Remote ATtestation ProcedureS
Internet-Draft
Intended status: Informational
Expires: 30 May 2026

J. Beaney
Intel Corporation
Y. Deshpande
ARM Corporation
A. Draper
Altera Corporation
V. Scarlata
Intel Corporation
N. Smith
Independent
F. Chinchilla
Intel Corporation
26 November 2025

Intel Profile for Remote Attestation
draft-cds-rats-intel-corim-profile-06

Abstract

This document is a profile of various IETF and TCG standards that support remote attestation. The profile supports Intel-specific adaptations and extensions for Evidence, Endorsements and Reference Values. This profile describes a particular application of CoRIM, EAT, CMW, TCG concise evidence, and TCG DICE specifications. In particular, CoRIM is extended to define measurement types that are unique to Intel and defines Reference Values types that support matching Evidence based on range and subset comparison. Multiple Evidence formats are anticipated, based on IETF and TCG specifications. Evidence formats are mapped to Reference Values expressions based on CoRIM and CoRIM extensions found in this profile. The Evidence to Reference Values mappings are either documented by industry specifications or by this profile. Reference Value Providers and Endorsers may use this profile to author mainifests containing Reference Values and Endorsements that require Intel profile support from parser implementations. Parser implementations can recognize the Intel profile by profile identifier values contained within attestation conceptual mmessages and from profile parameters to media types or profile specific content format identifiers.

About This Document

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

The latest revision of this draft can be found at https://fchinchilla.github.io/draft-cds-rats-intel-corim-profile/draft-cds-rats-intel-corim-profile.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-cds-rats-intel-corim-profile/.

Discussion of this document takes place on the Remote ATtestation ProcedureS 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/fchinchilla/draft-cds-rats-intel-corim-profile.

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 30 May 2026.

Copyright Notice

Table of Contents

1. Introduction

This profile describes extensions and restrictions placed on Reference Values, Endorsements, and Evidence that support attestation capabilities of Intel products containing Intel(R) SGX(TM) or Intel(R) TDX(TM) technology, or Intel(R) products that contain DICE [DICE.engine] root of trust, DICE layers [DICE.layer], or modules that implement SPDM [DMTF.SPDM].

The CoRIM specifications [DICE.CoRIM] and [I-D.ietf-rats-corim] define a baseline schema for Reference Values and Endorsements that are the basis for the extensions defined by this profile. CoRIM is also a baseline for Evidence (as specified by DiceTcbInfo [DICE.Attest], concise evidence (CoEV) [TCG.CE], and Security Protocol and Data Model (SPDM) [DMTF.SPDM]). Having a common baseline schema for Reference Values, Endorsements, and Evidence helps ensure compatibility across a spectrum of implementations.

This profile defines extensions to CoRIM that support appraisal matching that is not strictly exact-match. For example it defines _sets_, _masks_, _time_, and _ranges_.

The baseline CoRIM, as defined by [DICE.CoRIM] is a subset of the Intel profile. Intel products that implement exclusively the baseline CoRIM do not need this profile. Implementations based on the Intel profile do not necessarily imply an association with Intel products.

This profile extends CoMID measurement-values-map, as defined by [DICE.CoRIM] (see also [I-D.ietf-rats-corim]), with measurement types that are unique to Intel products. Some measurement types are specific to Reference Values where multiple reference states may be included in reference manifests. Intel profile extensions use a CBOR tagged value that defines a comparison operator and operands that instruct Verifiers regarding subset, range, and masked values matching semantics. For example, a numeric operator 'greater-than' instructs the Verifier to match a numeric Evidence value if it is greater than a numeric range operand.

This profile follows the Verifier behavior defined by [DICE.CoRIM] and extends Verifier behavior to include operator-operand matching. If no operator is specified by Reference Values statements, the

Verifier defaults to baseline [DICE.CoRIM] matching semantics. If Evidence matches Reference Values and Endorsements apply, Endorsed Values may be added to the accetped claims set. When all Evidence and Endorsements are processed, the Verifier's set of accepted claims is available for Attestation Results computations. This profile doesn't define Attestation Results. Rather, an Attestation Results profile, such as [I-D.kdyxy-rats-tdx-eat-profile] may be referenced instead.

This profile is compatible with multiple Evidence formats, as defined by [DICE.Attest], [TCG.CE], and [DMTF.SPDM]. It describes considerations when mapping Evidence formats to CoRIM [DICE.CoRIM] that a Verifier may use when performig appraisals.

2. Conventions and Definitions

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.

The reader is assumed to be familiar with the terms defined in Section 4 of [RFC9334] and [I-D.ietf-rats-endorsements].

3. Background

Complex platforms may contain a variety of hardware components, several of which may contain a hardware root of trust. Each root of trust may anchor one or more layers [DICE.layer] resulting in multiple instances of attestation Evidence. Evidence may be integrity protected by digital signatures, such as certificates [DICE.Attest], tokens [RFC8392] or by a secure transport [DMTF.SPDM]. For example, a system bus may allow dynamically configured peripheral devices that have attestation capabilities. Confidential computing environments, such as SGX, may extend an initial boundary to include a peripheral, or a peer enclave, that together forms a network of trustworthy nodes that a remote attestation Verifier may need to appraise. Multiple Evidence blocks may be combined into a composite Evidence block [I-D.ietf-rats-msg-wrap] that is more easily conveyed. Complex platforms may have one or more lead Attester endpoints that communicate with a remote Verifier to convey composite Evidence. The composition of the complex platform is partially represented in the composite Evidence.

However, composite Evidence may not fully describe platform composition. A complex platform may consist of multiple subsystems, such as network adapters, storage controllers, memory controllers,

special purpose processors, etc. The various sub-subsystem components vendors may create hardware bills of material (HBOM) that describe sub-system composition. A complex platform vendor may assemble various sub-system components whose composition is described by a platform HBOM. Although CoRIM may be used to create HBOMs, use of this profile for HBOM creation is unanticipated.

Nevertheless, a complex system may contain multiple identical instances of sub-sytem components that produce identical Evidence blocks. Additionally, dynamic insertion or removal of a component may result in composite Evidence blocks that reflect this dynamism.

4. Profile Identifier

This profile applies to Reference Values from a CoRIM manifest that a Verifier uses to process Evidence.

Profile identifier structures are defined by CoRIM [I-D.ietf-rats-corim], EAT [I-D.ietf-rats-eat] and Concise Evidence (CoEV) [TCG.CE].

4.1. Intel Profile

The profile identifier for the Intel Profile is the OID:

{joint-iso-itu-t(2) country(16) us(840) organization(1) intel(113741) (1) intel-comid(16) profile(1)}

2.16.840.1.113741.1.16.1

4.2. Media Types, Content Formats, and CBOR Tags

This profile utilizes and/or defines the following media types:

* "application/eat+cwt"
* "application/eat+cwt;eat_profile=2.16.840.1.113741.1.16.1"
* "application/rim+cbor"
* "application/rim+cbor";profile=2.16.840.1.113741.1.16.1"
* "application/toc+cbor"
* "application/toc+cbor;profile=2.16.840.1.113741.1.16.1"
* "application/ce+cbor"

* "application/ce+cbor;profile=2.16.840.1.113741.1.16.1"

This profile utilizes and/or defines the following content format identifiers (C-F ID):

Content Format	C-F ID	TN Function
"application/eat+cwt"	263	1668547081
"application/eat+cwt;eat_profile=2.16.840.1.113741.1.16.1"	10005	1668556861
"application/toc+cbor"	10570	1668557428
"application/ce+cbor"	10571	1668557429
"application/toc+cbor;profile=2.16.840.1.113741.1.16.1"	10572	1668557430
"application/ce+cbor;profile=2.16.840.1.113741.1.16.1"	10573	1668557431

Table 1

This profile uses the following CBOR tags:

CBOR Tag	Description
501	Concise Reference Integrity Manifest - (CoRIM)
570	Concise Table of Contents - (CoTOC)
571	Concise Evidence - (CoEv)
60010	Numeric expression
60020	Set of digests expression
60021	Set of strings expression

Table 2

5. Attester Anatomy

Attesters implement DICE layering using an initial Attesting Environment, also called a Root of Trust (RoT), that collection claims about one or more Target Environments. A Target Environment may become an Attesting Environment for a subsequent Target Environment, and so forth. There may be more than one RoT in the same Attester.

Attesting Environments generate Evidence by signing collected claims using an Attestation Key. Environments may have other keys besides attestation keys. Keys can be regarded as claims that are collected and reported as Evidence. Keys can also be regarded as Target Environments that have measurements that are specific to the key.

Confidential computing environments are Target Environments that can dynamically request Evidence from an Attesting Environment agent. Such Evidence may also be referred to as a ’Quote’.

Each DICE layer may produce signed Evidence. Evidence formats include both signature and measurements formats. Signature formats may include a mix of X.509 certificates and EAT CWTs. Evidence measurements formats may include a mix of ASN.1 and CBOR, where ASN.1 uses DiceTcbInfo and related varients and CBOR uses concise evidence, and CMW. Multiple Evidence blocks may be bundled using CMW collections.

Target Environments (other than cryptographic keys) are primarily identified using OIDs from Intel’s OID arc (2.16.840.1.113741). Keys are identified using key identifiers, public key, or certificate digests as defined by $crypto-key-type-choice [I-D.ietf-rats-corim].

6. Evidence Profile

Evidence may be integrity protected in various ways including: certificates [RFC5280], SPDM transcript [DMTF.SPDM], and CBOR web token (CWT) [RFC8392]. Evidence contained in a certificate may be encoded using DiceTcbInfo and DiceTcbInfoSeq [DICE.Attest]. Evidence contained in an SPDM payload may be encoded using the SPDM Measurement Block [DMTF.SPDM]. Evidence may be formatted as concise-evidence [TCG.CE] which may be encapsulated by alias certificates, SPDM Measurement Manifests, or EAT tokens.

The DiceTcbInfo and SPDM Evidence formats can be translated to CoMID. The concise evidence format is native to CoMID. This profile documents evidence mapping from DiceTcbInfo and SPDM Measurement Block to CoMID, as defined by [DICE.CoRIM].

The CoMID extensions defined by this profile Section 8.3 are applied to concise-evidence so that Verifiers that support this profile can consistently apply a common schema across Evidence, Reference Values, and Endorsements.

6.1. Evidence Hierarchy

Evidence hierarchy refers to DICE layering where the platform bootstrap components double as Attesting Environments that collect measurements of the other bootstrap components (as Target Environments) until the quoting agent (e.g., SGX Quoting Enclave (QE), TDX Quoting TD (QTD)) is initialized. Tenant trusted execution environment (TEE) components can be dynamically loaded then request Evidence from its quoting agent. Quoting agents locally verify then sign measurments using the QTD / QE attestation key. A hierarchy of Evidence consisting of all the Evidence from a RoT to the tenant environment describes the Attester.

A complex device may have multiple roots of trust, such as [DICE.engine], each contributing an evidence hierarchy that results in several Evidence "chains", that together, constitute a complete Evidence hierarchy for the Attester device.

The Evidence hierarchy should form a spanning tree that contains all Attester Evidence. All Attesting Environments within the device produce the spanning tree. CoRIM manifests contain Reference Values for the spanning tree so that Verifiers do not assume the spanning tree is defined by Evidence. Note that a failure or comporomise within the Attester device could result in a portion of the spanning tree being omitted.

Evidence examples:

* A DICE certificate chain with a DiceTcbInfo extension, a DiceTcbInfoSeq extension, and a ConceptualMessageWrapper (CMW) [I-D.ietf-rats-msg-wrap] extension containing a CBOR-encoded tagged-concise-evidence.

* An SPDM alias intermediate certification chain containing a CMW extension, and an SPDM measurement manifest containing tagged-concise-evidence.

6.2. Concise Evidence

Concise evidence is a CDDL representation of Evidence [TCG.CE] that uses expressions from CoMID, which are a subset of CoRIM. See [DICE.CoRIM] and [I-D.ietf-rats-corim]. Evidence describes the actual state of the Attester. tagged-concise-evidence uses a CBOR tag (571) to identify concise-evidence [TCG.CE]. This profile uses concise-evicence in conceptual message wrappers [I-D.ietf-rats-msg-wrap] and EAT tokens [I-D.ietf-rats-eat] to encode Evidence. This profile extends concise-evidence by extending measurement-values-map.

7. Reference Values and Endorsements Profile

The CoRIM specifications [DICE.CoRIM] and [I-D.ietf-rats-corim] define a baseline schema for Reference Values and Endorsements in this profile. The profile defines extensions to CoRIM for measurement types that are not representable by CoRIM or are more conveniently represented. This profile doesn't require use of extensions when base capabilities will suffice.

7.1. Concise Module ID Tag (CoMID)

This profile uses concise-mid-tag in conceptual message wrappers [I-D.ietf-rats-msg-wrap] and CoRIMs. This profile extends concise-mid-tag by extending measurement-values-map. Several extensions define two forms, one for representing actual state which is used for Endorsements and Evidence. The other form is used to represent reference state which is used for Reference Values.

7.2. Raw Value Measurements

Raw value measurements encode vendor-defined values opaquely. However, the mkey value can add vendor-specific semantics when used with raw-value and name measurement types. Additionally, specific environment-map values can supply vendor-specific semantics to raw-value and name measurement types.

Environments that project vendor-specific semantics are as follows:

Envoronment Identifier	Value	Semantics
class-id:OID=2.16.840.1.113741.1.5.3.6.8	560(bytes)	device type

Table 3

8. CoRIM Extensions

The Intel Profile extends measurement-values-map which is used by Evidence, Reference Values, and Endorsed Values by defining code points from the negative integer range.

Reference Values extensions define types that can have multiple Reference Values that "match" a singleton Evidence value called "non-exact match" matching. Reference state expressions define non-exact-match matching semantics in terms of numeric ranges, time, sets, and masks.

8.1. Data Types

8.1.1. Masked Values

Masked values are a string of bytes (e.g., bstr) that may have a companion mask value. The mask indicates which bits in the value are ignored when doing bit-wise equivalency comparisons. Verifier matching applies the equivalency test, allowing dissimilar Evidence and Reference values to be considered equivalent even if the two values (Evidence and Reference) are dissimilar. Evidence typically does not supply a mask. A Reference Value may omit the mask if bit-wise equivalency is desired.

The $masked-value-type type choice can be either ~tagged-bytes or $raw-value-type-choice. Evidence might be encoded as ~tagged-bytes or tagged-bytes which omits a mask value, while Reference Values of type tagged-masked-raw-value includes the mask value.

The Verifier MUST ensure the lengths of values and mask are equivalent. If the mask is shorter than the longest value, the mask is appended with zeros (0) until it is the same length as the longest value, either Evidence or Reference Value. If the mask is longer than the longest value, the mask is truncated to the length of the longest value. All values are evaluated from left to right (big-endian) applying bit-wise comparisons.

The masked value data types are as follows:

```
$masked-value-type /= ~tagged-bytes
$masked-value-type /= $raw-value-type-choice
```

8.2. Expressions

Expressions can be used with Reference Values or Endorsement conditions. Matching is applied using an operator and operands. There are two types of operators, numeric: such as greater-than or less-than, and sets: such as set membership.

Expressions are an array containing an operator followed by zero or more operands. The operator definition identifies the additional operands and their data types. A Verifier forms an expression using Evidence as the first operand and obtains the operator from the first entry in the expression array.

This profile describes operations using _infix_ notation where the first operand, _operand_1_, is obtained from Evidence, followed by the operator, followed by any remaining operands: _operand_2_, _operand_3_..., taken from Reference Values.

Expressions statements are CBOR tagged to indicate the values following the CBOR tag are to be evaluated as an expression equation. Expression statements found in Reference Values informs the Verifier that Evidence is needed to complete the expression equation.

Expressions are CBOR tagged to disambiguate the type of expression. See Section 12.

For example:

* #6.CBOR_Tag([operator, operand_2, operand_3, ...]).

Appraisal processing MUST evaluate expression equations to comply with this profile.

8.2.1. Expression Operators

There are three CBOR tagged operators as follows:

1. *60010*: A numeric expression with a numeric operator (Section 8.2.2) followed by a numeric operand: integer, unsigned integer, or floading point.

2. *60020*: A set of digests operator (Section 8.2.3) followed by a set of digests operand which is an array of digests.

3. *60021*: A set of strings operator (Section 8.2.3) followed by a set of strings operand which is an array of tstr.

The position of items in a set is not significant.

8.2.1.1. Equivalence Operator

By default, _exact_ match rules are assumed. Consequently, no operator artifact is needed when Evidence values are identical to Reference Values.

8.2.2. Numeric Expressions

Numeric expressions consist of an Evidence operand (Evidence_Operand) and an array containing a numeric operator and a numeric operand (Reference_Operand).

Numeric operators apply to values that are integers, unsigned integers or floating point numbers. There are four numeric operators:

1. *equal* (op.eq),

2. *greater-than* (op.gt),

3. *greater-than-or-equal* (op.ge),

4. *less-than* (op.lt),

5. *less-than-or-equal* (op.le).

Equivalence semantics can be achieved without using an expression with the op.eq operator by using the same data type for both Evidence and Reference Value.

The numeric operator data type definitions are as follows:

```
numeric-type = integer / unsigned / float
numeric-operators /= op.gt
numeric-operators /= op.ge
numeric-operators /= op.lt
numeric-operators /= op.le
numeric-expression = [ numeric-operators, numeric-type ]
tagged-numeric-expression = #6.60010(numeric-expression)
```

Evidence and Reference Values MUST be the same numeric type. For example, if a Reference Value numeric type is integer, then the Evidence numeric value must also be integer.

This profile defines four numeric expressions, one for each numeric operator:

* tagged-numeric-gt,

* tagged-numeric-ge,

* tagged-numeric-lt,

* tagged-numeric-le.

In each case, the numeric operator is used to evaluate a Reference Value operand against an Evidence value operand.

The expression is evaluated using _infix_ notation where Evidence_Operand is the left-hand-side of the numeric operator and the Reference_Operand is the right-hand-side.

Example:

* The expression: (7op.le9) evaluates to TRUE.

The numeric type definition is as follows:

```
tagged-numeric-gt = #6.60010( [
    op.gt .within numeric-operators,
    reference-value: numeric-type ] )
tagged-numeric-ge = #6.60010( [
    op.ge .within numeric-operators,
    reference-value: numeric-type ] )
tagged-numeric-lt = #6.60010( [
    op.lt .within numeric-operators,
    reference-value: numeric-type ] )
tagged-numeric-le = #6.60010( [
    op.le .within numeric-operators,
    reference-value: numeric-type ] )
```

8.2.3. Set Expressions

Set expressions consist of an Evidence operand (Evidence_Operand) and an array containing a set operator and a set operand (Reference_Set_Operand).

Sets have two operators:

* *op.mem* - operand_1 is a member of the set operand_2.

* *op.nmem* - operand_1 is NOT a member of the set operand_2.

Example:

* The expression: ("fox"op.mem["cat", "dog", "fox"]) evaluates to TRUE.

The set type is as follows:

```
set-operators /= op.mem
set-operators /= op.nmem
set-type<T> = [ * T ]

set-digest-type /= set-type<digest>
set-digest-expression = [ set-operators, set-digest-type ]
tagged-set-digest-expression = #6.60020( set-digest-expression )

set-tstr-type /= set-type<tstr>
set-tstr-expression = [ set-operators, set-tstr-type ]
tagged-set-tstr-expression = #6.60021( set-tstr-expression )
```

The set expression array contains a set operator followed by an array of values which are the members of a set of Reference Values. The set is defined by set-type.

The set expression definitions are as follows:

```
tagged-exp-digest-member = #6.60020([
    op.mem .within set-operators, set-digest-type ])

tagged-exp-digest-not-member = #6.60020([
    op.nmem .within set-operators, set-digest-type ])

tagged-exp-tstr-member = #6.60021([
    op.mem .within set-operators, set-tstr-type ])

tagged-exp-tstr-not-member = #6.60021([
    op.nmem .within set-operators, set-tstr-type ])
```

The Evidence_Operand MUST NOT be nil.

The Reference_Set_Operand MAY be the empty set - e.g. [ ].

8.3. Measurement Extensions

This profile extends the CoMID measurement-values-map with additional code point definitions, that accommodate Intel SGX and similar architectures. Measurement extensions don’t change Verifier behavior. An extension enables the Verifier to validate the profile compliance of the input evidence and reference values, as it defines the acceptable data types in evidence and the expression operator that is explicitly supplied with the Reference Values, see Section 8.2.1.

In cases where Evidence does not exactly match Reference Values, the operator definition determines the expected data types of the operands. Expected Verifier behavior is defined in Section 9

The measurement extensions that follow are assumed to be appraised according to the appriasal steps described in Section 8.1 of [I-D.ietf-rats-corim].

8.3.1. The tee.advisory-ids Measurement Extension

The tee.advisory-ids extension enables Attesters to report known security advisories and for Reference Values Providers (RVP) to assert updated security advisories. It can also be used by Endorsers to assert security advisory information through conditional endorsement.

The $tee-advisory-ids-type is used to specify a set of security advisories, where each identifier is represented using a string. Evidence may report a set of advisories the Attester believes are relevant. The set of advisories are constrained by the set-tstr-type structure.

As a Reference Value expression, an empty set can be used to signify that no outstanding advisories are expected. If the Evidence also contains the empty set then the Reference corroborates the Evidence.

The $tee-advisory-ids-type is a list of strings, each identifying a single security advisory. When used with Evidence the set-tstr-type type is used. When used with Reference Values or Endorsements the set-tstr-type, tagged-exp-tstr-member, or tagged-exp-tstr-not-member types can be used.

```
$$measurement-values-map-extension //= (
  &(tee.advisory-ids: -89) => $tee-advisory-ids-type
)
$tee-advisory-ids-type /= set-tstr-type
$tee-advisory-ids-type /= tagged-exp-tstr-not-member
$tee-advisory-ids-type /= tagged-exp-tstr-member
```

8.3.1.1. The tee-advisory-ids-type Comparison Algorithm

The comparison algorithm for tee-advisory-ids-type is used when Endorsement or Reference Values triples conditions are matched with an Environment Claims Tuple (ECT) in the Verifier’s Accepted Claims Set (ACS). The triple condition containing a tee-advisory-ids-type Claim matches an ACS ECT according to the comparison algorithm for set of strings as defined in Section 9.2.

8.3.2. The tee.attributes Measurement Extension

The tee.attributes extension enables the Attester to report TEE attributes and an RVP to assert a reference TEE attributes and mask.

The $tee-attributes-type is used to specify TEE attributes in 8 or 16 byte words. If Evidence uses an 8 byte mask, then the Reference Values expression also uses an 8 byte value and mask.

The $tee-attributes-type is a singleton value omitting the mask value when used as Endorsement or Evidence and a tuple containing the reference and mask when used as a Reference Value.

```
$$measurement-values-map-extension //= (
  &(tee.attributes: -82) => $tee-attributes-type
)
$tee-attributes-type /= $masked-value-type
```

Alternatively, the TEE attributes may be encoded using mkey where mkey contains the non-negative tee.attributes and mval.raw-value contains the $tee-attributes-type.mask-type value.

8.3.3. The tee.cryptokeys Measurement Extension

The tee.cryptokeys extension identifies cryptographic keys associated with a Target Environment. If multiple $crypto-key-type-choice measurements are supplied, array position disambiguates each entry. Appraisal compares values indexed by array position.

```
$$measurement-values-map-extension //= (
  &(tee.cryptokeys: -91) => [ + $tee-cryptokey-type ]
)
$tee-cryptokey-type /= $crypto-key-type-choice
```

Alternatively, the TEE cryptokeys may be encoded using mkey where mkey contains the non-negative tee.cryptokeys and mval.cryptokeys contains the $tee-cryptokey-type value.

8.3.4. The tee.tcbdate Measurement Extension

The tee.tcbdate (code point -72) extension is used by Endorsers to assert validity of a TEE component. For example, a conditional endorsement might locate a component based on a few expected Claims, then augment them with a tee.tcbdate Claim.

The $tee-date-type can be expressed in several ways:

* ISO 8601 strings of the form YYYY-MM-DDTHH:MM:SSZ.

* POSIX time which is the number of seconds since January 1, 1970 (midnight UTC).

* RFC9581 etime [RFC9581].

```
$$measurement-values-map-extension //= (
  &(tee.tcbdate: -72) => $tee-date-type
)
$tee-date-type /= ~tdate
$tee-date-type /= tdate
$tee-date-type /= time
$tee-date-type /= etime ; RFC9581
$tee-date-type /= period ; RFC9581
```

~tdate strings must be converted to a numeric value (i.e.,~time) before operations involving time are applied.

Alternatively, tee.tcbdate may be encoded using mkey where mkey contains the non-negative code point value and where mval.name contains the string representation $tee-date-type without the CBOR tag (i.e., ~tdate - see Section 3.7 [RFC8610]).

8.3.5. The tee.mrtee and tee.mrsigner Measurement Extension

The tee.mrtee extension enables an Attester to report digests for the SGX enclave or TDX TD (e.g., MRENCLAVE, MRTD). The tee.mrsigner extension enables an Attester to report the signer of the TEE digest (e.g., MRSIGNER).

The $tee-digest-type has multiple type structures involving digest values. A singleton digest has a hash algorithm identifier and the digest value. When used as Evidence, either a signleton digest or a set of digests can be reported. When used as Reference Values or Endorsements, a set of digests can be asserted signifying equivalence matching. Alternatively, matching may be expressed as set membership or set difference expressions.

```
$$measurement-values-map-extension //= (
  &(tee.mrtee: -83) => $tee-digest-type
)
$$measurement-values-map-extension //= (
  &(tee.mrsigner: -84) => $tee-digest-type
)
$tee-digest-type /= digest ; see corim
$tee-digest-type /= digests-type ; see corim
;$tee-digest-type /= set-digest-type
$tee-digest-type /= tagged-exp-digest-member
$tee-digest-type /= tagged-exp-digest-not-member
```

Alternatively, the TEE digests may be encoded using mkey where mkey contains the non-negative tee.mrtee or tee.mrsigner and mval.digests contains a digests-type value.

8.3.5.1. The tee-digest-type Comparison Algorithm

The comparison algorithm for tee-digest-type is used when the condition statement in an Endorsement or Reference Values triple is matched with an Environment Claim Tuple (ECT) from the Verifier's Accepted Claims Set (ACS). The comparison algorithm for set of digests is defined in Section 9.2.

8.3.6. The tee.platform-instance-id Measurement Extension

Platform Instance ID is a globally unique identifier generated by the platform during Platform Establishment. This value remains consistent across trusted computing base (TCB) recoveries, but is regenerated during Platform Establishment due to desire to reset keys or to add and remove hardware. See (Section 3.7 [INTEL.DCAP]).

The tee.platform-instance-id extension enables the Attester to report the platform instance identifier as an Evidence value and the RVP to assert an exact-match Reference Value.

The $tee-platform-instance-id-type is a bstr.

```
$$measurement-values-map-extension //= (
  &(tee.platform-instance-id: -101) => $tee-platform-instance-id-type
)
$tee-platform-instance-id-type /= bstr
```

Alternatively, the platform instance ID may be encoded using mkey where mkey contains the non-negative tee.platform-instance-id code point and mval.raw-value contains the $tee-platform-instance-id-type value.

8.3.7. The tee.isvprodid Measurement Extension

The tee.isvprodid extension enables the Attester to report the ISV product identifier Evidence value and the RVP to assert an exact-match Reference Value.

The $tee-isvprodid-type is an unsigned integer.

The $tee-isvprodid-type is an exact match measurement.

```
$$measurement-values-map-extension //= (
  &(tee.isvprodid: -85) => $tee-isvprodid-type
)
$tee-isvprodid-type /= uint
$tee-isvprodid-type /= bstr
```

Alternatively, the TEE product ID may be encoded using mkey where mkey contains the non-negative tee.isvprodid and mval.raw-value contains the $tee-isvprodid-type value.

8.3.8. The tee.miscselect Measurement Extension

The tee.miscselect extension enables the Attester to report the (TBD:miscselect-description) Evidence value and the RVP to assert a Reference Value and mask.

The $tee-miscselect-type is a 4 byte value and mask.

The $tee-miscselect-type is a singleton mask-type value when used as Endorsement or Evidence and a tagged-masked-raw-value when used a Reference Value.

```
$$measurement-values-map-extension //= (
  &(tee.miscselect: -81) => $tee-miscselect-type
)
$tee-miscselect-type /= $masked-value-type
```

Alternatively, the TEE miscselect may be encoded using mkey where mkey contains the non-negative tee.miscselect and mval.raw-value contains the measurement value and mval.raw-value-mask‘ contains the mask value.

8.3.9. The tee.model Measurement Extension

The tee.model extension enables the Attester to report the TEE model string as Evidence and the RVP to assert an exact-match Reference Value.

The $tee-model-type is a string.

The $tee-model-type is an exact match measurement.

```
$$measurement-values-map-extension //= (
  &(tee.model: -71) => $tee-model-type
)
$tee-model-type /= tstr
```

Alternatively, the TEE model may be encoded using mkey where mkey contains the non-negative tee.model and mval.name contains the $tee-model-type value.

8.3.10. The tee.pceid Measurement Extension

The tee.pceid extension enables the Attester to report the PCEID as Evidence and the RVP to assert an exact-match Reference Value.

The $tee-pceid-type is a string or a uint. As string, PCEID is a four character decimal value such as "0000".

The $tee-pceid-type is an exact match measurement.

```
$$measurement-values-map-extension //= (
  &(tee.pceid: -80) => $tee-pceid-type
)
$tee-pceid-type /= tstr
$tee-pceid-type /= uint
```

Alternatively, the PCEID may be encoded using mkey where mkey contains the non-negative tee.pceid and mval.name (code point 11) contains the string representation. Or, mval.raw-int (code point 15) contains the integer representation.

8.3.11. The tee.isvsvn Measurement Extension

The tee.isvsvn extension enables the Attester to report the SVN for the independent software vendor supplied component as Evidence and the RVP to assert a Reference Value that is greater-than-or-equal to the reported SVN.

The $tee-svn-type is either an unsigned integer when reported as Evidence, or a tagged numeric expression that contains an SVN and a numeric greater-than-or-equal operator. The Verifier ensures the Evidence value is greater-that-or-equal to the Reference Value.

The $tee-svn-type is a svn-type when used as Endorsement or Evidence and a tagged-numeric-expression when used as a Reference Value.

```
$$measurement-values-map-extension //= (
  &(tee.isvsvn: -73) => $tee-svn-type
)
$tee-svn-type /= svn-type .within numeric-type
$tee-svn-type /= tagged-numeric-ge
$tee-svn-type /= tagged-int-range
$tee-svn-type /= tagged-min-svn
```

Alternatively, the TEE isvsvn may be encoded using mkey where mkey contains the non-negative tee.isvsvn and mval.svn contains the svn value as svn-type.

8.3.12. The tee.tcb-comp-svn Measurement Extension

The tee.tcb-comp-svn extension enables the Attester to report an array of SVN values for the TCB when asserted as Evidence and an array of tagged-numeric-ge entries when asserted as a Reference Value.

The $tee-tcb-comp-svn-type is an array containing 16 SVN values when reported as Evidence and an array of 16 expression records each containing the numeric ge operator and a reference SVN value. The Verifier evaluates each SVN in the Evidence array with the corresponding reference expression, by array position. If all Evidence values match their respective expressions, evaluation is successful. The array of SVN Evidence is accepted.

```
$$measurement-values-map-extension //= (
  &(tee.tcb-comp-svn: -125) => $tee-tcb-comp-svn-type
)
$tee-tcb-comp-svn-type /= [ 16*16 svn-type .within numeric-type ]
$tee-tcb-comp-svn-type /= [ 16*16 tagged-numeric-ge ]
$tee-tcb-comp-svn-type /= [ 16*16 tagged-int-range ]
$tee-tcb-comp-svn-type /= [ 16*16 tagged-min-svn ]
```

8.3.13. The tee.tcb-eval-num Measurement Extension

The tee.tcb-eval-num extension enables the Attester to report a TCB evaluation number as Evidence and the RVP to assert a Reference Value expression that compares the tcb-eval-num Evidence with the Reference Value using the greater-than-or-equal operator.

The $tee-tcb-eval-num-type is an unsigned integer when reported as Evidence and a tagged numeric expression when asserted as Reference Values.

```
$$measurement-values-map-extension //= (
  &(tee-tcb-eval-num: -86) => $tee-tcb-eval-num-type
)
$tee-tcb-eval-num-type /= uint .within numeric-type
$tee-tcb-eval-num-type /= tagged-numeric-ge
$tee-tcb-eval-num-type /= tagged-int-range
```

Alternatively, the TEE tcb-eval-num Evidence may be encoded using mkey where mkey contains the non-negative tee.tcb-eval-num and mval.raw-value contains the tcb-eval-num encoded as 4-byte bstr value.

8.3.14. The tee.tcb-status Measurement Extension

The tee.tcb-status extension enables Attesters to report the status of the TEE trusted computing base (TCB) and for Reference Value Providers (RVP) to assert expected TCB status. It can also be used by Endorsers to assert TCB status through conditional endorsement.

The tee-tcbstatus-type is used to specify TCB status as a set of status strings or as an expression with a set membership operator.

The $tee-tcbstatus-type is a status array containing strings describing TCB status values. When describing Evidence the set-tstr-type type is used. When describing Reference Values or Endorsements the set-tstr-type, tagged-exp-tstr-member, or tagged-exp-tstr-not-member types can be used.

```
$$measurement-values-map-extension //= (
  &(tee.tcbstatus: -88) => $tee-tcbstatus-type
)
$tee-tcbstatus-type /= set-tstr-type
$tee-tcbstatus-type /= tagged-exp-tstr-member
$tee-tcbstatus-type /= tagged-exp-tstr-not-member
```

8.3.14.1. The tee-tcbstatus-type Comparison Algorithm

The comparison algorithm for tee-tcbstatus-type is used when Endorsement or Reference Values triples conditions are matched with an Environment Claims Tuple (ECT) in the Verifier’s Accepted Claims Set (ACS). The triple condition containing a tee-tcbstatuss-type Claim matches an ACS ECT according to the comparison algorithm for set of strings as defined in Section 9.2.

8.3.15. The tee.vendor Measurement Extension

The tee.vendor extension enables the Attester to report the TEE vendor name as Evidence and for the RVP to assert the TEE vendor name.

The $tee-vendor-type is a string containing the vendor name as a string. The vendor string in Evidence must exactly match the vendor string in Reference Values.

The $tee-vendor-type is an exact match measurement.

```
$$measurement-values-map-extension //= (
  &(tee.vendor: -70) => $tee-vendor-type
)
$tee-vendor-type /= tstr
```

Alternatively, the TEE vendor may be encoded using mkey where mkey contains the non-negative tee.vendor and mval.name contains the $tee-vendor-type value.

9. Appraisal Algorithm

The Intel profile anticipates appraisal algorithms will be based on the appraisal algorithm defined in [I-D.ietf-rats-corim]. This profile extends the appraisal algorithm to recognize profile extensions that form equations. An Evidence measurement forms one of the operands: (evidence operand). A Reference Value forms the operator and remaining operands:

* [expression operator, reference value operand, etc...]

For example, if a numeric Reference Value is 14, and the expressions operator is gt the Reference Value might contain the Claim: #6.60010([1, 14]). Given Evidence contains the value: 15. The in-fix construction of the equation would be: 15 gt 14. The Verifier evaluates whether 15 is greater-than 14.

9.1. Complex Expressions

Complex expressions can be used to assess whether the Target Environment is in a particular state before certain Endorsement claims can be asserted. For example, if an SGX enclave has an svn value that is less than the prescribed minimum svn, the enclave status may be considered "OutOfDate" or may have a known security advisory. The CoMID conditional-endorsement-triples or conditional-endorsement-series-triples describe complex Endorsement expressions.

This profile uses these triples with the reference measurement values extensions described in Section 8.3.

9.2. Comparison Algorithm for Sets

The comparison algorithm for sets describes set equivalence, set membership, and set difference (not membership). The Verifier’s Accepted Claims Set (ACS) contains a list of Environment Claims Tuples (ECT)[I-D.ietf-rats-corim]. The condition ECTs are compared to ACS ECTs based on this comparison algorithm.

The set comparison algorithm processes sets of strings and sets of digests.

9.2.1. Comparison Algorithm for Set of Strings

There are three string set representations: set-tstr-type, tagged-exp-tstr-member, and tagged-exp-tstr-not-member.

* set-tstr-type - Every string in the condition set-tstr-type MUST match a string in the ACS.ECT.element-map.element-claims.set-tstr-type set. The string position in the array is not significant. The ACS.ECT.element-map.element-claims.set-tstr-type set MUST be equivalent to the condition set-tstr-type set (i.e., the two sets have the same cardinality and the same set members).

* tagged-exp-tstr-member - The condition ECT set operator MUST equal member and every string in the condition set-tstr-type MUST match a string in the ACS.ECT.element-map.element-claims.set-tstr-type set. The string position in the array is not significant. The ACS.ECT.element-map.element-claims.set-tstr-type set MAY contain strings not found in the condition set-tstr-type.

* tagged-exp-tstr-not-member - The condition ECT set operator MUST equal not-member and every string in the condition set-tstr-type MUST NOT match a string in the ACS.ECT.element-map.element-claims.set-tstr-type set. The string position in the array is not significant.

9.2.2. Comparison Algorithm for Set of Digests

There are five digest set representations: digest, digest-type, set-digest-type, tagged-exp-digest-member, and tagged-exp-digest-not-member.

* digest - The singleton digest in the condition MUST match at least one digest in the ACS.ECT.element-map.element-claims.set-digest-type set.

* digest-type and set-digest-type - Every digest in the condition digest-type or set-digest-type MUST match a digest in the ACS.ECT.element-map.element-claims.set-digest-type set. The digest position in the array is not significant. The ACS.ECT.element-map.element-claims.set-digest-type set MUST be equivalent to the condition set-digest-type set (i.e., the two sets have the same cardinality and the same set members). Matching based on the empty set is permitted when the set-digest-type is used.

* tagged-exp-digest-member - The condition ECT set operator MUST equal member and every digest in the condition set-digest-type MUST match a digest in the ACS.ECT.element-map.element-claims.set-digest-type set. The digest position in the array is not significant. The ACS.ECT.element-map.element-claims.set-digest-type set MAY contain digests not found in the condition set-digest-type.

* tagged-exp-digest-not-member - The condition ECT set operator MUST equal not-member and every digest in the condition set-digest-type MUST NOT match a digest in the ACS.ECT.element-map.element-claims.set-digest-type set. The digest position in the array is not significant.

10. Reporting Attestation Results

Attestation verification can be performed by a pipeline consisting of multiple stages where each input manifest demarks a stage. The final stage prepares Attestation Results according to Relying Party specifications. This profile does not define an attestation results format. The Relying Party should specify suitable Attestation Results formats such as [I-D.ietf-rats-ar4si] or [I-D.kdyxy-rats-tdx-eat-profile].

The precise Attestation Results format used, if negotiated by Verifier and Relying Party, should reference this profile to acknowledge that the Relying Party and Verifier both support the extensions defined in this document.

11. Security Considerations

The security of this profile depends on the security considerations of the various normative references.

12. IANA Considerations

IANA has allocated the following tags in the CBOR Tags registry [IANA.cbor-tags].

Tag #	Data Item	Semantics	Reference
60010	array	Contains a numeric expression, see Section 8.2.2	RFCthis
60020	array	Contains a set of digest expression, see Section 8.2.3	RFCthis
60021	array	Contains a set of tstr expression, see Section 8.2.3	RFCthis

Table 4

13. References

13.1. Normative References

[DICE.Attest]
Trusted Computing Group (TCG), "DICE Attestation Architecture", Version 1.1, Revision 18 , January 2024, <https://trustedcomputinggroup.org/wp-content/uploads/DICE-Attestation-Architecture-Version-1.1-Revision-18_pub.pdf>.

[DICE.CoRIM]
Trusted Computing Group (TCG), "DICE Endorsement Architecture for Devices", Version 1.0, Revision 0.38 , November 2022, <https://trustedcomputinggroup.org/wp-content/uploads/TCG-Endorsement-Architecture-for-Devices-V1-R38_pub.pdf>.

[DICE.layer]
Trusted Computing Group (TCG), "DICE Layering Architecture", Version 1.0, Revision 0.19 , July 2020, <https://trustedcomputinggroup.org/wp-content/uploads/DICE-Layering-Architecture-r19_pub.pdf>.

[I-D.ietf-rats-corim]
Birkholz, H., Fossati, T., Deshpande, Y., Smith, N., and W. Pan, "Concise Reference Integrity Manifest", Work in Progress, Internet-Draft, draft-ietf-rats-corim-09, 20 October 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-rats-corim-09>.

[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-msg-wrap]
Birkholz, H., Smith, N., Fossati, T., Tschofenig, H., and D. Glaze, "RATS Conceptual Messages Wrapper (CMW)", Work in Progress, Internet-Draft, draft-ietf-rats-msg-wrap-21, 18 November 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-rats-msg-wrap-21>.

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

[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>.

[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>.

[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>.

[TCG.CE] Trusted Computing Group (TCG), "TCG DICE Concise Evidence Binding for SPDM", Version 1.0, Revision 0.54 , January 2024, <https://trustedcomputinggroup.org/wp-content/uploads/TCG-DICE-Concise-Evidence-Binding-for-SPDM-Version-1.0-Revision-54_pub.pdf>.

13.2. Informative References

[DICE.engine]
Trusted Computing Group (TCG), "Requirements for a Device Identifier Composition Engine", Family "2.0", Level 00 Revision 78 , March 2018, <https://trustedcomputinggroup.org/wp-content/uploads/Hardware-Requirements-for-Device-Identifier-Composition-Engine-r78_For-Publication.pdf>.

[DMTF.SPDM]
Distributed Managability Task Force (DMTF), "Security Protocol and Data Mmodel (SPDM) Specification", Version 1.2.1 , May 2022, <https://www.dmtf.org/sites/default/files/standards/documents/DSP0274_1.2.1.pdf>.

[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-09, 15 August 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-rats-ar4si-09>.

[I-D.ietf-rats-endorsements]
Thaler, D., Birkholz, H., and T. Fossati, "RATS Endorsements", Work in Progress, Internet-Draft, draft-ietf-rats-endorsements-07, 25 August 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-rats-endorsements-07>.

[I-D.kdyxy-rats-tdx-eat-profile]
Kostal, G., Dittakavi, S., Yeluri, R., Xia, H., and J. Yu, "EAT profile for Intel(r) Trust Domain Extensions (TDX) attestation result", Work in Progress, Internet-Draft, draft-kdyxy-rats-tdx-eat-profile-02, 13 December 2024, <https://datatracker.ietf.org/doc/html/draft-kdyxy-rats-tdx-eat-profile-02>.

[INTEL.DCAP]
Intel Corporation, "Intel(R) Software Guard Extensions (Intel(R) SGX) Data Center Attestation Primitives ECDSA Quote Library API", August 2023, <https://download.01.org/intel-sgx/latest/dcap-latest/linux/docs/Intel_SGX_ECDSA_QuoteLibReference_DCAP_API.pdf>.

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

[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>.

[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>.

Appendix A. Acknowledgments

The authors wish to thank Shanwei Cen, Piotr Zmijewski, and Dionna Amalie Glaze for their valuable contributions.

Appendix B. Full Intel Profile CDDL

```
; This cddl file depends on these cddl files: coev.cddl corim-autogen.cddl

tagged-numeric-gt = #6.60010( [
    op.gt .within numeric-operators,
    reference-value: numeric-type ] )
tagged-numeric-ge = #6.60010( [
    op.ge .within numeric-operators,
    reference-value: numeric-type ] )
tagged-numeric-lt = #6.60010( [
    op.lt .within numeric-operators,
    reference-value: numeric-type ] )
tagged-numeric-le = #6.60010( [
    op.le .within numeric-operators,
    reference-value: numeric-type ] )

numeric-type = integer / unsigned / float
numeric-operators /= op.gt
numeric-operators /= op.ge
numeric-operators /= op.lt
numeric-operators /= op.le
numeric-expression = [ numeric-operators, numeric-type ]
tagged-numeric-expression = #6.60010(numeric-expression)

Etime = #6.1001(etime-detailed)

etime-framework = {
  uint => any ; at least one base time
  * (nint/text) => any ; elective supplementary information
  * uint => any ; critical supplementary information
}
```

```
   etime-detailed = ({
     $$ETIME-BASETIME
     ClockQuality-group
     * $$ETIME-ELECTIVE
     * $$ETIME-CRITICAL
     * ((nint/text) .feature "etime-elective-extension") => any
     * (uint .feature "etime-critical-extension") => any
   }) .within etime-framework

   $$ETIME-BASETIME //= (1: ~time)
   $$ETIME-BASETIME //= (4: ~decfrac)
   $$ETIME-BASETIME //= (5: ~bigfloat)
   $$ETIME-ELECTIVE //= (-3: uint)
   $$ETIME-ELECTIVE //= (-6: uint)
   $$ETIME-ELECTIVE //= (-9: uint)
   $$ETIME-ELECTIVE //= (-12: uint)
   $$ETIME-ELECTIVE //= (-15: uint)
   $$ETIME-ELECTIVE //= (-18: uint)
   $$ETIME-ELECTIVE //= (-1 => $ETIME-TIMESCALE)
   $$ETIME-ELECTIVE //= (-13 => $ETIME-TIMESCALE)
   $$ETIME-CRITICAL //= (13 => $ETIME-TIMESCALE)
   $ETIME-TIMESCALE /= &(etime-utc: 0)
   $ETIME-TIMESCALE /= &(etime-tai: 1)

   ClockQuality-group = (
     ? &(ClockClass: -2) => uint .size 1 ; PTP/RFC8575
     ? &(ClockAccuracy: -4) => uint .size 1 ; PTP/RFC8575
     ? &(OffsetScaledLogVariance: -5) => uint .size 2 ; PTP/RFC8575
     ? &(Uncertainty: -7) => ~time/~duration
     ? &(Guarantee: -8) => ~time/~duration
   )

   Duration = #6.1002(etime-detailed)

   simple-Period = #6.1003([
     start: ~Etime / null
     end: ~Etime / null
     ? duration: ~Duration
   ])

   Period = #6.1003([
     (start: ~Etime,
      ((end: ~Etime) //
       (end: null,
        duration: ~Duration))) //
     (start: null,
      end: ~Etime,
      duration: ~Duration)
```

```
   ])

   etime = #6.1001({* (int/tstr) => any})
   duration = #6.1002({* (int/tstr) => any})
   period = #6.1003([~etime/null, ~etime/null, ?~duration])

   set-operators /= op.mem
   set-operators /= op.nmem
   set-type<T> = [ * T ]

   set-digest-type /= set-type<digest>
   set-digest-expression = [ set-operators, set-digest-type ]
   tagged-set-digest-expression = #6.60020( set-digest-expression )

   set-tstr-type /= set-type<tstr>
   set-tstr-expression = [ set-operators, set-tstr-type ]
   tagged-set-tstr-expression = #6.60021( set-tstr-expression )

   tagged-exp-digest-member = #6.60020([
       op.mem .within set-operators, set-digest-type ])

   tagged-exp-digest-not-member = #6.60020([
       op.nmem .within set-operators, set-digest-type ])

   tagged-exp-tstr-member = #6.60021([
       op.mem .within set-operators, set-tstr-type ])

   tagged-exp-tstr-not-member = #6.60021([
       op.nmem .within set-operators, set-tstr-type ])

   $masked-value-type /= ~tagged-bytes
   $masked-value-type /= $raw-value-type-choice

   $$measurement-values-map-extension //= (
     &(tee.advisory-ids: -89) => $tee-advisory-ids-type
   )
   $tee-advisory-ids-type /= set-tstr-type
   $tee-advisory-ids-type /= tagged-exp-tstr-not-member
   $tee-advisory-ids-type /= tagged-exp-tstr-member

   $$measurement-values-map-extension //= (
     &(tee.attributes: -82) => $tee-attributes-type
   )
   $tee-attributes-type /= $masked-value-type

   $$measurement-values-map-extension //= (
     &(tee.cryptokeys: -91) => [ + $tee-cryptokey-type ]
   )
```

```
$tee-cryptokey-type /= $crypto-key-type-choice

$$measurement-values-map-extension //= (
  &(tee.tcbdate: -72) => $tee-date-type
)
$tee-date-type /= ~tdate
$tee-date-type /= tdate
$tee-date-type /= time
$tee-date-type /= etime ; RFC9581
$tee-date-type /= period ; RFC9581

$$measurement-values-map-extension //= (
  &(tee.mrtee: -83) => $tee-digest-type
)
$$measurement-values-map-extension //= (
  &(tee.mrsigner: -84) => $tee-digest-type
)
$tee-digest-type /= digest ; see corim
$tee-digest-type /= digests-type ; see corim
$tee-digest-type /= tagged-exp-digest-member
$tee-digest-type /= tagged-exp-digest-not-member

$$measurement-values-map-extension //= (
  &(tee.isvprodid: -85) => $tee-isvprodid-type
)
$tee-isvprodid-type /= uint
$tee-isvprodid-type /= bstr

$$measurement-values-map-extension //= (
  &(tee.miscselect: -81) => $tee-miscselect-type
)
$tee-miscselect-type /= $masked-value-type

$$measurement-values-map-extension //= (
  &(tee.model: -71) => $tee-model-type
)
$tee-model-type /= tstr

op.eq=0
op.gt=1
op.ge=2
op.lt=3
op.le=4
op.mem=6
op.nmem=7

$$measurement-values-map-extension //= (
  &(tee.pceid: -80) => $tee-pceid-type
```

```
   )
   $tee-pceid-type /= tstr
   $tee-pceid-type /= uint

   $$measurement-values-map-extension //= (
     &(tee.isvsvn: -73) => $tee-svn-type
   )
   $tee-svn-type /= svn-type .within numeric-type
   $tee-svn-type /= tagged-numeric-ge
   $tee-svn-type /= tagged-int-range
   $tee-svn-type /= tagged-min-svn

   $$measurement-values-map-extension //= (
     &(tee.tcb-comp-svn: -125) => $tee-tcb-comp-svn-type
   )
   $tee-tcb-comp-svn-type /= [ 16*16 svn-type .within numeric-type ]
   $tee-tcb-comp-svn-type /= [ 16*16 tagged-numeric-ge ]
   $tee-tcb-comp-svn-type /= [ 16*16 tagged-int-range ]
   $tee-tcb-comp-svn-type /= [ 16*16 tagged-min-svn ]

   $$measurement-values-map-extension //= (
     &(tee-tcb-eval-num: -86) => $tee-tcb-eval-num-type
   )
   $tee-tcb-eval-num-type /= uint .within numeric-type
   $tee-tcb-eval-num-type /= tagged-numeric-ge
   $tee-tcb-eval-num-type /= tagged-int-range

   $$measurement-values-map-extension //= (
     &(tee.tcbstatus: -88) => $tee-tcbstatus-type
   )
   $tee-tcbstatus-type /= set-tstr-type
   $tee-tcbstatus-type /= tagged-exp-tstr-member
   $tee-tcbstatus-type /= tagged-exp-tstr-not-member

   $$measurement-values-map-extension //= (
     &(tee.vendor: -70) => $tee-vendor-type
   )
   $tee-vendor-type /= tstr

   $$measurement-values-map-extension //= (
     &(tee.platform-instance-id: -101) => $tee-platform-instance-id-type
   )
   $tee-platform-instance-id-type /= bstr
```

Authors' Addresses

James D. Beaney
Intel Corporation

Email: james.d.beaney@intel.com

Yogesh Deshpande
ARM Corporation
Email: yogesh.deshpande@arm.com

Andrew Draper
Altera Corporation
Email: andrew.draper@altera.com

Vincent R. Scarlata
Intel Corporation
Email: vincent.r.scarlata@intel.com

Ned Smith
Independent
Email: ned.smith.ietf@outlook.com

Francisco J. Chinchilla
Intel Corporation
Email: francisco.j.chinchilla@intel.com