RATS Working Group
Internet-Draft
Intended status: Standards Track
Expires: 19 November 2026

H. Birkholz
Fraunhofer SIT
T. Fossati
Linaro
W. Pan
Huawei Technologies
I. Mihalcea
Arm
C. Bormann
Universit辰t Bremen TZI

18 May 2026

Epoch Markers
draft-ietf-rats-epoch-markers-04

Abstract

This document defines Epoch Markers as a means to establish a notion of freshness among actors in a distributed system. Epoch Markers are similar to "time ticks" and are produced and distributed by a dedicated system known as the Epoch Bell. Systems receiving Epoch Markers do not need to track freshness using their own understanding of time (e.g., via a local real-time clock). Instead, the reception of a specific Epoch Marker establishes a new epoch that is shared among all recipients. This document defines Epoch Marker types, including CBOR time tags, RFC 3161 TimeStampToken, and nonce-like structures. It also defines a CWT Claim to embed Epoch Markers in RFC 8392 CBOR Web Tokens, which serve as vehicles for signed protocol messages.

About This Document

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

Status information for this document may be found at https://datatracker.ietf.org/doc/draft-ietf-rats-epoch-markers/.

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

Source for this draft and an issue tracker can be found at https://github.com/ietf-rats-wg/draft-birkholz-rats-epoch-marker.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Systems that need to interact securely often require a shared understanding of the freshness of conveyed information. This is certainly the case in the domain of remote attestation procedures. In general, securely establishing a shared notion of freshness of the exchanged information among entities in a distributed system is not a simple task.

The entire Appendix A of [RFC9334] deals solely with the topic of freshness, which is in itself an indication of how relevant, and complex, it is to establish a trusted and shared understanding of freshness in a RATS system.

This document defines Epoch Markers as a way to establish a notion of freshness among actors in distributed systems. Epoch Markers are similar to "time ticks" and are produced and distributed by a dedicated system, the Epoch Bell. Actors in a system that receive Epoch Markers do not have to track freshness using their own understanding of time (e.g., via a local real-time clock). Instead, the reception of a certain Epoch Marker establishes a new epoch that is shared between all recipients. In essence, the emissions and corresponding receptions of Epoch Markers are like the ticks of a clock, with these ticks being conveyed over the Internet.

In general (barring highly symmetrical topologies), epoch ticking incurs differential latency due to the non-uniform distribution of receivers with respect to the Epoch Bell. This introduces skew that needs to be taken into consideration when Epoch Markers are used.

While all Epoch Markers share the same core property of behaving like clock ticks in a shared domain, various "Epoch ID" values are defined as Epoch Marker types in this document to accommodate different use cases and diverse kinds of Epoch Bells.

While most Epoch Markers types are encoded in CBOR [STD94], and many of the Epoch ID types are themselves encoded in CBOR, a prominent format in this space is the TimeStampToken (TST) defined by [RFC3161], a DER-encoded TSTInfo value wrapped in a CMS envelope [RFC5652]. TSTs are produced by Time-Stamp Authorities (TSA) and exchanged via the Time-Stamp Protocol (TSP). At the time of writing, TSAs are the most common providers of secure time-stamping services. Therefore, reusing the core TSTInfo structure as an Epoch ID type for Epoch Markers is instrumental for enabling smooth migration paths and promote interoperability. There are, however, several other ways to represent a signed timestamp or the start of a new freshness epoch, respectively, and therefore other Epoch Marker types.

To inform the design, this document discusses a number of interaction models in which Epoch Markers are expected to be exchanged. The default top-level structure of Epoch Markers described in this document is CBOR Web Tokens (CWT) [RFC8392]. The present document specifies an extensible set of Epoch Marker types, along with the em CWT claim to include them in CWTs. CWTs are signed using COSE [STD96] and benefit from wide tool support. However, CWTs are not the only containers in which Epoch Markers can be embedded. Epoch Markers can be included in any type of message that allows for the embedding of opaque bytes or CBOR data items. Examples include the Collection CMW in [I-D.ietf-lamps-csr-attestation], Evidence formats such as [TCG-CoEvidence] or [I-D.ietf-rats-eat], Attestation Results formats such as [I-D.ietf-rats-ar4si], or the CWT Claims Header Parameter of [I-D.ietf-scitt-architecture].

Epoch markers can be used in the following ways:

* as embeddings in other data formats

* as information elements in protocols

* in systems that integrate the aforementioned protocols or data formats

* in the deployment of such systems

All of these can be considered "users" of Epoch Markers and will be referred to as entities "using Epoch Markers” throughout the document.

1.1. Terminology

This document makes use of the following terms from other documents:

* "conceptual messages" as defined in Section 8 of [RFC9334]

* "freshness" and "epoch" as defined in Section 10 of [RFC9334]

* "handle" as defined in Section 6 of [I-D.ietf-rats-reference-interaction-models]

* "Time-Stamp Authority" as defined by [RFC3161]

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.

In this document, CDDL [RFC8610] is used to describe the data formats. The examples in Appendix A use the CBOR Extended Diagnostic Notation (EDN, [I-D.ietf-cbor-edn-literals]).

2. Epoch IDs

The RATS architecture introduces the concept of Epoch IDs that mark certain events during remote attestation procedures ranging from simple handshakes to rather complex interactions including elaborate freshness proofs. The Epoch Markers defined in this document are a solution that includes the lessons learned from TSAs, the concept of Epoch IDs defined in the RATS architecture, and provides several means to identify a new freshness epoch. Some of these methods are introduced and discussed in Section 10.3 of [RFC9334] (the RATS architecture).

3. Interaction Models

The interaction models illustrated in this section are derived from the RATS Reference Interaction Models [I-D.ietf-rats-reference-interaction-models]. In general, there are three major interaction models used in remote attestation:

* ad-hoc requests (e.g., via challenge-response requests addressed at Epoch Bells), corresponding to Section 7.1 of [I-D.ietf-rats-reference-interaction-models]

* unsolicited distribution (e.g., via uni-directional methods, such as broad- or multicasting from Epoch Bells), corresponding to Section 7.2 of [I-D.ietf-rats-reference-interaction-models]

* solicited distribution (e.g., via a subscription to Epoch Bells), corresponding to Section 7.3 of [I-D.ietf-rats-reference-interaction-models]

In all three interaction models, Epoch Markers can be used as content for the generic information element handle as introduced by [I-D.ietf-rats-reference-interaction-models]. Handles are used to establish freshness in ad-hoc, unsolicited, and solicited distribution mechanisms of an Epoch Bell. For example, an Epoch Marker can be used as a nonce in challenge-response remote attestation (e.g., for limiting the number of ad-hoc requests by a Verifier). If embedded in a CWT, an Epoch Marker can be used as a handle by extracting the value of the em Claim or by using the complete CWT including an em Claim (e.g., functioning as a signed time-stamp token). Using an Epoch Marker requires the challenger to acquire an Epoch Marker beforehand, which may introduce a sensible overhead compared to using a simple nonce.

When an Epoch Marker is used as the handle in the Passport or Background-Check challenge-response flows defined by [I-D.ietf-rats-reference-interaction-models], the Verifier MUST follow the relevant Epoch Marker validation and acceptance policy. That policy MUST fulfil the applicable requirements presented in this document, such as accepted epoch windows, replay or rollback detection state, and deployment-specific skew introduced by the Evidence path.

For Background-Check, the policy MUST account for the additional path through the Relying Party before Evidence reaches the Verifier. For Passport, any Attestation Result derived from such Evidence MUST remain bound to the Epoch Marker, or to the handle containing it, so that subsequent freshness decisions can be made according to Relying Party policy.

4. Epoch Marker Structure

This section specifies the structure of Epoch Marker types using CDDL [RFC8610] and illustrates their usage and relationship with other IETF work (e.g, [RFC3161]) where applicable. In general, Epoch Markers are intended to be conveyed securely, e.g., by being included in a signed data structure, such as a CBOR Web Token (CWT), or by being sent via a secure channel. The specification of such "outer" structures and protocols and the means how to secure them is beyond the scope of this document. This document defines the different

types of Epoch Markers in Section 7.1. When the media type application/epoch-marker+cbor is used to label content as an Epoch Marker, the em-type media type parameter can optionally specify the Epoch Marker type by referencing its CBOR tag number. For example, an Epoch Marker can be used to construct a CBOR-based trusted time stamp token, similar in function to a [RFC3161] TimeStampToken, using CWT and the em Claim defined in this document (see Figure 5 for an illustration). The value(s) that an Epoch Marker represents are intended to demonstrate freshness of messages and protocols, but they can also serve other purposes in cases where trusted timestamps or time intervals are required. Taken as an opaque value, it is possible to use Epoch Markers as values for a nonce field in existing data structures or protocols that already support extra data fields, such as extraData in TPMS_ATTEST [TCG-TPM2]. Similarities in the usage of nonces and Epoch Markers can sometimes lead to applications where both are used in the same interaction, albeit in different places and for different purposes. One example of such an application scenario is the "nested" use of classical nonces and Epoch Markers, whereby an Epoch Marker is requested to be used as a nonce value for a specific data structure, while a locally generated nonce is used to retrieve that Epoch Marker via an "outer" ad hoc interaction (e.g., nonce retrieval protocols that interact with an Epoch Bell to fetch an Epoch Marker to be used as a nonce). As some Epoch Marker types represent certain timestamp variants, these Epoch Markers or the secure conveyance method they are used in do not necessarily have a hard-coded message imprint, as is always the case with [RFC3161] TimeStampTokens. This means that not all Epoch Marker types support binding a message to an Epoch Marker (unlike the example in Figure 5.

The following Epoch Marker types are defined in this document:

```
epoch-marker = $tagged-epoch-id

; epoch-id types independent of interaction model
$tagged-epoch-id /= cbor-time
$tagged-epoch-id /= #6.26980(classical-rfc3161-TST-info)
$tagged-epoch-id /= #6.26981(TST-info-based-on-CBOR-time-tag)
$tagged-epoch-id /= #6.26982(epoch-tick)
$tagged-epoch-id /= #6.26983(epoch-tick-list)
$tagged-epoch-id /= #6.26984(strictly-monotonic-counter)
$tagged-epoch-id /= #6.26985(epoclet)
```

Figure 1: Epoch Marker types (tag numbers 2698x are suggested, not yet allocated)

```
$$Claims-Set-Claims //= (&(em: 2000) => epoch-marker)
```

Figure 2: Epoch Marker as a CWT Claim (CWT claim number 2000 is suggested, not yet allocated)

4.1. Epoch Marker Types

This section specifies the Epoch Marker types listed in Figure 1.

4.1.1. CBOR Time Tags

CBOR Time Tags are CBOR time representations choosing from CBOR tag 0 (tdate, RFC3339 time as a string), tag 1 (time, POSIX time as int or float), or tag 1001 (extended time data item).

```
cbor-time = tdate / time / etime

;# import etime from rfc9581
```

The CBOR Time Tag represents a freshly sourced timestamp represented as either time or tdate (Sections 3.4.2 and 3.4.1 of RFC 8949 [STD94], Appendix D of [RFC8610]), or etime (Section 3 of [RFC9581]). The etime rule shown in the CDDL above is imported from Section 6 of [RFC9581]; RFC 9581 remains the authoritative specification of CBOR tag 1001, and this document neither updates nor redefines it. This document merely profiles etime for use with Epoch Markers.

4.1.1.1. Creation

To generate the cbor-time value, the emitter MUST follow the requirements in Section 4.2.

4.1.2. Classical RFC 3161 TST Info

DER-encoded [X.690] TSTInfo [RFC3161]. See Appendix A.1 for the layout.

```
classical-rfc3161-TST-info = bytes
```

The following describes the classical-rfc3161-TST-info type.

classical-rfc3161-TST-info:
: The DER-encoded TSTInfo generated by a [RFC3161] Time Stamping Authority.

4.1.2.1. Creation

The Epoch Bell MUST use the following value as MessageImprint in its request to the TSA:

```
SEQUENCE {
  SEQUENCE {
    OBJECT      2.16.840.1.101.3.4.2.1 (sha256)
    NULL
  }
  OCTET STRING
    BF4EE9143EF2329B1B778974AAD445064940B9CAE373C9E35A7B23361282698F
}
```

This is the sha-256 hash of the string "EPOCH_BELL".

The TimeStampToken obtained from the TSA MUST be stripped of the TSA signature. Only the TSTInfo is to be kept the rest MUST be discarded. The Epoch Bell COSE signature will replace the TSA signature.

4.1.3. CBOR-encoded RFC3161 TST Info

The TST-info-based-on-CBOR-time-tag is semantically equivalent to classical [RFC3161] TSTInfo, rewritten using the CBOR type system.

```
TST-info-based-on-CBOR-time-tag = {
  &(version : 0) => v1
  &(policy : 1) => oid
  &(messageImprint : 2) => MessageImprint
  &(serialNumber : 3) => integer
  &(eTime : 4) => profiled-etime
  ? &(ordering : 5) => bool .default false
  ? &(nonce : 6) => integer
  ? &(tsa : 7) => GeneralName
  * $$TSTInfoExtensions
}

v1 = 1

oid = #6.111(bstr) / #6.112(bstr)

MessageImprint = [
  hashAlg : int
  hashValue : bstr
]

profiled-etime = #6.1001(timeMap)
timeMap = {
  1 => ~time
  ? -8 => profiled-duration
  * int => any
}
profiled-duration = {* int => any}

GeneralName = [ GeneralNameType : int, GeneralNameValue : any ]
; See Section 4.2.1.6 of RFC 5280 for type/value
```

The following describes each member of the TST-info-based-on-CBOR-time-tag map.

version:
: The integer value 1. Cf. version, Section 2.4.2 of [RFC3161].

policy:
: A [RFC9090] object identifier tag (111 or 112) representing the TSA’s policy under which the tst-info was produced. Cf. policy, Section 2.4.2 of [RFC3161].

messageImprint:
: A [RFC9054] COSE_Hash_Find array carrying the hash algorithm identifier and the hash value of the time-stamped datum. Cf. messageImprint, Section 2.4.2 of [RFC3161].

serialNumber:
A unique integer value assigned by the TSA to each issued tst-info. Cf. serialNumber, Section 2.4.2 of [RFC3161].

eTime:
The time at which the tst-info has been created by the TSA. Cf. genTime, Section 2.4.2 of [RFC3161]. Encoded as extended time [RFC9581], indicated by CBOR tag 1001. A separate tag is unnecessary because the profiled form remains an etime item carried with tag 1001. RFC 9581 defines tag 1001 and its semantics; this document constrains one valid etime shape for Epoch Markers but does not introduce a new CBOR tag or alter the definition of etime. The resulting profiled-etime remains a subset/profile of etime encoded with tag 1001 and is defined as follows:

* The "base time" is encoded using key 1, indicating POSIX time as int or float, to align the profile with the baseline representation defined in [RFC9581].

* The stated "accuracy" is encoded using key -8, which indicates the maximum allowed deviation from the value indicated by "base time". The duration map is profiled to disallow string keys. This is an optional field specialized to the needs of this profile.

* The map MUST include key 1 and MAY include additional negative integer keys, which can be used to encode supplementary information. Unsigned integer keys, including 4, 5, and the keys in the $$ETIME-CRITICAL group choice, MUST NOT be used, unless the emitter wants to ensure that only implementations that understand the critical key interoperate.

Fixing key 1 for the base time and profiling key -8 provide a consistent shape for the profiled etime maps consumed by Epoch Markers without changing the meaning of the corresponding keys in [RFC9581]. Key 1 is pinned so all profiled etime values share the same base-time representation, and key -8 is specialized to carry the accuracy bound expected by the TSA-style timestamps used here. The permissive * int => any entry remains to allow other members defined by [RFC9581] and future extensions; implementations still rely on [RFC9581] for the full semantics and validation of etime beyond the constraints listed here.

ordering:
boolean indicating whether tst-info issued by the TSA can be ordered solely based on the "base time". This is an optional field, whose default value is "false". Cf. ordering, Section 2.4.2 of [RFC3161].

nonce:
int value echoing the nonce supplied by the requestor. Cf. nonce, Section 2.4.2 of [RFC3161].

tsa:
a single-entry GeneralNames array Section 11.8 of [I-D.ietf-cose-cbor-encoded-cert] providing a hint in identifying the name of the TSA. Cf. tsa, Section 2.4.2 of [RFC3161].

$$TSTInfoExtensions:
A CDDL socket (Section 3.9 of [RFC8610]) to allow extensibility of the data format. Note that any extensions appearing here MUST match an extension in the corresponding request. Cf. extensions, Section 2.4.2 of [RFC3161].

4.1.3.1. Creation

The Epoch Bell MUST use the following value as messageImprint in its request to the TSA:

```
[
    / hashAlg   / -16, / sha-256 /
    / hashValue / h'BF4EE9143EF2329B1B778974AAD44506
                    4940B9CAE373C9E35A7B23361282698F'
]
```

This is the sha-256 hash of the string "EPOCH_BELL".

4.1.4. Epoch Tick

An Epoch Tick is a single opaque blob sent to multiple consumers.

```
; Epoch-Tick

epoch-tick = tstr / bstr / int
```

The following describes the epoch-tick type.

epoch-tick:
Either a string, a byte string, or an integer used by RATS roles within a trust domain as extra data (handle) included in conceptual messages [RFC9334]. Similarly to the use of nonces, this allows the conceptual messages to be associated with a certain epoch. However, unlike nonces (which require uniqueness), Epoch Markers can be used in multiple interactions by every consumer involved.

4.1.4.1. Creation

The emitter MUST follow the requirements in Section 4.3.

4.1.5. Epoch Tick List

A list of Epoch Ticks sent to multiple consumers. The consumers use each Epoch Tick in the list sequentially. Similarly to the use of nonces, this allows each interaction to be associated with a certain epoch. However, unlike nonces (which require uniqueness), Epoch Markers can be used in multiple interactions by every consumer involved.

```
; Epoch-Tick-List

epoch-tick-list = [ + epoch-tick ]
```

The following describes the Epoch Tick List type.

epoch-tick-list:
: A sequence of byte strings used by RATS roles in trust domain as extra data (handle) in the generation of conceptual messages as specified by the RATS architecture [RFC9334] to associate them with a certain epoch. Each Epoch Tick in the list is used in a consecutive generation of a conceptual message. Asserting freshness of a conceptual message including an Epoch Tick from the epoch-tick-list requires some state on the receiver side to assess if that Epoch Tick is the appropriate next unused Epoch Tick from the epoch-tick-list.

4.1.5.1. Creation

The emitter MUST follow the requirements in Section 4.3.

4.1.5.2. Usage

Proving freshness requires receiver-side state to identify the “next unused” tick. Systems using Epoch Tick lists SHOULD define how missing/out-of-order ticks are handled and how resynchronization occurs, as per Section 4.4.

4.1.6. Strictly Monotonically Increasing Counter

A strictly monotonically increasing counter.

The counter context is defined by the Epoch Bell.

```
strictly-monotonic-counter = uint
```

The following describes the strictly-monotonic-counter type.

strictly-monotonic-counter:
An unsigned integer used by RATS roles in a trust domain as extra data in the production of conceptual messages as specified by the RATS architecture [RFC9334] to associate them with a certain epoch. Each new strictly-monotonic-counter value must be higher than the last one.

4.1.6.1. Usage

Systems that use Epoch Markers SHOULD follow the guidance in Section 4.4 in establishing an Epoch Marker acceptance policy for receivers. To prove freshness, receivers SHOULD track the highest accepted counter and ensure it fulfills the acceptance policy.

4.1.7. Epoclet

In a highly available service (e.g., a cloud attestation Verifier), maintaining per-session nonce state can cause scalability issues. One alternative is to use time-synchronized servers that share a symmetric key and produce and consume nonces based on epoch ticks signed using the shared secret. This means that a nonce minted by one server can be processed by any other server, avoiding the need for session "stickiness".

An epoclet is an Epoch ID variant that supports the above use case by encoding a POSIX time (i.e., the epoch identifier) alongside a minimal set of metadata. This is all authenticated with a symmetric key in a self-contained and compact token that fits within 64 bytes. This makes it easy to use with common Evidence retrieval ABIs, which tend to limit the size of the challenge parameter to 64 bytes.

```
epoclet = [
  TimeToken
  AuthTag: bytes .size 32
]

TimeToken = [
  KeyID: bytes .size 1
  Timestamp: ~posix-time
  Pad: bytes .size (0..20)
]

posix-time = #6.1(int)
```

The following describes the epoclet type.

KeyID:
A one-byte opaque identifier that is shared across the server pool. It is used to identify the key used to compute the AuthTag. Its interpretation is deployment-specific.

Timestamp:
The time associated with the current epoch encoded as the CBOR tag for POSIX time. It MUST use the int format. The CBOR tag MUST be removed. Note that this encoding restricts the granularity of this Epoch ID to one second.

Pad:
Zero or more (up to 20) bytes of padding. This field is used to adjust the overall size of the epoclet, ranging from a minimum of 44 bytes (0 bytes of padding) to a maximum of 64 bytes (20 bytes of padding). The padding bytes can assume any value.

AuthTag:
32 bytes string which is the result of applying HMAC [RFC2104] with SHA-256 over the CBOR serialization of the TimeToken array. The serialization MUST use the CBOR deterministic encoding as specified in Section 4.2 of RFC 8949 [STD94].

4.1.7.1. Usage

The same size limit defined in Section 4.3 applies: an encoded epoclet MUST be no more than 64 bytes in length.

The requirements in Section 4.2 apply. Furthermore, when used in a server farm, the clocks of the servers that participate in the protocol MUST be synchronized.

It is RECOMMENDED that the handling (i.e., storage, replication and rotation) of the symmetric key used to generate the authentication tag follows the recommendations and best practices described in [NIST-SP-800-pt2].

4.2. Time Requirements

Time MUST be sourced from a trusted clock (see Section 10.1 of [RFC9334]).

4.3. Nonce Requirements

A nonce value used in a protocol or message to retrieve an Epoch Marker MUST be freshly generated. The generated value MUST have at least 64 bits of entropy (before encoding). The generated value MUST be generated via a cryptographically secure random number generator.

A maximum nonce size of 512 bits is set to limit the memory requirements. All receivers MUST be able to accommodate the maximum size.

4.4. State and Sequencing Management

Data structures containing Epoch Markers could be reordered in-flight even without malicious intent, leading to perceived sequencing issues. Some Epoch Marker types thus require receiver state to detect replay/rollback or establish sequencing. Systems that use Epoch Markers SHOULD define an explicit acceptance policy (e.g., bounded acceptance window) that accounts for reordering of markers.

There is a trade-off between keeping a single "global" epoch view versus per-Attester state at the Verifier: global-only policies can exacerbate latency-induced false replay rejections, while per-Attester tracking can be costly. Systems that use Epoch Markers SHOULD document whether they use global epoch tracking or per-Attester state and, if necessary, the associated window.

5. Signature Requirements

The signature over an Epoch Marker MUST be generated by the Epoch Bell. Conversely, applying the first signature to an Epoch Marker always makes the issuer of a signed message containing an Epoch Marker an Epoch Bell.

6. Security Considerations

In distributed systems that rely on Epoch Markers for conveyance of freshness, the Epoch Bell plays a significant role in the assumed trust model. Freshness decisions derived from Epoch Markers depend on the Epoch Bell's key(s) and correct behavior. If the Epoch Bell key is compromised, or the Bell is malicious/misconfigured, an attacker can emit valid-looking "fresh" Epoch Markers. System deployments using Epoch Markers generally need to protect Bell signing keys (secure storage, rotation, revocation) and scope acceptance to the intended trust domain (e.g., expected issuer/trust anchor). Similarly, the Bell's clock needs to be securely sourced and managed, to prevent attacks that skew the Bell's perception of time.

6.1. Epoch Signalling Issues

Section 12.3 of [RFC9334] provides a good introduction to attacks on conveyance of Epoch Markers. A network adversary can replay validly signed Epoch Markers or delay distribution, and differential latency can lead to different parties having different views of the "current" epoch.

The epoch (acceptable window) duration is an operational security parameter: if too long, an Attester can create "good" Evidence in a good state and release it later while the epoch is still acceptable (notably for epoch-tick, epoch-tick-list, and strictly-monotonic-counter); if too short, distant Attesters may be rejected as stale due to latency. Epoch Markers are also designed to be reusable by multiple consumers, unlike nonces. Where per-session uniqueness is required, protocols typically need to bind Epoch Markers to an explicit nonce (e.g., see Section 4). Finally, system deployments using Epoch Markers are normally required to pin which Epoch Marker types are acceptable for a given trust domain to avoid downgrade.

6.2. Operational Examples

The following illustrative cases highlight "reasonable best practice" choices for balancing freshness, replay protection, and scalability.

* _Nonce-bound Bell interaction_: When a Verifier uses a nonce challenge to trigger Evidence creation, the Attester can forward that nonce to the Epoch Bell to request an Epoch Marker with the nonce echoed inside. For reuse and caching, the typical pattern is to keep the marker generic and embed the Verifier nonce alongside the marker in the Evidence: if the Bell signs a nonce-echoed marker, that marker is not reusable across sessions. The nonce and marker are thus either bound by the Bell's signature, or by the attester's signature on the Evidence. The Verifier checks that (1) the nonce matches its challenge, (2) the Epoch Marker signature chains to the expected Bell key, and (3) the marker satisfies the acceptance policy (e.g., highest-seen counter or time window). This pairing gives per-session uniqueness while still allowing Epoch Marker reuse by multiple consumers.

* _Long-latency paths (e.g., LoRaWAN or DTN profiles)_: High propagation and queuing delays make tight epoch windows brittle. In system deployments using Epoch Markers, epoch-tick-list can be pre-provisioned to Attesters so that each interaction consumes the next tick, with the Verifier keeping per-Attester sequencing state (Section 4.4). Epoch duration should cover worst-case delivery plus clock skew of the Bell, and acceptance policies should allow an overlap (e.g., current and immediately previous epoch) to absorb in-flight drift while still rejecting replays beyond that window.

* _Large fleets sharing a Bell_: When many Attesters reuse the same Epoch Marker, per-Attester state at the Verifier may be impractical. One approach is to accept a global highest-seen epoch (with a bounded replay window) while requiring each Evidence record to bind the Epoch Marker to the Attester identity and, when feasible, a Verifier-provided nonce. This limits cross-attester replay of a single Epoch Marker while keeping the Bell stateless, which allows Epoch Markers to be cached and enables their broadcast distribution at scale.

7. IANA Considerations

// RFC Editor: please replace RFCthis with the RFC number of this RFC
// and remove this note.

7.1. New CBOR Tags

IANA is requested to allocate the following tags in the "CBOR Tags" registry [IANA.cbor-tags], preferably with the specific CBOR tag value requested:

Tag	Data Item	Semantics	Reference
26980	bytes	DER-encoded RFC3161 TSTInfo	Section 4.1.2 of RFCthis
26981	map	CBOR representation of RFC3161 TSTInfo semantics	Section 4.1.3 of RFCthis
26982	tstr / bstr / int	a nonce that is shared among many participants but that can only be used once by each participant	Section 4.1.4 of RFCthis
26983	array	a list of multi-nonce	Section 4.1.5 of RFCthis
26984	uint	strictly monotonically increasing counter	Section 4.1.6 of RFCthis
26985	array	an "epoclet"	Section 4.1.7 of RFCthis

Table 1: New CBOR Tags

7.2. New EM CWT Claim

This specification adds the following value to the "CBOR Web Token Claims" registry [IANA.cwt].

* Claim Name: em
* Claim Description: Epoch Marker
* Claim Key: 2000 (IANA: suggested assignment)
* Claim Value Type(s): CBOR array
* Change Controller: IETF
* Specification Document(s): Section 4 of RFCthis

7.3. New Media Type application/epoch-marker+cbor

IANA is requested to add the application/epoch-marker+cbor media types to the "Media Types" registry [IANA.media-types], using the following template:

Type name: application
Subtype name: epoch-marker+cbor
Required parameters: no
Optional parameters: em-type: the CBOR tag number that identifies the specific Epoch Marker. This value is encoded as an unsigned decimal integer, without leading zeroes, except for the actual number 0. For example, em-type=26982 identifies an Epoch Tick.
Encoding considerations: binary (CBOR)
Security considerations: Section 6 of RFCthis
Interoperability considerations: n/a
Published specification: RFCthis
Applications that use this media type: RATS Attesters, Verifiers, Endorsers and Reference-Value providers, and Relying Parties that need to transfer Epoch Markers payloads over HTTP(S), CoAP(S), and other transports.
Fragment identifier considerations: The syntax and semantics of fragment identifiers are as specified for "application/cbor". (No fragment identification syntax is currently defined for "application/cbor".)
Person & email address to contact for further information: RATS WG mailing list (rats@ietf.org)
Intended usage: COMMON
Restrictions on usage: none
Author/Change controller: IETF
Provisional registration: no

7.4. New CoAP Content-Formats

IANA is requested to register the following Content-Format IDs in the "CoAP Content-Formats" registry, within the "Constrained RESTful Environments (CoRE) Parameters" registry group [IANA.core-parameters]:

Content-Type	Content Coding	ID	Reference
application/epoch-marker+cbor	-	TBD1	RFCthis
application/epoch-marker+cbor; em-type=0	-	TBD2	Section 4.1.1 of RFCthis
application/epoch-marker+cbor; em-type=1	-	TBD3	Section 4.1.1 of RFCthis
application/epoch-marker+cbor; em-type=1001	-	TBD4	Section 4.1.1 of RFCthis
application/epoch-marker+cbor; em-type=26980	-	TBD5	Section 4.1.2 of RFCthis
application/epoch-marker+cbor; em-type=26981	-	TBD6	Section 4.1.3 of RFCthis
application/epoch-marker+cbor; em-type=26982	-	TBD7	Section 4.1.4 of RFCthis
application/epoch-marker+cbor; em-type=26983	-	TBD8	Section 4.1.5 of RFCthis
application/epoch-marker+cbor; em-type=26984	-	TBD9	Section 4.1.6 of RFCthis
application/epoch-marker+cbor; em-type=26985	-	TBD10	Section 4.1.7 of RFCthis

Table 2: New CoAP Content Formats

If possible, TBD1..TBD10 should be assigned in the 256..9999 range.

8. References

8.1. Normative References

[I-D.ietf-cbor-edn-literals]
Bormann, C., "CBOR Extended Diagnostic Notation (EDN)", Work in Progress, Internet-Draft, draft-ietf-cbor-edn-literals-24, 11 May 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-cbor-edn-literals-24>.

[I-D.ietf-cose-cbor-encoded-cert]
Mattsson, J. P., Selander, G., Raza, S., Hglund, J., Furuhed, M., and L. Liao, "CBOR Encoded X.509 Certificates (C509 Certificates)", Work in Progress, Internet-Draft, draft-ietf-cose-cbor-encoded-cert-19, 11 May 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-cose-cbor-encoded-cert-19>.

[IANA.core-parameters]
IANA, "Constrained RESTful Environments (CoRE) Parameters", <https://www.iana.org/assignments/core-parameters>.

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

[NIST-SP-800-pt2]
Barker, E. and W. Barker, "Recommendation for key management:: part 2 -- best practices for key management organizations", National Institute of Standards and Technology, DOI 10.6028/nist.sp.800-57pt2r1, May 2019, <https://doi.org/10.6028/nist.sp.800-57pt2r1>.

[RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, DOI 10.17487/RFC2104, February 1997, <https://www.rfc-editor.org/rfc/rfc2104>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC3161] Adams, C., Cain, P., Pinkas, D., and R. Zuccherato, "Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP)", RFC 3161, DOI 10.17487/RFC3161, August 2001, <https://www.rfc-editor.org/rfc/rfc3161>.

[RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, DOI 10.17487/RFC5652, September 2009, <https://www.rfc-editor.org/rfc/rfc5652>.

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

[RFC8392] Jones, M., Wahlstroem, E., Erdtman, S., and H. Tschofenig, "CBOR Web Token (CWT)", RFC 8392, DOI 10.17487/RFC8392, May 2018, <https://www.rfc-editor.org/rfc/rfc8392>.

[RFC8610] Birkholz, H., Vigano, C., and C. Bormann, "Concise Data Definition Language (CDDL): A Notational Convention to Express Concise Binary Object Representation (CBOR) and JSON Data Structures", RFC 8610, DOI 10.17487/RFC8610, June 2019, <https://www.rfc-editor.org/rfc/rfc8610>.

[RFC9054] Schaad, J., "CBOR Object Signing and Encryption (COSE): Hash Algorithms", RFC 9054, DOI 10.17487/RFC9054, August 2022, <https://www.rfc-editor.org/rfc/rfc9054>.

[RFC9090] Bormann, C., "Concise Binary Object Representation (CBOR) Tags for Object Identifiers", RFC 9090, DOI 10.17487/RFC9090, July 2021, <https://www.rfc-editor.org/rfc/rfc9090>.

[RFC9581] Bormann, C., Gamari, B., and H. Birkholz, "Concise Binary Object Representation (CBOR) Tags for Time, Duration, and Period", RFC 9581, DOI 10.17487/RFC9581, August 2024, <https://www.rfc-editor.org/rfc/rfc9581>.

[STD94] Internet Standard 94, <https://www.rfc-editor.org/info/std94>. At the time of writing, this STD comprises the following:

Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", STD 94, RFC 8949, DOI 10.17487/RFC8949, December 2020, <https://www.rfc-editor.org/info/rfc8949>.

[STD96] Internet Standard 96, <https://www.rfc-editor.org/info/std96>. At the time of writing, this STD comprises the following:

Schaad, J., "CBOR Object Signing and Encryption (COSE): Structures and Process", STD 96, RFC 9052, DOI 10.17487/RFC9052, August 2022, <https://www.rfc-editor.org/info/rfc9052>.

Schaad, J., "CBOR Object Signing and Encryption (COSE): Countersignatures", STD 96, RFC 9338, DOI 10.17487/RFC9338, December 2022, <https://www.rfc-editor.org/info/rfc9338>.

[X.690] International Telecommunications Union, "Information technology — ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)", ITU-T Recommendation X.690, August 2015, <https://www.itu.int/rec/T-REC-X.690>.

8.2. Informative References

[I-D.ietf-lamps-csr-attestation] Ounsworth, M., Tschofenig, H., Birkholz, H., Wiseman, M., and N. Smith, "Use of Remote Attestation with Certification Signing Requests", Work in Progress, Internet-Draft, draft-ietf-lamps-csr-attestation-25, 5 May 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-lamps-csr-attestation-25>.

[I-D.ietf-rats-ar4si] Voit, E., Birkholz, H., Hardjono, T., Fossati, T., and V. Scarlata, "Attestation Results for Secure Interactions", Work in Progress, Internet-Draft, draft-ietf-rats-ar4si-10, 18 May 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-rats-ar4si-10>.

[I-D.ietf-rats-eat] Lundblade, L., Mandyam, G., O'Donoghue, J., and C. Wallace, "The Entity Attestation Token (EAT)", Work in Progress, Internet-Draft, draft-ietf-rats-eat-31, 6 September 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-rats-eat-31>.

[I-D.ietf-rats-reference-interaction-models] Birkholz, H., Eckel, M., Pan, W., and E. Voit, "Reference Interaction Models for Remote Attestation Procedures", Work in Progress, Internet-Draft, draft-ietf-rats-reference-interaction-models-17, 2 April 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-rats-reference-interaction-models-17>.

[I-D.ietf-scitt-architecture]
Birkholz, H., Delignat-Lavaud, A., Fournet, C., Deshpande, Y., and S. Lasker, "An Architecture for Trustworthy and Transparent Digital Supply Chains", Work in Progress, Internet-Draft, draft-ietf-scitt-architecture-22, 10 October 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-scitt-architecture-22>.

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

[IANA.cwt] IANA, "CBOR Web Token (CWT) Claims", <https://www.iana.org/assignments/cwt>.

[RFC9334] Birkholz, H., Thaler, D., Richardson, M., Smith, N., and W. Pan, "Remote ATtestation procedureS (RATS) Architecture", RFC 9334, DOI 10.17487/RFC9334, January 2023, <https://www.rfc-editor.org/rfc/rfc9334>.

[TCG-CoEvidence]
Trusted Computing Group, "TCG DICE Concise Evidence Binding for SPDM", Version 1.00, June 2023, <https://trustedcomputinggroup.org/wp-content/uploads/TCG-DICE-Concise-Evidence-Binding-for-SPDM-Version-1.0-Revision-53_1August2023.pdf>.

[TCG-TPM2] Trusted Computing Group, "Trusted Platform Module 2.0 Library Part 2: Structures", Version 184, March 2025, <https://trustedcomputinggroup.org/wp-content/uploads/Trusted-Platform-Module-2.0-Library-Part-2-Version-184_pub.pdf>.

Appendix A. Examples

The example in Figure 3 shows an Epoch Marker with an etime as the Epoch Marker type.

```
/ epoch-marker for
  1996-12-19T16:39:57-08:00[America//Los_Angeles][u-ca=hebrew] /
/ etime / 1001({
    1: 851042397,
  -10: "America/Los_Angeles",
  -11: { "u-ca": "hebrew" }
})
```

Figure 3: CBOR Epoch Marker based on ‘etime‘ (EDN)

The encoded data item in CBOR pretty-printed form (hex with comments) is shown in Figure 4.

```
d9 03e9                                 # tag(1001)
   a3                                   # map(3)
      01                                # unsigned(1)
      1a 32b9e05d                       # unsigned(851042397)
      29                                # negative(9)
      73                                # text(19)
         416d65726963612f4c6f735f416e67656c6573 # "America/Los_Angeles"
      2a                                # negative(10)
      a1                                # map(1)
         64                             # text(4)
            752d6361                    # "u-ca"
         66                             # text(6)
            686562726577                # "hebrew"
```

Figure 4: CBOR Epoch Marker based on ‘etime‘ (pretty hex)

The example in Figure 5 shows an Epoch Marker with an etime as the Epoch Marker type carried within a CWT.

```
=============== NOTE: '\' line wrapping per RFC 8792 ================

18([
  / protected / << {
    / alg / 1: -7 / ECDSA 256 /
  } >>,
  / unprotected / {},
  / payload / << {
    / epoch marker / 2000: / etime / 1001({
      1: 851042397,
      -10: "America/Los_Angeles",
      -11: { "u-ca": "hebrew" }
    }),
    / eat_nonce /  10: h'\
   c53a8c924f5a27877951ace250709aa64a45311840ca1c55da09af026a7a9c1c',
    / iss / 1 : "ACME epoch bell",
    / aud / 3 : "ACME protocol clients",
    / nbf / 5 : 1757929800,
    / exp / 4 : 1757929860
  } >>,
  / signature / h'737461747574617279'
])
```

Figure 5: CBOR Epoch Marker based on ‘etime‘ carried within a CWT (EDN)

The encoded data item in CBOR pretty-printed form (hex with comments) is shown in Figure 6.

```
================ NOTE: '\' line wrapping per RFC 8792 ================

d2                                      # tag(18)
   84                                   # array(4)
      43                                # bytes(3)
         a10126                         # "\xA1\u0001&"
      a0                                # map(0)
      58 88                             # bytes(136)
         \
a61907d0d903e9a3011a32b9e05d2973416d65726963612f4c6f735f416e67656c65\
732aa164752d6361666865627265770a5820c53a8c924f5a27877951ace250709aa6\
4a45311840ca1c55da09af026a7a9c1c016f41434d452065706f63682062656c6c03\
7541434d452070726f746f636f6c20636c69656e7473051a68c7e148041a68c7e18\
4 # "\xA6\u0019\a\xD0\xD9\u0003\xE9\xA3\u0001\u001A2\xB9\xE0])\
sAmerica/Los_Angeles*\xA1du-cafhebrew\nX \xC5:\x8C\x92OZ'\x87yQ\xAC\\
xE2Pp\x9A\xA6JE1\u0018@\xCA\u001CU\xDA\t\xAF\u0002jz\x9C\u001C\\
u0001oACME epoch bell\u0003uACME protocol clients\u0005\u001Ah\xC7\\
                                    xE1H\u0004\u001Ah\xC7\xE1\x84"
      49                                 # bytes(9)
         737461747574617279              # "statutary"
```

Figure 6: CBOR Epoch Marker based on 'etime' carried within a CWT (pretty hex)

A.1. RFC 3161 TSTInfo

As a reference for the definition of TST-info-based-on-CBOR-time-tag the code block below depicts the original layout of the TSTInfo structure from [RFC3161].

```
   TSTInfo ::= SEQUENCE  {
      version                      INTEGER  { v1(1) },
      policy                       TSAPolicyId,
      messageImprint               MessageImprint,
        -- MUST have the same value as the similar field in
        -- TimeStampReq
      serialNumber                 INTEGER,
       -- Time-Stamping users MUST be ready to accommodate integers
       -- up to 160 bits.
      genTime                      GeneralizedTime,
      accuracy                     Accuracy                 OPTIONAL,
      ordering                     BOOLEAN             DEFAULT FALSE,
      nonce                        INTEGER                  OPTIONAL,
        -- MUST be present if the similar field was present
        -- in TimeStampReq.  In that case it MUST have the same value.
      tsa                          [0] GeneralName          OPTIONAL,
      extensions                   [1] IMPLICIT Extensions   OPTIONAL  }
```

Acknowledgements

The authors would like to thank Carl Wallace, Jeremy O’Donoghue and Jun Zhang for their reviews, suggestions and comments.

Authors’ Addresses

Henk Birkholz
Fraunhofer SIT
Rheinstrasse 75
64295 Darmstadt
Germany
Email: henk.birkholz@ietf.contact

Thomas Fossati
Linaro
Switzerland
Email: Thomas.Fossati@linaro.org

Wei Pan
Huawei Technologies
Email: william.panwei@huawei.com

Ionu Mihalcea
Arm
United Kingdom
Email: ionut.mihalcea@arm.com

Carsten Bormann
Universitt Bremen TZI
Bibliothekstr. 1
D-28359 Bremen
Germany
Phone: +49-421-218-63921
Email: cabo@tzi.org