the "aud" claim defined in Section 4.1.3 of [RFC7519], except that the value of the audience claim is a StringOrURI when it is not an array or each of the audience array element values is a StringOrURI when the audience claim value is an array. (StringOrURI is defined in Section 2 of this specification.) 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 and processing rules as the "exp" claim defined in Section 4.1.4 of [RFC7519], except that the value is a NumericDate, as defined in Section 2 of this specification. 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 and processing rules as the "nbf" claim defined in Section 4.1.5 of [RFC7519], except that the value is a NumericDate, as defined in Section 2 of this specification. 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 and processing rules as the "iat" claim defined in Section 4.1.6 of [RFC7519], except that the value is a NumericDate, as defined in Section 2 of this specification. 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 and processing rules as the "jti" claim defined in Section 4.1.7 of [RFC7519], except that the value is a byte string. The Claim Key 7 is used to identify this claim.
4. Summary of the Claim Names, Keys, and Value Types

<table>
<thead>
<tr>
<th>Name</th>
<th>Key</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>iss</td>
<td>1</td>
<td>text string</td>
</tr>
<tr>
<td>sub</td>
<td>2</td>
<td>text string</td>
</tr>
<tr>
<td>aud</td>
<td>3</td>
<td>text string</td>
</tr>
<tr>
<td>exp</td>
<td>4</td>
<td>integer or floating-point number</td>
</tr>
<tr>
<td>nbf</td>
<td>5</td>
<td>integer or floating-point number</td>
</tr>
<tr>
<td>iat</td>
<td>6</td>
<td>integer or floating-point number</td>
</tr>
<tr>
<td>cti</td>
<td>7</td>
<td>byte string</td>
</tr>
</tbody>
</table>

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 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 Section 4 of [RFC8152] ("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 Section 6 of [RFC8152] ("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 Section 5 of [RFC8152] ("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.

Syntactically, the signing and encryption operations for Nested CWTs may be applied in any order; however, if both signing and encryption are necessary, producers normally 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

IANA has created the "CBOR Web Token (CWT) Claims" registry
[IANA.CWT.Claims].

Registration requests are evaluated using the criteria described in
the Claim Key instructions in the registration template below after a
due three-week review period on the cwt-reg-review@ietf.org mailing list,
on the advice of one or more Designated Experts [RFC8126]. 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.

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. Registrations for the limited set
of values between -256 and 255 and strings of length 1 are to be
restricted to claims with general applicability.

IANA must only accept registry updates from the Designated Experts
and should direct all requests for registration to the review mailing
list.

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.

Since a high degree of overlap is expected between the contents of
the "CBOR Web Token (CWT) Claims" registry and the "JSON Web Token
Claims" registry, overlap in the corresponding pools of Designated
Experts would be useful to help ensure that an appropriate level of
coordination between the registries is maintained.
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. Different ranges of values use different registration policies [RFC8126]. Integer values from -256 to 255 and strings of length 1 are designated as Standards Action. Integer values from -65536 to -257 and from 256 to 65535 along with strings of length 2 are designated as Specification Required. Integer values 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): [RFC8392]
- 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 [RFC8392]

- 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 [RFC8392]

- 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 [RFC8392]

- 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 [RFC8392]

- 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 [RFC8392]

- 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 [RFC8392]
9.2. Media Type Registration

IANA has registered 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 [RFC8392]
- Interoperability considerations: N/A
- Published specification: [RFC8392]
- 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

IANA has registered 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: 61
- Reference: [RFC8392]

9.4. CBOR Tag registration

IANA has registered the CWT CBOR tag in the "CBOR Tags" registry [IANA.CBOR.Tags].

9.4.1. Registry Contents

- CBOR Tag: 61
- Data Item: CBOR Web Token (CWT)
- Semantics: CBOR Web Token (CWT), as defined in [RFC8392]
- Reference: [RFC8392]
- Point of Contact: Michael B. Jones, mbj@microsoft.com

10. References

10.1. Normative References

10.2. Informative References


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.

a70175636f61703a2f2f61732e6578616d706c65636f6d02656572696573818636f61703a2f2f61732e6578616d706c65636f6d041a5612aeb051a5610d9f0061a5610d9f007420b71

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

```
a42050231f4c4d4d3051f0d2ec0a3851d5b3830104024c53796d6d6574726963
313238030a
```

Figure 4: 128-Bit Symmetric COSE_Key as Hex String

```
{  
  / k /   -1: h'231f4c4d4d3051f0d2ec0a3851d5b383'  
  / 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'403697de87af64611c1d32a05dab0fe1fcb715a86ab435f1ec99192d795693880104024c53796d6d6574726963323536030a'  
  / kty /  1: 4 / Symmetric /,  
  / kid /  4: h'53796d6d65747269633235363' / 'Symmetric256' /,  
  / alg /  3: 4 / HMAC 256/64 /  
}
```

Figure 7: 256-Bit Symmetric COSE_Key in CBOR Diagnostic Notation
A.2.3. Elliptic Curve Digital Signature Algorithm (ECDSA) P-256 256-Bit COSE Key

a72358206c1382765aec5358f117733d281c1c7bdc39884d04a45a1e6c67c858bc206c1922582060f7f1a780d8a783bf7a2dd6b2796e8128dbbcecf9d3d168db9529971a36e7b9215820143329c7e786e416927599cf65a34f3ce2ffda55a7eca69ed8919a394d4f0f2001010202524173796d6d6574726963454453413235360326

Figure 8: ECDSA 256-Bit COSE Key as Hex String

```
{ 
  /d/ -4: h'6c1382765aec5358f117733d281c1c7bdc39884d04a45a1e6c67c858bc206c19'
  /y/ -3: h'60f7f1a780d8a783bf7a2dd6b2796e8128dbbcecf9d3d168db9529971a36e7b9'
  /x/ -2: h'143329c7e786e416927599cf65a34f3ce2ffda55a7eca69ed8919a394d4f0f'
  /crv/ -1: 1 / P-256 /
  /kty/  1: 2 / EC2 /
  /kid/  2: h'4173796d6d57472696345445341323536' / '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.

d2843a10126a104524173796d6d574726963454453413235365850a7017563f61703a2f2f61732e6578616d706c652e636f6d02656572696b77037818636f61732e6578616d706c652e636f6d041a5612ae0051a5610d9f0061a5610d9f007420b7158405427c1ff28d23fbd1f29c4c7c6a555e6016df6a29f9179bc3d7438bacaca5acd08c8d4d4f96131680c429a01f85951eceee743a52b9b63632c57290120e1c9e30

Figure 10: Signed CWT as Hex String
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.

d83dd18443a10104a1044c53796d6d65747269633235365850a70175636f61703a2f2f61732e6578616d706c6f6d041a5612aeb0051a5610d9f06
1a5610d9f007420b714803101ef6d789200

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
f400ee1d0dc60f6b559c02421fd384fc2ebe22d7071378b0ea7428fff157444d
45f7e6afcd1aae5f6495830c58627087fc5b4974f319a8707a635dd643b

Figure 14: Encrypted CWT as Hex String
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.

Figure 16: Signed and Encrypted CWT as Hex String

```bash
Figure 16: Signed and Encrypted CWT as Hex String
```
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.

d18443a10104a1044c53796d6d6574726963235364ba106fb41d584367c2000
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

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, Alissa Cooper, Esko Dijk, Benjamin Kaduk, Warren Kumari, Carlos Martinez, Alexey Melnikov, Kathleen Moriarty, Eric Rescorla, Dan Romascanu, Adam Roach, Kyle Rose, Jim Schaad, Ludwig Seitz, and Goeran Selander.
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