TRUSTED® COMPUTING GROUP

SPECIFICATION

TCG TSS 2.0 JSON Data Types and Policy Language Specification

Version 0.7
Revision 04
September 25, 2019

Contact: admin@trustedcomputinggroup.org

Work in Progress

This document is an intermediate draft for comment only and is subject to change without notice. Readers should not design products based on this document.

PUBLIC REVIEW

DISCLAIMERS, NOTICES, AND LICENSE TERMS

NORMATIVE-INFORMATIVE LANGUAGE

The key words “MUST,” “MUST NOT,” “REQUIRED,” “SHALL,” “SHALL NOT,” “SHOULD,” “SHOULD NOT,” “RECOMMENDED,” “MAY,” and “OPTIONAL” in this document’s normative statements are to be interpreted as described in RFC-2119, Key words for use in RFCs to Indicate Requirement Levels.

ACKNOWLEDGEMENTS

TCG and the TSS Work Group would like to thank the following people for their work on this specification:

- Will Arthur Raytheon
- Brenda Baggaley Security Innovation (OnBoard Security)
- Dave Challener Johns Hopkins University
- Mike Cox Security Innovation (OnBoard Security)
- Andreas Fuchs Fraunhofer SIT
- Ken Goldman IBM
- Dominic Grauvogl Infineon Technologies AG
- Jürgen Repp Fraunhofer SIT
- Philip Tricca Intel
- Lee Wilson Security Innovation (OnBoard Security)
- Paul England Microsoft
- James Nguyen Wave

CHANGE HISTORY

REVISION	DATE	DESCRIPTION
0.6		• Initial Rewrite and definition
0.7r01	August 15, 2019	• Incorporate feedback • Adjusting examples to redefinitions
0.7r02	September 24, 2019	• Incorporate TC feedback
0.7r03	September 24, 2019	• Incorporate feedback from TSS-WG call
0.7r04	September 25, 2019	• Added table captions • Incorporated TC feedback

CONTENTS

1 Introduction

TPM 2.0 has a flexible design for authorizing actions on keys and other TPM objects. This document describes an interoperable scheme for describing policies and interpreting them to authorize TPM actions. This document further specifies a JSON encoding for this scheme and all TPM data types. This standardized interoperable form for policies allows different participants to author and consume policy expressions. For example, an enterprise management utility might author policies for key migration and recovery. These policies may then be consumed by any TPM software stack library. For an introduction to enhanced authorization policies please refer to the TPM Library Specification part 1 [11].

1.1 Acronyms

For definitions of the acronyms used in the TSS 2.0 specifications please see the TCG TSS 2.0 Overview and Common Structures Specification [22].

1.2 TCG Software Stack 2.0 (TSS 2.0) Specification Structure

At the time of writing, the documents that specify the TSS 2.0 are:

[1] TCG TSS 2.0 Overview and Common Structures Specification

[2] TCG TSS 2.0 TPM Command Transmission Interface (TCTI) API Specification

[3] TCG TSS 2.0 Marshaling/Unmarshaling (MU) API Specification

[4] TCG TSS 2.0 System API (SAPI) Specification

[5] TCG TSS 2.0 Enhanced System API (ESAPI) Specification

[6] TCG TSS 2.0 Feature API (FAPI) Specification

[7] TCG TSS 2.0 TAB and Resource Manager Specification

[8] TCG TSS 2.0 TSS Response Code (RC) API Specification

[9] TCG TSS 2.0 JSON Data Type and Policy Language Specification

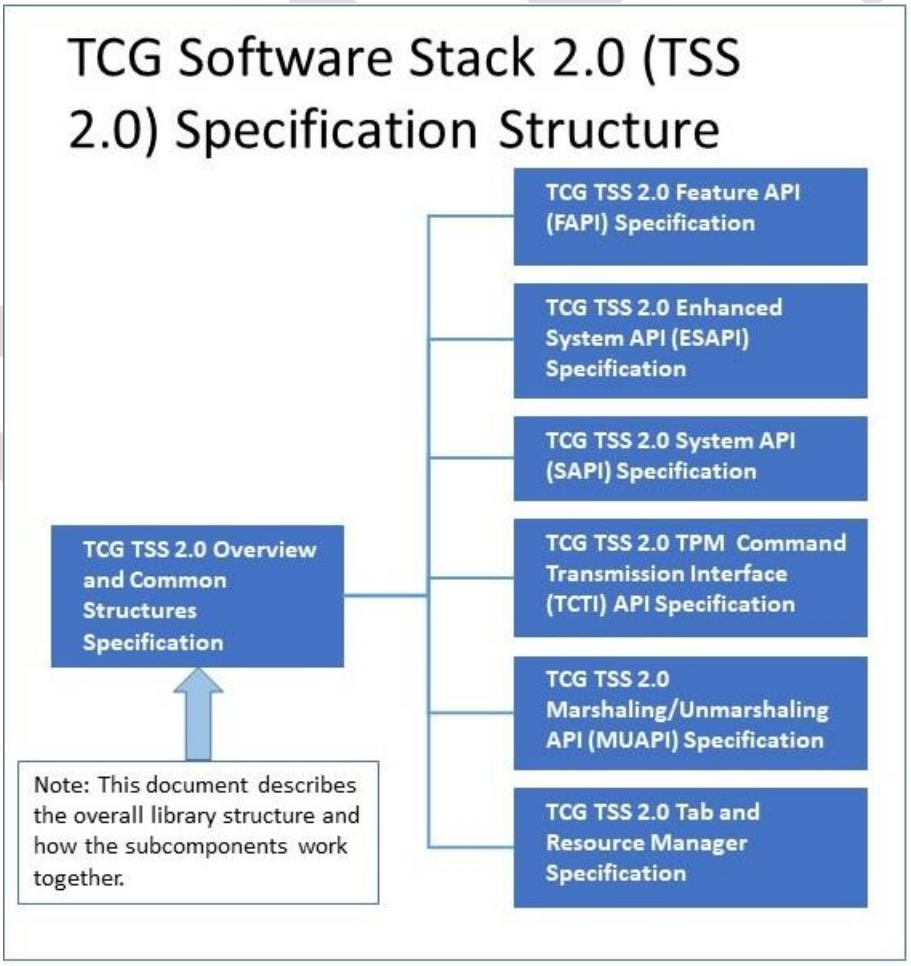

Figure 1: TSS 2.0 Specification Structure

1.3 References

Most references for the TSS 2.0 specifications are provided in the TCG TSS 2.0 Overview and Common Structures Specification [1].

The following additional references are used by this specification. The numbering continues from section 1.2:

[10] ISO C99 standard

[11] Trusted Platform Module Library Part 1: Architecture Family “2.0”

[12] Trusted Platform Module Library Part 2: Structures Family “2.0”

2 JSON encoding for TPM data types

This section provides a comprehensive list of transformation rules from TPM data type definitions (see TPM 2.0 Library Specification Part 2 [12]) to JSON.

This specification sometimes allows multiple representations of the same data type. A parser SHALL support all of these representations in order to allow human authors maximum flexibility. A serializer SHALL return one specific version which will be denoted as the “normal form”.

2.1 TPM base types

2.1.1 UINT8 to INT32

The TPM base types UINT8, INT8, UINT16, INT16, UINT32, INT32 SHOULD be represented as JSON integers or MAY be represented as JSON strings containing an integer in decimal or “0x” prefixed hex notation. JSON integers have a size of 54 bits and are signed, which easily accommodates all of the above data types.

The normal form produced by TSS components SHOULD be a JSON integer.

This encoding SHALL also be used for typedefs and TPMI interface types derived from the base types. For example, TPM2_HANDLE is represented the same as a UINT32.

Examples:

1 (normal form)
2 (normal form)
3 (normal form)
0x81000001
“1”
“0x8103ADce”

2.1.2 UINT64 and INT64

The TPM base types UINT64 and INT64 do not fit into the JSON integer representation. If the UINT64 or INT64 value is small enough for a signed 54 bit representation, the JSON integer SHOULD be used. If the value does not fit into the signed 54 bit representation, an array of two 32-bit values SHOULD be used – i.e. *[(value / (2^32)), (value % (2^32))].* UINT64 and INT64 values MAY be represented as JSON strings containing an integer in decimal or “0x”-prefix hex notation.

The normal form produced by TSS components SHOULD be a single JSON integer if the values are small enough or a JSON array of two JSON integers containing the higher and low 32-bit values respectively.

Examples:

1 (normal form for <=54 signed bits)
“2”
“0x1234123412341234”
[123, 456] (normal form for >54 bits)

2.1.3 Named constants and Enumerations

Named constants and enumerations SHOULD be represented as a JSON string containing the value names from the TPM 2.0 library Specification Part 2 with TPM2_ as a prefix. These value names SHOULD be shortened by removing any TPM2_ prefixes or by removing their types’ name prefixes. The casing of the characters SHALL be ignored.

Alternatively, such values MAY be represented in the same way as the base types from which they derive as specified in section 2.1.1.

The normal form produced by TSS components SHOULD be the shortened string representation as described in the first paragraph of this subsection (section 2.1.3).

Examples:

"TPM2_RH_OWNER"
"RH_OWNER"
"OWNER" (normal form)
"0x40000001"

"TPM2_ALG_SHA1"
"ALG_SHA1"
"SHA1" (normal form)
"0x0004"

Data type aliases are treated as aliases for the purpose of this specification. Thus, the representation of a TPM2_HANDLE is the same as that of a UINT32.

2.2 TPM2Bs byte buffer types

TPM2Bs are differentiated into simple TPM2Bs and complex TPM2Bs. Simple TPM2Bs are defined by the TPM types specification as a size and a sized array of bytes. Complex TPM2Bs are defined as a size and an embedded TPMS or TPMT.

2.2.1 Simple TPM2Bs

Simple TPM2Bs SHOULD be represented as JSON strings containing a BASE16 (HEX) representation of the byte array (without any prefix), including any leading zeros. A prefix of "0x" MAY be prepended. The size field of the TPM2B the length of the JSON string (without the prefix) divided by 2. Alternatively, a simple TPM2B MAY be represented as a JSON array of JSON integers (one per byte) and the size inferred directly from the number of elements of the array.

The normal form produced by TSS components SHOULD be the JSON string representation without the prefix..

Examples:

"0a0d82fd00" (normal form)
"0x0a0d82fd00382838848499239823823894 94"
[10, 22, 145, 21, 0, 123]

2.2.2 Complex TPM2Bs

Complex TPM2Bs do not have a JSON representation. Instead the TPMT or TPMS embedded in a complex TPM2B SHALL be JSON encoded and inserted in place of the complex TPM2B.
Implementers note: When parsing a complex TPM2B field from JSON into a C data type as defined in the TCG TSS Overview and Common Structures Specification [1], the size field of the complex TPM2B can be set to 0, since it is ignored and inferred during marshaling, as described in the TCG TSS System API Specification [4].

Examples:

```
//TPM2B_ECC_POINT using the TPMS_ECC_POINT directly:
{
   "x": "1a2b3c",
   "y": "4d5e6g"
}
```

2.3 TPML list types

A TPML type SHALL be represented as a JSON array. The size of the TPML type is inferred from the number of elements of the JSON array.

Example:

```
//TPML_CC an array of TPM2_CC types:
[ "0x0000011f", "TPM22_CC_EvictControl", "HierarchyControl" ]
```


2.4 TPMS struct types

A TPMS type SHALL be represented as JSON objects – also referred to as dicts. Each of the TPMS fields is a key of the object and the associated content is the value – which may be a TPM2B, Basetype, TPMT, etc.

Note: The order of the fields in the JSON object can differ from that specified in the TPMS type since JSON objects are unordered data type.

Example:

```
//Example TPMS_CLOCK_INFO:
{
   “clock”: 12345,
   “resetCount”: 20,
   “restartCount”: 0,
   “safe”: “no”
}
```

Exceptions:

The types TPMS_PCR_SELECT, the pcrSelect field inside TPMS_PCR_SELECTION, and TPMS_TAGGED_PCR_SELECT are special cases. Instead of providing an array of the values they SHALL be represented as an array of bits that are set. The sizeofSelect fields are not represented.

Examples:

```
//TPMS_PCR_SELECT:
[ 0, 1, 2, 5, 17 ]

//TPMS_PCR_SELECTION:
{
   “hash”: “sha1”,
   “pcrSelect”: [ 0, 2, 4, 6 ]
}

//TPMS_TAGGED_PCR_SELECT:
{
   “tag”: “extend_l0”,
   “pcrSelect”: [ 0, 2, 4, 6 ]
}
```

2.5 TPMU union types

TPMU unions do not have a JSON representation. Instead the data type embedded in a TPMU union SHALL be JSON encoded and inserted in place of the TPM union.

Exception:

The TPMU_NAME does not contain a surrounding type selector. In this case the value can be inferred from the JSON type if it is a JSON integer or string for a TPM2_HANDLE or if it is a dict for a TPMT_HA.

Examples:

```
//TPMS_CAPABILITY containing a TPMU_CAPABILITY:
{
   “capability”: “pp_commands”,
   “data”: [ “UndefineSpaceSpecial” ]
}
```



```
//Examples of TPMU_NAMEs
0x40000001
“TPM2_RH_OWNER”
{
   “hashAlg”: “sha1”,
   “digest”: “0x0102030405060708090a0b0c0d0e0f01020304”
}
```

2.6 TPMA attribute types

TPMA attribute fields SHOULD be represented as JSON objects or MAY be represented as JSON arrays of strings or as the corresponding base type. If an attribute is not present it SHALL be assumed as 0 / CLEAR / NO.

Example:

```
{                              (normal form)
            “TPMA_OBJECT_NODA”: 1,
            “noda”: 1,
            “fixedparent”: 1,
            “authwrite”: 0,
            “fixedtpm”: “SET”,
            “TPM2_NT”: “TPM2_NT_PIN_FAIL”
}
[ “noda”, “fixedparent”, “TPM2_NT_PIN_FAIL” ]
“0x0005”
“0000000101b”
5
```


3 Additional intermediate data types

This section introduces a set of data types that are used in the Policy Language Elements of the following section.

The tables in this section describe the mandatory data definitions.

3.1 Additional TPM-like data types

This set of TPM-like data types are defined with the same semantics as in the TPM 2.0 Library Specification Part 2 and can encoded in JSON as described in section 2.

3.1.1 TPMS_POLICYAUTHORIZATION

This structure is used to represent a signature with an accompanying public key. Depending on the value of the type field, its structure contains the fields listed in one of the following two tables. A value of “tpm” indicates the structure of Table 1, a value of “pem” indicates the structure of Table 2.

Table 1 TPMS_POLICYAUTHORIZATION of type “tpm”

Field name	Type	Description
type	String	Value “tpm”
key	TPMT_PUBLIC	The public key capable of verifying the signature. The key is encoded in the format of the TPM data type TPMT_PUBLIC.
policyRef	TPM2B_NONCE	A byte array to be included in the signature. Optional. Default is 0-length.
signature	TPMT_SIGNATURE	Signature of the policy authorization in the format of the TPM data type TPMT_SIGNATURE.

Table 2 TPMS_POLICYAUTHORIZATION of type “pem”

Field name	Type	Description
type	String	Value “pem”
key	String (PEM pubkey)	The public key capable of verifying the signature. The key is encoded in the PEM format. Note: Includes the algorithm used and the numbits+exponent or curveid.
rsaScheme	TPMT_RSA_SCHEME	SSA (PKCS1) or PSS for RSA. Ignored for ECDSA. Optional. Default is SSA (PKCS1)
hashAlg	TPMI_HASH_ALG	Used for the key’s name calculation and for the policy’s aHash calculation (for ECDSA and RSA). Optional. Default is SHA256
policyRef	TPM2B_NONCE	A byte array to be included in the signature. Optional. Default is 0-length.

Field name	Type	Description
signature	String (hex)	Hex encoded signature of the policy authorization. In case of ECDSA, the format from IETF RFC 5480 is used.

Example:

```
{
   “type”: “tpm”,
   “key”: {
      “type”: “ECDSA”,
      “nameAlg”: “SHA256”,
      “objectAttributes”: “SIGN”,
      “authPolicy”: “0x123abc123abc1234def”,
      “parameters”: {
         “symmetric”: …,
         “scheme”: “ECDSA”,
         “curveId”: “NIST_P256”,
         “kdf”: …
      },
      “unique”: {
         “x”: “0x123123…”,
         “y”: “0x456456…”
      }
   },
   “signature”: {
      “hash”: “SHA256”,
      “signatureR”: “0x789789…”,
      “signatrueS”: “0xabcabc…”
   }
}
{
   “type”: “pem”,
   “key”: “--- BEGIN RSA PUBLIC KEY --- \n3df…adf==\n--- END RSA PUBLIC KEY --“,
   “signature”: “0x1238af8…38”
}
```

3.1.2 TPMU_POLICYELEMENT

The union in Table 3 describes all possible policy elements.

Table 3 TPMU_POLICYELEMENT

Field name	Type	Selector
policyOr	TPMS_POLICYOR	or
policySigned	TPMS_POLICYSIGNED	signed
policySecret	TPMS_POLICYSECRET	secret
policyPCR	TPMS_POLICYPCR	pcr
policyLocality	TPMS_POLICYLOCALITY	locality
policyNV	TPMS_POLICYNV	nv

Field name	Type	Selector
policyCounterTimer	TPMS_POLICYCOUNTERTIMER	counterTimer
policyCommandCode	TPMS_POLICYCOMMANDCODE	commandCode
policyPhysicalPresence	TPMS_POLICYPHYSICALPRESENCE	physicalPresence
policyCpHash	TPMS_POLICYCPHASH	cpHash
policyNameHash	TPMS_POLICYNAMEHASH	nameHash
policyDuplicationSelect	TPMS_POLICYDUPLICATIONSELECT	duplicationSelect
policyAuthorize	TPMS_POLICYAUTHORIZE	authorize
policyAuthValue	TPMS_POLICYAUTHVALUE	authValue
policyPassword	TPMS_POLICYPASSWORD	password
policyNvWritten	TPMS_POLICYNVWRITTEN	nvWritten
policyTemplate	TPMS_POLICYTEMPLATE	template
policyAuthorizeNv	TPMS_POLICYAUTHORIZENV	authorizeNv
policyAction	TPMS_POLICYACTION	action

Examples of each union instance are in section 4.

3.1.3 TPMS_PCRVALUE

The structure in Table 4 represents the value of a specific PCR in a specific PCR bank (denoted by hashAlg).

Table 4 TPMS_PCRVALUE

Field name	Type	Description
pcr	TPM2_HANDLE	Handle of the PCR index.
hashAlg	TPMI_ALG_HASH	Selector for the PCR bank.
[hashAlg] digest	TPMU_HA	The digest value of the PCR index in this bank.

Example:

```
{
   “pcr”: 1,
   “hashAlg”: “SHA256”,
   “digest”: “0x123…123”
}
```

3.2 Additional non-TPM-like list types

This section includes a list of non-TPM like data definitions. They mostly resemble a TPML data type from TPM 2.0 Library Specification Part 2 but differ in that the lists are not bounded by a maximum value. As such the JSON representation SHALL be the same as that of a TPM-like TPML. The C struct representation is variable and left to the implementer.

3.2.1 TPML_POLICYAUTHORIZATIONS

This data type SHALL be encoded as a JSON array of TPMS_POLICYAUTHORIZATION.

Example:

```
[
  {
    “type”: “tpm”,
    “key”: {
      “type”: “ECDSA”,
      “nameAlg”: “SHA256”,
      “objectAttributes”: “SIGN”,
      “authPolicy”: “0x123abc123abc1234def”,
      “parameters”: {
        “symmetric”: …,
        “scheme”: “ECDSA”,
        “curveId”: “NIST_P256”,
        “kdf”: …
      },
      “unique”: {
        “x”: “0x123123…”,
        “y”: “0x456456…”
      }
    },
    “signature”: {
      “hash”: “SHA256”,
      “signatureR”: “0x789789…”,
      “signatrueS”: “0xabcabc…”
    }
  }, {
    “type”: “pem”,
    “key”: “--- BEGIN RSA PUBLIC KEY --- \n3df…adf==\n--- END RSA PUBLIC KEY ---“,
    “signature”: “0x1238af8…38”
  }
]
```

3.2.2 TPML_POLICYELEMENTS

This data type SHALL be encoded as a JSON array of TPMT_POLICYELEMENT.

Example:

```
[
  {
    “type”: “signed”,
    …
  }, {
    “type”: “pcr”,
    …
  }
]
```

3.2.3 TPML_POLICYBRANCHES

This data type SHALL be encoded as a JSON array of TPMS_POLICYBRANCH.

Example:

```
[
  {
    “name”: “SmartCard”,
    “description”: “SmartCard-based authentication”,
    “policy”: [ {
      “type”: “PolicySigned”,
      …
    } ]
  }, {
    “name”: “Password”,
    “description”: “Password-based authentication”,
    “policy”: [ {
      “type”: “PolicyPassword”
    } ]
  }
]
```

3.2.4 TPML_PCRVALUES

This data type SHALL be encoded as a JSON array of TPMS_PCRVALUES.

Example:

```
[
  {
    “pcr”: 1,
    “hashAlg”: “SHA256”,
    “digest”: “0x123…123”
  }, {
    “pcr”: 2,
    “hashAlg”: “SHA256”,
    “digest”: “0xabc…abc”
  }
]
```

3.3 Additional non-TPM-like types

3.3.1 TPMI_POLICYTYPE (JSON String)

The values in Table 5 are a list of selector keywords for the type of a policy, used in TPMT_POLICYELEMENT.

Table 5 TPMI_POLICYTYPE

Value	Description
or	PolicyOr
signed	PolicySigned
secret	PolicySecret
pcr	PolicyPcr
locality	PolicyLocality
nv	PolicyNV
counterTimer	PolicyCounterTimer
commandCode	PolicyCommandCode
physicalPresence	PolicyPhysicalPresence

Value	Description
cpHash	PolicyCpHash
nameHash	PolicyNameHash
duplicationSelect	PolicyDuplicationSelect
authorize	PolicyAuthorize
authValue	PolicyAuthValue
password	PolicyPassword
nvWritten	PolicyNvWritten
template	PolicyTemplate
authorizeNv	PolicyAuthorizeNv
action	PolicyAction. The only non-TPM policy element.

Examples:

"or"

3.3.2 TPMT_POLICYELEMENT

The data type whose fields are listed in Table 6 represents one element within the policy tree. Note that the fields of the TPMU_POLICYELEMENT appear on the same level as the TPMI_POLICYTYPE.

Table 6 TPMT_POLICYELEMENT

Field name	Type	Description
type	TPMI_POLICYTYPE	The type of policy
policyDigests	TPML_DIGEST_VALUES	Optional. A digest for this policy subtree.
[type] -	TPMU_POLICYELEMENT	The union does is not embedded inside a field.

Example:

```
{
   "type": "locality",
   "locality": "L0"
}
```

3.3.3 TPMS_POLICYBRANCH

The data type whose fields are listed in Table 7 represents the root of a subtree of policies beneath a PolicyOr statement. The name and description are used to query the user during policy evaluation, to determine which branch the user wants to pursue.

Table 7 TPMS_POLICYBRANCH

Field name	Type	Description
name	String [a-zA-Z0-9_-]	Short name of the or-branch.

Field name	Type	Description
description	String	Optional. A human readable description of the or-branch.
policyDigests	TPML_DIGEST_VALUES	Optional.
policy	TPML_POLICYELEMENTS	List of policy elements.

Example:

```
{
   “name”: “SmartCard”,
   “description”: “SmartCard-based authentication”,
   “policy”: [ {
      “type”: “signed”,
      …
   } ]
}
```


4 The Policy Language

This section describes the basic policy harness as well as all possible elements of a policy or policy template statement.

A policy template (in contrast to a policy) also allows certain additional statements to be made. Most notably, for PoicyPCR, it allows the selection of current PCR values without requiring the specific value to be provided. When the template is used during object creation, the values of the PCRs are automatically filled in from the TPM's current PCRs and the policy is then calculated. This means that a policy template will not carry policy digest values or policy authorization signatures.

Many types in this section contain a path element that refers to a TPM object inside a Feature API key store. In each of these cases an alternative exists where an explicit value can be provided instead, making the language portable and applicable without a FAPI implementation.

The tables in this section describe the mandatory data definitions.

- X denotes a required field.
- O denotes an optional field.
- C denotes choices for the user. Exactly one of those fields must be present.
- - denotes fields that are not allowed.

Note that the examples in this section are not test cases, since they are oftentimes abbreviated.

4.1 Policy root element (TPMS_POLICY)

The data type whose fields are listed in Table 8 describes the top level element of every policy or policy template statement.

Table 8 TPMS_POLICY

Field name	Type	Policy	Template	Description
shortname	String [a-zA-Z0-9]+	X	X	The name of the policy
description	String [UTF-8]	O	O	The human readable description
policyDigests	TPML_DIGEST_VALUES	O	-	A list of digest values for the encoded policy for different hash algorithms
policyAuthorizations	TPML_POLICYAUTHORIZATIONS	O	-	A list of signed authorizations for the encoded policy.
policy[]	TPML_POLICYELEMENTS	X	X	An arbitrarily long list of policy statements

Example:


```
{
	“name”: “MyFirstPolicy”,
	“description”: “This is the first example policy’s description for humans”,
	“policyDigests”: [
		{
			“hashAlg”: “sha1”,
			“digest”: “02389ca80402389ca80402389ca80402389ca804”
		}, {
			“hashAlg”: “sha256”,
			“digest”: “12340202389ca80402389ca80402389ca80402389ca804389ca80402389ca8040238”
		}
	],
	“policyAuthorizations”: [
		{
			“type”: “pem”,
			“key”: “–– BEGIN RSA PUBLIC KEY –– \n3df…adf==\n–– END RSA PUBLIC KEY ––“,
			“signature”: “0x1238af8…38”
		}
	],
	“policy”: [
		{
			“type”: “password”
		}
	]
}
```

4.2 PolicyOr (TPMS_POLICYOR)

The data type whose fields are listed in Table 9 represents a PolicyOr statement with a set of branches

Table 9 TPMS_POLICYOR

Field name	Type	Policy	Template	Description
branches	TPML_POLICYBRANCHES	X	X	An (infinite) array of policy elements.

The number of elements in branches array is infinite. If there are more than 8 branches, this array SHALL be represented as a B+-Tree of TPM2_PolicyOr statements. This means that if a new layer of TPM2_PolicyOr statements is introduced, the upper layers will contain only TPM2_PolicyOrs, i.e. they will not contain actual branches.

Examples where bn denotes the nth branch in the JSON array:

- 8 branches: Or(b0, b1 … b7)
- 9 branches: Or(Or(b0, b1 … b7), Or(b8))
- 65 branches: Or(Or(Or(b0, b1 … b7) … Or(b55, b56 … b63)), Or(Or(b65)))

Typical JSON encoded policies SHOULD contain only a single PolicyOr statement at the top of the tree that is extended as described.

Example:

```
{
	“name”: “MyOrPolicy”,
	“description”: “This is the first example policy’s description for humans”,
	“policy”: [
		{
```



```
                "type": "or",
                "branches": [
                        {
                                "name": "branchA",
                                "description": "The first possible branch",
                                "policy": [
                                        {
                                                "type": "password"
                                        }, {
                                                "type": "locality",
                                                "locality": "L0"
                                        }
                                ]
                        }, {
                                "name": "branchB",
                                "description": "The second possible branch",
                                "policy": [
                                        {
                                                "type": "authValue."
                                        }
                                ]
                        }
                ]
        }
    ]
}
```

4.3 PolicySigned (TPMS_POLICYSIGNED)

The data type whose fields are listed in Table 10 is used for signature based authorizations. It allows smartcards or other means to be used for authorizing a policy.

Table 10 TPMS_POLICYSIGNED

Field name	Type	Policy	Template	Description
nonceTPM	TPM2B_NONCE	-	-	This is a value returned by TPM2_StartAuthSession and thus not used in the Policy language.
cpHashA	TPM2B_DIGEST	O	O	Default is zero-length.
policyRef	TPM2B_NONCE	O	O	Default is zero-length.
expiration	INT32	-	-	This value SHALL be set to zero by a JSON policy parser.
auth	TPMT_SIGANTURE	-	-	This value is generated from at runtime via a callback.
publicKey	TPM2B_NAME	-	-	This SHALL be automatically generated from keyPath, keyPublic or keyPEM by JSON policy parsers.

Field name	Type	Policy	Template	Description
keyPath	String	-	C	A reference to a key inside the FAPI keystore.
keyPublic	TPMT_PUBLIC	C	C	A TPMT_PUBLIC structure from which the key name is derived.
keyPEM	String [PEM pubkey]	C	C	A TPM2B_NAME derived from keyPEM SHALL be constructed with a TPMT_PUBLIC from this key with the following attribute set: .policyDigest.size = 0, .objectAttributes = SIGN and .hashAlg = keyPEM->hashAlg.
keyPEMhashAlg	TPMI_ALG_HASH	O	O	Default value is SHA256.

Example:

```
{
        “name”: “MySignedPolicy”,
        “description”: “This is the first example policy’s description for humans”,
        “policy”: [
                {
                        “type”: “signed”,
                        “policyRef”: “039af9039af9039af900039af9039af9039af900”,
                        “keyPEM”: “--- BEGIN RSA PUBLIC KEY ---\n1231231233\n--- END RSA PUBLIC KEY ---“
                }
        ]
}
```

4.4 PolicySecret (TPMS_POLICYSECRET)

The data type whose fields are listed in Table 11 describes the values for a policy secret statement.

Table 11 TPMS_POLICYSECRET

Field name	Type	Policy	Template	Description
nonceTPM	TPM2B_NONCE	-	-	This is a value returned by TPM2_StartAuthSession and thus not used in the Policy language
cpHashA	TPM2B_DIGEST	O	O	Default is zero-length
policyRef	TPM2B_NONCE	O	O	Default is zero length
expiration	INT32	-	-	This value SHALL be set to zero by JSON policy parsers.
objectPath	String	C	C	Path of the object

Field name	Type	Policy	Template	Description
objectName	TPM2B_DIGEST	C	C	Public name of the object

Example:

```
{
        “name”: “MySecretPolicy”,
        “description”: “This is the first example policy’s description for humans”,
        “policy”: [
                {
                        “type”: “secret”,
                        “objectPath”: “/nv/myIndex”
                }, {
                        “type”: “secret”,
                        “policyRef”: “039af9039af9039af900039af9039af9039af900”,
                        “objectName”: “3918d919283918d919283918d919283918d91928”
                }
        ]
}
```

4.5 PolicyPCR (TPMS_POLICYPCR)

The data type whose fields are listed in Table 12 describes a PCR restriction for a policy.

Table 12 TPMS_POLICYPCR

Field name	Type	Policy	Template	Description
pcrs	TPML_PCRVALUES	X	C	The list of pcr values.
currentPCRs	TPMS_PCR_SELECT	-	C	An implementation supporting this field SHALL construct the list of actual pcr values using its current default hashAlg and read the current PCR values from the TPM.
currentPCRandBanks	TPML_PCR_SELECTION	-	C	An implementation supporting this field SHALL construct the list of actual pcr values using the provided hashAlg and read the current PCR values from the TPM.

In the case of a policy template, the user has the option to provide currentPCRs instead of actual values. The template SHALL then be instantiated and stored on use during creation of an object.

Example:

```
{
        “name”: “MyPCRPolicy”,
        “description”: “This is the first example policy’s description for humans”,
```



```
	"policy": [
		{
			"type": "pcr",
			"pcrs": [
				{ "hashAlg": "sha1", "pcr": 0, "digest": "01939233923" },
				{ "hashAlg": "sha1", "pcr": 2, "digest": "01939393923" },
				{ "hashAlg": "sha1", "pcr": 7, "digest": "01939393923" },
				{ "hashAlg": "sha256", "pcr": 17, "digest": "019393992339342" },
				{ "hashAlg": "sha256", "pcr": 18, "digest": "019393992323432" },
				{ "hashAlg": "sha256", "pcr": 19, "digest": "019393992323432" }
			]
		}, {
			"type": "pcr",
			"currentPCRs": [ 0, 2, 7, 17, 18, 19 ],
			"currentPCRandBanks": [ { "hashAlg": "sha1", "select": [ 0, 2, 7 ] },
				{ "hashAlg": "sha256", "select": [ 17, 18, 19 ] } ],
		}
	]
}
```

4.6 PolicyLocality (TPMS_POLICYLOCALITY)

The data type whose fields are listed in Table 13 describes a policy element that restricts the usage of an object to a certain locality.

Table 13 TPMS_POLICYLOCALITY

Field name	Type	Policy	Template	Description
locality	TPMA_LOCALITY	X	X	The list of localities required.

Example:

```
{
	"name": "MyOrPolicy",
	"description": "This is the first example policy's description for humans",
	"policy": [
		{
			"type": "locality",
			"locality": [ "ZERO", "TWO" ]
		}
	]
}
```

4.7 PolicyNV (TPMS_POLICYNV)

The data type whose fields are listed in Table 14 describes a policy element that requires a certain NV index to contain certain values.

Table 14 TPMS_POLICYNV

Field name	Type	Policy	Template	Description
nvPath	String	C	C	The path to the NV index.
nvIndex	TPMI_RH_NV_INDEX	C	C	The handle value to the NV index.

Field name	Type	Policy	Template	Description
authHandle	TPMI_RH_NV_AUTH	-	-	This is determined by FAPI at runtime.
operandB	TPM2B_OPERAND	X	X	The operation used by the TPM to compare the current content of the NV index.
offset	UINT16	O	O	Default value is 0.
operation	TPM2_EO	O	O	Default value is “EQUAL”.

Example:

```
{
        “name”: “MyNvPolicy”,
        “description”: “This is the first example policy’s description for humans”,
        “policy”: [
                {
                        “type”: “nv”,
                        “nvPath”: “/nv/mylittleIndex”,
                        “operandB”: “1288a8394”
                }, {
                        “type”: “nv”,
                        “nvIndex”: “0x82000012”,
                        “offset”: 20,
                        “operandB”: “1288a8394”,
                        “operation”: “UNSIGNED_GT”
                }
        ]
}
```

4.8 PolicyCounterTimer (TPMS_POLICYCOUNTERTIMER)

The data type whose fields are listed in Table 15 describes a policy element that allows applying time restrictions on TPM object usage.

Table 15 TPMS_POLICYCOUNTERTIMER

Field name	Type	Policy	Template	Description
operandB	TPM2B_OPERAND	X	X	The value with which the TPM will compare its current timers.
offset	UINT16	O	O	Default is 0
operation	TPM2_EO	X	X	The operation used by the TPM for comparison.

Example:

```
{
        “name”: “MyCounterTimerPolicy”,
        “description”: “This is the first example policy’s description for humans”,
        “policy”: [
                {
```



```
                "type": "counterTimer",
                "operandB": "0a02",
                "operation": "UNSIGNED_GT"
            }
        ]
}
```

4.9 PolicyCommandCode (TPMS_POLICYCOMMANDCODE)

The data type whose fields are listed in Table 16 describes a policy element that allows the restriction of the command code to be used for this policy branch.

Table 16 TPMS_POLICYCOMMANDCODE

Field name	Type	Policy	Template	Description
code	TPM2_CC	X	X	The command code that is allowed by this policy.

Note: Applying policies to commands with the "Auth Role: Admin" requires a PolicyCommandCode; e.g. TPM2_NV_ChangeAuth().

Example:

```
{
        "name": "MyOrPolicy",
        "description": "This is the first example policy's description for humans",
        "policy": [
            {
                "type": "commandCode",
                "code": "NV_INCREMENT"
            }
        ]
}
```

4.10 PolicyPhysicalPresence (TPMS_POLICYPHYSICALPRESENCE)

This policy element requires the TPM to check for physical presence. It contains no further fields as shown in its data type definition in Table 17.

Table 17 TPMS_POLICYPHYSICALPRESENCE

Field name	Type	Policy	Template	Description
-				PolicyPhysicalPresence contains no further fields.

Example:

```
{
        "name": "MyOrPolicy",
        "description": "This is the first example policy's description for humans",
        "policy": [
            {
                "type": "physicalPresence"
            }
        ]
}
```

4.11 PolicyCpHash (TPMS_POLICYCPHASH)

The data type whose fields are listed in Table 18 describes a policy element that allows the restriction of allowed commands by restricting the allowed cpHash of the command.

Table 18 TPMS_POLICYCPHASH

Field name	Type	Policy	Template	Description
cpHash	TPM2B_DIGEST	X	X	The hash value of the command parameters that is granted by this policy.

Example:

```
{
        “name”: “MyOrPolicy”,
        “description”: “This is the first example policy’s description for humans”,
        “policy”: [
                {
                        “type”: “cpHash”,
                        “cpHash”: “939c922188aadf939c922188aadf939c922188aa”
                }
        ]
}
```

4.12 PolicyNameHash (TPMS_POLICYNAMEHASH)

The data type whose fields are listed in Table 19 describes a policy element that allows the restriction of this policy to only apply to an object with the specified name.

Table 19 TPMS_POLICYNAMEHASH

Field name	Type	Policy	Template	Description
nameHash	TPM2B_DIGEST	X	C	The hash value of the object to be used during the command.
namePath	String	-	C	The path to the object to be used during the command. Based on the namePath the actual nameHash value SHALL derived by the implementation.

Example:

```
{
        “name”: “MyNameHashPolicy”,
        “description”: “This is the first example policy’s description for humans”,
        “policy”: [
                {
                        “type”: “nameHash”,
                        “namePath”: “/nv/myIndex”
                }, {
                        “type”: “nameHash”,
                        “nameHash”: “82000012”
```



```
            },
            …
        ]
}
```

4.13 PolicyDuplicationSelect (TPMS_DUPLICATIONSELECT)

The data type whose fields are listed in Table 20 describes a policy element that can be used to select a duplication target for a key.

Table 20 TPMS_DUPLICATIONSELECT

Field name	Type	Policy	Template	Description
objectName	TPM2B_NAME	O	O	The name of the object to be duplicated.
newParentName	TPM2B_NAME	X	C	Automatically calculated if newParentPath or newParentPublic are used instead.
includeObject	TPM2_YES_NO	-	-	Will be YES if objectName.size > 0.
newParentPath	String	-	C	The path to the new parent object. An implementation will use this to calculate the actual newParentName.
newParentPublic	TPMT_PUBLIC	-	C	The public key of the new parent object. An implementation will use this to calculate the actual newParentName.

Example:

```
{
        “name”: “MyDuplicationPolicy”,
        “description”: “This is the first example policy’s description for humans”,
        “policy”: [
            {
                “type”: “duplicationSelect”,
                “newParentName”: “0105af700105af700105af700105af700105af70”
            }
        ]
}
```

4.14 PolicyAuthorize (TPMS_POLICYAUTHORIZE)

The data type whose fields are listed in Table 21 describes a policy element that can be used to add authorization for new policy trees after the creation of the original object policy by providing a signature to the authorizations list in the policy harness.

Table 21 TPMS_POLICYAUTHORIZE

Field name	Type	Policy	Template	Description
approvedPolicy	TPM2B_DIGEST	-	-	The digest value of the approved Policy. Will be retrieved from the policyAuthorizations field of a TPMS_POLICY upon usage of the policy.
policyRef	TPM2B_NONCE	O	O	An arbitrary value to be used in the policyRef field of policy authorizations.
keyName	TPM2B_NAME	-	-	Not exposed in JSON, but generated from keyPath, keyPublic or keyPEM.
checkTicket	TPMT_TK_VERIFIED	-	-	The signature ticket. Will be retrieved from a TPM2_VerifySignature operation on the policyAuthorizations field of a TPMS_POLICY upon usage of the policy.
keyPath	String	-	C	A reference to a key inside the FAPI keystore.
keyPublic	TPMT_PUBLIC	C	C	The public key used for validating authorized policies.
keyPEM	String [PEM public key]	C	C	An implementation SHALL construct a TPM2B_NAME from a TPMT_PUBLIC containing this key and the following attribute set: .policyDigest.size = 0, .objectAttributes = SIGN and hashAlg = keyPEM->hashAlg.
keyPEMhashAlg	TPMI_ALG_HASH	O	O	Default is SHA256.

Example:

```
{
        “name”: “MyAuthorizePolicy”,
        “description”: “This is the first example policy’s description for humans”,
        “policy”: [
                {
                        “type”: “authorize”,
                        “keyPath”: “/UDK/myAuthorizingKey”
                }, {
                        “type”: “authorize”,
```



```
                    “keyPublic”: {
                            “type”: “ECDSA”,
                            “nameAlg”: “SHA256”,
                “objectAttributes”: [ “sign”, “fixedTPM” ],
                “policyDigest”: “123895823…123891238”,
                “parms”: {
                        “scheme”: “ECDSA”,
                        “curveId”: “NIST_P256”
                },
                            “unique”: {
                         “x”: “1838as939...123932”,
                         “y”: “1838as939...123932”
                }
                    }
                }, {
                    “type”: “authorize”,
                    “keyPEM”: “--- BEGIN RSA PUBLIC KEY ---\n1218adf…23==\n--- END RSA PUBLIC KEY ---”
                }
        ]
}
```

4.15 PolicyAuthValue (TPMS_POLICYAUTHVALUE)

This policy element is used to require the auth value of the object to be provided. The data type does not contain any further fields, as shown in Table 22.

Note that PolicyAuthValue will use the auth value of the object in the TPM command buffer in plain text instead of transferring an HMAC derived from the auth value. It is recommended to use PolicyPassword instead of PolicyAuthValue for improved security, since the auth value is not transferred in plain text.

Table 22 TPMS_POLICYAUTHVALUE

Field name	Type	Policy	Template	Description
-				PolicyAuthValue contains no further fields.

Example:

```
{
        “name”: “MyAuthValuePolicy”,
        “description”: “This is the first example policy’s description for humans”,
        “policy”: [
                {
                    “type”: “authValue”
                }
        ]
}
```

4.16 PolicyPassword (TPMS_POLICYPASSWORD)

This policy element requires the auth value of the object to be included in the final HMAC calculation. The data type does not contain any further fields, as shown in Table 23.

Table 23 TPMS_POLICYPASSWORD

Field name	Type	Policy	Template	Description
-				PolicyPassword contains no further fields.

Example:

```
{
        “name”: “MyPasswordPolicy”,
        “description”: “This is the first example policy’s description for humans”,
        “policy”: [
                {
                        “type”: “password”
                }
        ]
}
```

4.17 PolicyNvWritten (TPMS_POLICYNVWRITTEN)

The data type whose fields are listed in Table 24 describes a policy element that requires an NV-Space to have been written at least once or never have been written.

Table 24 TPMS_POLICYNVWRITTEN

Field name	Type	Policy	Template	Description
writtenSet	TPMI_YES_NO	O	O	Default is “yes”

Example:

```
{
        “name”: “MyNvWrittenPolicy”,
        “description”: “This is the first example policy’s description for humans”,
        “policy”: [
                {
                        “type”: “nvWritten”,
                        “writtenSet”: “no”
                }
        ]
}
```

4.18 PolicyTemplate (TPMS_POLICYTEMPLATE)

The data type whose fields are listed in Table 25 describes a policy element that can be used to restrict the templates of a subkey by providing either a hash of a public area or the template itself. A policy template can furthermore refer to a FAPI template that is being evaluated together with the current crypto profile during object creation.

Table 25 TPMS_POLICYTEMPLATE

Field name	Type	Policy	Template	Description
templateHash	TPM2B_DIGEST	C	C	The digest of the template for the inPublic parameter of an object creation command.
templatePublic	TPM2B_PUBLIC	C	C	The template for the inPublic parameter of an object creation command.

Example:

```
{
        “name”: “MyTemplatePolicy”,
        “description”: “This is the first example policy’s description for humans”,
        “policy”: [
                {
                        “type”: “template”,
                        “templatePublic”: {
                                “hashAlg”: “sha256”,
                                …
                        }
                }
        ]
}
```

4.19 PolicyAuthorizeNv (TPMS_POLICYAUTHORIZENV)

The data type described in Table 26 describes a policy element that is used to defer the authorization of an object to the authorization of an NV index.

Table 26 TPMS_POLICYAUTHORIZENV

Field name	Type	Policy	Template	Description
nvPath	String	-	C	The path to an NV index containing the digest of an authorized policy. An implementation will calculate the actual public area used for the policy.
nvPublic	TPM2B_PUBLIC	X	C	The public area of an NV index containing the digest of an authorized policy.

Example:

```
{
        “name”: “MyAuthorizeNvPolicy”,
        “description”: “This is the first example policy’s description for humans”,
        “policy”: [
                {
                        “type”: “authorizeNv”,
                        “nvPath”: “/nv/myindex”
                },
                …
        ]
}
```

4.20 PolicyAction (TPMS_POLICYACTION)

The data type whose fields are listed in Table 27 describes a policy element that is used as a placeholder for application actions. It can be used to “remind” an application to perform certain operations during the evaluation of a policy. The embedded JSON object is handed back to the application as JSON string in a callback. The following actions are

highly application specific and the contents of that object are ignored by the policy evaluation, other than waiting for the callback to return. In the simplest case the action field can be a plain string or integer.

Table 27 TPMS_POLICYACTIOIN

Field name	Type	Policy	Template	Description
action	JSON object	X	X	The FAPI will return a string representation of the JSON subtree under this key

Example:

```
{
        “name”: “MyActionPolicy”,
        “description”: “This is the first example policy’s description for humans”,
        “policy”: [
                {
                        “type”: “PolicyAction”,
                        “action”: {  “myaction”: “do somethimg” }
                }, {
                        “type”: “PolicyAction”,
                        “action”: “whatever”
                }
        ]
}
```

Appendix – Policy Normal PolicyOr Form

The most general valid TPM policy expression is a tree in which nodes with two edges are policy actions, and *OR* operators are nodes with between two and 8 inputs. The TPM (and this specification) allows arbitrarily branching trees (such trees are generally the most compact). However, this specification recommends an alternative representation of policies as a list-of-lists. This alternate form (referred to in this specification as normal form) is a subset of the general case that is easier to visualize and represent programmatically. Specifically, the objective is to enable implementations to ask a potential user for a branch selection only once per operation, instead of a series of branch selection questions.

Normal-form is an alternate representation of policies based on a list-of-lists. This representation is just as expressive as the general case, but is easier to visualize and create (and also easier to compose policy fragments from third parties). For instance, the following policy expression could be re-written:

General Form

p = A | B & (C | D) | (E & F)

Normal Form

p = A | (B & C) | (B & D) | (E & F)

Any valid TPM policy expression can be written in this normal form through Boolean-algebraic manipulation. Normal form has no bearing on the algorithms and data structures in this specification, but it is anticipated that normal form will be a favored representation for tools and libraries.

Note, of course, that policies can only be transformed to normal form prior to publication and use. This is because, while transformations result in the same authorization rules, the actual policy hash will change.

Appendix – Examples

PolicyPassword

```
{
   “name”: “Password”,
   “description”: “A policy requiring the password for the object”,
   “policy”: [ {
      “type”: “Password”
   } ]
}
```

PolicyPCR and PolicyPassword

```
{
   “name”: “PCRandPassword”,
   “description”: “A policy requiring both certain PCR values and the password for the object”,
   “policy”: [ {
      “type”: “PCR”,
      “pcrs”: [
         { “hashAlg”: “sha1”, pcr: 0, digest: “123abc123abc123abc123abc123abc123abc94” },
         { “hashAlg”: “sha1”, pcr: 2, digest: “defabc123abc123abc123abc123abc123abc94” },
         { “hashAlg”: “sha256”, pcr: 8, digest:
“6946946946946946946946941 23abc123abc123abc123abc123abc123abc94” },
      ]
   }, {
      “type”: “Password”
   } ]
}
```

```
{
   “name”: “PCRandPassword”,
   “description”: “A policy requiring both certain PCR values and  the password for the object”,
   “policy”: [ {
      “type”: “PCR”,
      “currentpcrs”: [
         { “hashAlg”: “sha1”, pcr: 0 },
         { “hashAlg”: “sha1”, pcr: 2},
         { “hashAlg”: “sha256”, pcr: 8 },
      ]
   }, {
      “type”: “Password”
   } ]
}
```

PolicyPCR or PolicyPassword

```
{
   “name”: “PCRorPassword”,
   “description”: “A policy requiring either certain PCR values or the password for the object”,
   “policy”: [ {
      “type”: “or”,
      “branches”: [ {
```

```
        "name": "pcrs",
        "description": "require certain PCR values",
        "policy": [ {
          "type": "PCR",
          "pcrs": [
            { "hashAlg": "sha1", pcr: 0, digest: "123abc123abc123abc123abc123abc123abc94" },
            { "hashAlg": "sha1", pcr: 2, digest: "defabc123abc123abc123abc123abc123abc94" },
            { "hashAlg": "sha256", pcr: 8, digest:
"6946946946946946946946941 23abc123abc123abc123abc123abc123abc94" },
          ]
        } ]
      }, {
        "name": "password",
        "description": "require the password for the object",
        "policy": [ {
          "type": "Password",
          "key": {
        } ]
      } ]
    } ]
}
```

PolicyPassword OR PolicyAuthorize PLUS PolicyAuthorization for PolicyPCR

```
{
  "name": "PCRorPassword",
  "description": "A policy requiring either certain PCR values or the password for the object",
  "policy": [ {
    "type": "or",
    "branches": [ {
      "name": "password",
      "description": "require the password for the object",
      "policy": [ {
        "type": "Password"
      } ]
    }, {
    "type": "or",
    "branches": [ {
      "name": "authorization",
      "description": "link to authorized policies",
      "policy": [ {
        "type": "Authorize",

      } ]
    } ]
  } ]
}
```

PolicySecret tied to Owner Authorization

```
{
  "name": "OwnerSecret",
  "description": "A policy requiring owner authorization",
  "policy": [ {
    "type": "secret",
```



```
      "objectPath": "/HS"
   } ]
}
```

PolicySecret tied to Endorsement Authorization

```
{
   "name": "PrivacySecret",
   "description": "A policy requiring privacy administrator authorization",
   "policy": [ {
      "type": "secret",
      "objectPath": "/HE"
   } ]
}
```

PolicyCommandCode for NV Reading

```
{
   "name": "CommandCodeNVRead",
   "description": "A policy allowing access to NV Read for anyone",
   "policy": [ {
      "type": "commandCode",
      "code": "NV_Read"
   } ]
}
```


Appendix – TPM Policy Schema

The following CDDL schema represents the format for TPM policy specification.

```
; TCG TPM Structures CDDL data definition
; Based on version: Family 2.0, Level 00, Rev 01.38, Sep 29, 2016
; Encoding-agnostic ROOT CDDL definition

; requires: tcg-algorithm-registry.cddl
; requires: tcg-tpm2.0-tss-structures_JSON-postlude.cddl

; START OF TEST ROOT

dummy-root = [ "dummy-root"
             TPM_STRUCTURES,
             dummy-base-array,
           ]

; END OF TEST ROOT

; START OF TEST HEADER

dummy-base-array = [ BYTE,
                     UINT8,
                     INT8,
                     BOOL,
                     UINT16,
                     INT16,
                     UINT32,
                     INT32,
                     UINT64,
                     INT64,
                     dummy-logic-values,
                     dummy-misc-types,
                     dummy-spec-constants,
                     dummy-attribute-structures,
                     dummy-interfaces,
                     TPM_HANDLE,
                     TPM_GENERATED_VALUE,
                     $TPM_ALG_ID,
                     $TPM_ECC_CURVE,
                     $TPM_CC,
                     $TPM_ST,
                     $TPM_CAP,
                     $TPM_PT,
                     $TPM_PT_PCR,
                   ]

dummy-logic-values = [ TRUE,
                       FALSE,
                       YES,
                       NO,
                       SET,
                       CLEAR,
                     ]

dummy-misc-types = [ TPM_ALGORITHM_ID,
```



```
                TPM_MODIFIER_INDICATOR,
                TPM_AUTHORIZATION_SIZE,
                TPM_PARAMETER_SIZE,
                TPM_KEY_SIZE,
                TPM_KEY_BITS,
              ]

dummy-spec-constants = [ TPM_SPEC_FAMILY,
                  TPM_SPEC_LEVEL,
                  TPM_SPEC_VERSION,
                  TPM_SPEC_YEAR,
                  TPM_SPEC_DAY_OF_YEAR,
                ]
dummy-attribute-structures = [ TPMA_ALGORITHM,
                      TPMA_OBJECT,
                      TPMA_SESSION,
                      TPMA_LOCALITY,
                      TPMA_PERMANENT,
                      TPMA_STARTUP_CLEAR,
                      TPMA_MEMORY,
                      TPMA_CC,
                      TPMA_MODES,
                    ]

dummy-interfaces = [ TPMI_YES_NO,
                TPMI_DH_OBJECT,
                TPMI_DH_PARENT,
                TPMI_DH_PERSISTENT,
                TPMI_DH_ENTITY,
                TPMI_DH_PCR,
                TPMI_SH_AUTH_SESSION,
                TPMI_SH_HMAC,
                TPMI_SH_POLICY,
                TPMI_DH_CONTEXT,
                TPMI_RH_HIERARCHY,
                TPMI_RH_ENABLES,
                TPMI_RH_HIERARCHY_AUTH,
                TPMI_RH_PLATFORM,
                TPMI_RH_OWNER,
                TPMI_RH_ENDORSEMENT,
                TPMI_RH_PROVISION,
                TPMI_RH_CLEAR,
                TPMI_RH_NV_AUTH,
                TPMI_RH_LOCKOUT,
                TPMI_RH_NV_INDEX,
                TPMI_ALG_HASH,
                TPMI_ALG_ASYM,
                TPMI_ALG_SYM,
                TPMI_ALG_SYM_OBJECT,
                TPMI_ALG_SYM_MODE,
                TPMI_ALG_KDF,
                TPMI_ALG_SIG_SCHEME,
                TPMI_ECC_KEY_EXCHANGE,
                TPMI_ST_COMMAND,
              ]
```

```
; TPM structures to be tested
; in order of appearance :)

TPM_STRUCTURES = [ "tpm_structures",
                   TPMS_EMPTY,
                   TPMS_ALGORITHM_DESCRIPTION,
                   TPMT_HA,
                   TPM2B_DIGEST,
                   TPM2B_DATA,
                   TPM2B_NONCE,
                   TPM2B_AUTH,
                   TPM2B_OPERAND,
                   TPM2B_EVENT,
                   TPM2B_MAX_BUFFER,
                   TPM2B_MAX_NV_BUFFER,
                   TPM2B_TIMEOUT,
                   TPM2B_IV,
                   TPM2B_NAME,
                   TPMS_PCR_SELECT,
                   TPMS_PCR_SELECTION,
                   TPMT_TK_CREATION,
                   TPMT_TK_VERIFIED,
                   TPMT_TK_AUTH,
                   TPMT_TK_HASHCHECK,
                   TPMS_ALG_PROPERTY,
                   TPMS_TAGGED_PROPERTY,
                   TPMS_TAGGED_PCR_SELECT,
                   TPMS_TAGGED_POLICY,
                   TPML_CC,
                   TPML_CCA,
                   TPML_ALG,
                   TPML_HANDLE,
                   TPML_DIGEST,
                   TPML_DIGEST_VALUES,
                   TPML_PCR_SELECTION,
                   TPML_ALG_PROPERTY,
                   TPML_TAGGED_TPM_PROPERTY,
                   TPML_TAGGED_PCR_PROPERTY,
                   TPML_ECC_CURVE,
                   TPML_TAGGED_POLICY,
                   TPMS_CAPABILITY_DATA,
                   TPMS_CLOCK_INFO,
                   TPMS_TIME_INFO,
                   TPMS_TIME_ATTEST_INFO,
                   TPMS_CERTIFY_INFO,
                   TPMS_QUOTE_INFO,
                   TPMS_COMMAND_AUDIT_INFO,
                   TPMS_SESSION_AUDIT_INFO,
                   TPMS_CREATION_INFO,
                   TPMS_NV_CERTIFY_INFO,
                   TPM2B_ATTEST,
                   TPMS_AUTH_COMMAND,
                   TPMS_AUTH_RESPONSE,
                 ]

; END OF TEST HEADER
```

```
; START OF TCG TPM STRUCTURES CDDL BODY
; (intended to be encoding-agnostic)

; Primitive Types (5.1)

BYTE = uint .size 1
UINT8 = uint8_t
INT8 = int8_t
BOOL = 0x0..0x1
UINT16 = uint16_t
INT16 = int16_t
UINT32 = uint32_t
INT32 = int32_t
UINT64 = uint64_t
INT64 = int64_t

; type mapping for Base Types
; PLEASE NOTE:
; there is no .size control for int
; due to int = uint/nint

uint8_t = uint .size 1
int8_t = -0x7F..0x7F
uint16_t = uint .size 2
int16_t = -0x7FFF..0x7FFF
uint32_t = uint .size 4
int32_t = -0x7FFFFFFF..0x7FFFFFFF
uint64_t = uint .size 8
int64_t = -0x7FFFFFFFFFFFFFFF..0x7FFFFFFFFFFFFFFF

; Logic Values (5.2)

TRUE = 1
FALSE = 0
YES = 1
NO = 0
SET = 1
CLEAR = 0

; Misc Types (5.3)

TPM_ALGORITHM_ID = UINT32
TPM_MODIFIER_INDICATOR = UINT32
TPM_AUTHORIZATION_SIZE = UINT32
TPM_PARAMETER_SIZE = UINT32
TPM_KEY_SIZE = UINT16
TPM_KEY_BITS = UINT16

; Misc Super Types (FIXME)

TPM_HANDLE = UINT32

; TPM Spec Constants (6.1)

TPM_SPEC_FAMILY = 0x322E3000
```

```
TPM_SPEC_LEVEL = 0x0
TPM_SPEC_VERSION = 138
TPM_SPEC_YEAR = 2016
TPM_SPEC_DAY_OF_YEAR = 273

; TPM Generated Constants (6.2)

TPM_GENERATED_VALUE = 0xFF544347

; TPM Algorithm Identifier (6.3)
; (effectively TCG Algorithm Registry, Section 4)

; please include tcg-algorithm-registry.cddl

;TPM ECC Curve Constants (6.4)
; (effectively TCG Algorithm Registry, Section 5)

; please include tcg-algorithm-registry.cddl

; TPM Command Codes (6.5)

$TPM_CC /= TPM_CC_FIRST
$TPM_CC /= TPM_CC_NV_UndefineSpaceSpecial
$TPM_CC /= TPM_CC_EvictControl
$TPM_CC /= TPM_CC_HierarchyControl
$TPM_CC /= TPM_CC_NV_UndefineSpace
$TPM_CC /= TPM_CC_ChangeEPS
$TPM_CC /= TPM_CC_ChangePPS
$TPM_CC /= TPM_CC_Clear
$TPM_CC /= TPM_CC_ClearControl
$TPM_CC /= TPM_CC_ClockSet
$TPM_CC /= TPM_CC_HierarchyChangeAuth
$TPM_CC /= TPM_CC_NV_DefineSpace
$TPM_CC /= TPM_CC_PCR_Allocate
$TPM_CC /= TPM_CC_PCR_SetAuthPolicy
$TPM_CC /= TPM_CC_PP_Commands
$TPM_CC /= TPM_CC_SetPrimaryPolicy
$TPM_CC /= TPM_CC_FieldUpgradeStart
$TPM_CC /= TPM_CC_ClockRateAdjust
$TPM_CC /= TPM_CC_CreatePrimary
$TPM_CC /= TPM_CC_NV_GlobalWriteLock
$TPM_CC /= TPM_CC_GetCommandAuditDigest
$TPM_CC /= TPM_CC_NV_Increment
$TPM_CC /= TPM_CC_NV_SetBits
$TPM_CC /= TPM_CC_NV_Extend
$TPM_CC /= TPM_CC_NV_Write
$TPM_CC /= TPM_CC_NV_WriteLock
$TPM_CC /= TPM_CC_DictionaryAttackLockReset
$TPM_CC /= TPM_CC_DictionaryAttackParameters
$TPM_CC /= TPM_CC_NV_ChangeAuth
$TPM_CC /= TPM_CC_PCR_Event
$TPM_CC /= TPM_CC_PCR_Reset
$TPM_CC /= TPM_CC_SequenceComplete
$TPM_CC /= TPM_CC_SetAlgorithmSet
$TPM_CC /= TPM_CC_SetCommandCodeAuditStatus
$TPM_CC /= TPM_CC_FieldUpgradeData
```

$TPM_CC /= TPM_CC_IncrementalSelfTest
$TPM_CC /= TPM_CC_SelfTest
$TPM_CC /= TPM_CC_Startup
$TPM_CC /= TPM_CC_Shutdown
$TPM_CC /= TPM_CC_StirRandom
$TPM_CC /= TPM_CC_ActivateCredential
$TPM_CC /= TPM_CC_Certify
$TPM_CC /= TPM_CC_PolicyNV
$TPM_CC /= TPM_CC_CertifyCreation
$TPM_CC /= TPM_CC_Duplicate
$TPM_CC /= TPM_CC_GetTime
$TPM_CC /= TPM_CC_GetSessionAuditDigest
$TPM_CC /= TPM_CC_NV_Read
$TPM_CC /= TPM_CC_NV_ReadLock
$TPM_CC /= TPM_CC_ObjectChangeAuth
$TPM_CC /= TPM_CC_PolicySecret
$TPM_CC /= TPM_CC_Rewrap
$TPM_CC /= TPM_CC_Create
$TPM_CC /= TPM_CC_ECDH_ZGen
$TPM_CC /= TPM_CC_HMAC
$TPM_CC /= TPM_CC_Import
$TPM_CC /= TPM_CC_Load
$TPM_CC /= TPM_CC_Quote
$TPM_CC /= TPM_CC_RSA_Decrypt
$TPM_CC /= TPM_CC_HMAC_Start
$TPM_CC /= TPM_CC_SequenceUpdate
$TPM_CC /= TPM_CC_Sign
$TPM_CC /= TPM_CC_Unseal
$TPM_CC /= TPM_CC_PolicySigned
$TPM_CC /= TPM_CC_ContextLoad
$TPM_CC /= TPM_CC_ContextSave
$TPM_CC /= TPM_CC_ECDH_KeyGen
$TPM_CC /= TPM_CC_EncryptDecrypt
$TPM_CC /= TPM_CC_FlushContext
$TPM_CC /= TPM_CC_LoadExternal
$TPM_CC /= TPM_CC_MakeCredential
$TPM_CC /= TPM_CC_NV_ReadPublic
$TPM_CC /= TPM_CC_PolicyAuthorize
$TPM_CC /= TPM_CC_PolicyAuthValue
$TPM_CC /= TPM_CC_PolicyCommandCode
$TPM_CC /= TPM_CC_PolicyCounterTimer
$TPM_CC /= TPM_CC_PolicyCpHash
$TPM_CC /= TPM_CC_PolicyLocality
$TPM_CC /= TPM_CC_PolicyNameHash
$TPM_CC /= TPM_CC_PolicyOR
$TPM_CC /= TPM_CC_PolicyTicket
$TPM_CC /= TPM_CC_ReadPublic
$TPM_CC /= TPM_CC_RSA_Encrypt
$TPM_CC /= TPM_CC_StartAuthSession
$TPM_CC /= TPM_CC_VerifySignature
$TPM_CC /= TPM_CC_ECC_Parameters
$TPM_CC /= TPM_CC_FirmwareRead
$TPM_CC /= TPM_CC_GetCapability
$TPM_CC /= TPM_CC_GetRandom
$TPM_CC /= TPM_CC_GetTestResult
$TPM_CC /= TPM_CC_Hash

```
$TPM_CC /= TPM_CC_PCR_Read
$TPM_CC /= TPM_CC_PolicyPCR
$TPM_CC /= TPM_CC_PolicyRestart
$TPM_CC /= TPM_CC_ReadClock
$TPM_CC /= TPM_CC_PCR_Extend
$TPM_CC /= TPM_CC_PCR_SetAuthValue
$TPM_CC /= TPM_CC_NV_Certify

TPM_CC_FIRST = 0x0000011F
TPM_CC_NV_UndefineSpaceSpecial = 0x0000011F
TPM_CC_EvictControl = 0x00000120
TPM_CC_HierarchyControl = 0x00000121
TPM_CC_NV_UndefineSpace = 0x00000122
TPM_CC_ChangeEPS = 0x00000124
TPM_CC_ChangePPS = 0x00000125
TPM_CC_Clear = 0x00000126
TPM_CC_ClearControl = 0x00000127
TPM_CC_ClockSet = 0x00000128
TPM_CC_HierarchyChangeAuth = 0x00000129
TPM_CC_NV_DefineSpace = 0x0000012A
TPM_CC_PCR_Allocate = 0x0000012B
TPM_CC_PCR_SetAuthPolicy = 0x0000012C
TPM_CC_PP_Commands = 0x0000012D
TPM_CC_SetPrimaryPolicy = 0x0000012E
TPM_CC_FieldUpgradeStart = 0x0000012F
TPM_CC_ClockRateAdjust = 0x00000130
TPM_CC_CreatePrimary = 0x00000131
TPM_CC_NV_GlobalWriteLock = 0x00000132
TPM_CC_GetCommandAuditDigest = 0x00000133
TPM_CC_NV_Increment = 0x00000134
TPM_CC_NV_SetBits = 0x00000135
TPM_CC_NV_Extend = 0x00000136
TPM_CC_NV_Write = 0x00000137
TPM_CC_NV_WriteLock = 0x00000138
TPM_CC_DictionaryAttackLockReset = 0x00000139
TPM_CC_DictionaryAttackParameters = 0x0000013A
TPM_CC_NV_ChangeAuth = 0x0000013B
TPM_CC_PCR_Event = 0x0000013C
TPM_CC_PCR_Reset = 0x0000013D
TPM_CC_SequenceComplete = 0x0000013E
TPM_CC_SetAlgorithmSet = 0x0000013F
TPM_CC_SetCommandCodeAuditStatus = 0x00000140
TPM_CC_FieldUpgradeData = 0x00000141
TPM_CC_IncrementalSelfTest = 0x00000142
TPM_CC_SelfTest = 0x00000143
TPM_CC_Startup = 0x00000144
TPM_CC_Shutdown = 0x00000145
TPM_CC_StirRandom = 0x00000146
TPM_CC_ActivateCredential = 0x00000147
TPM_CC_Certify = 0x00000148
TPM_CC_PolicyNV = 0x00000149
TPM_CC_CertifyCreation = 0x0000014A
TPM_CC_Duplicate = 0x0000014B
TPM_CC_GetTime = 0x0000014C
TPM_CC_GetSessionAuditDigest = 0x0000014D
TPM_CC_NV_Read = 0x0000014E
```

```
TPM_CC_NV_ReadLock = 0x0000014F
TPM_CC_ObjectChangeAuth = 0x00000150
TPM_CC_PolicySecret = 0x00000151
TPM_CC_Rewrap = 0x00000152
TPM_CC_Create = 0x00000153
TPM_CC_ECDH_ZGen = 0x00000154
TPM_CC_HMAC = 0x00000155
TPM_CC_Import = 0x00000156
TPM_CC_Load = 0x00000157
TPM_CC_Quote = 0x00000158
TPM_CC_RSA_Decrypt = 0x00000159
TPM_CC_HMAC_Start = 0x0000015B
TPM_CC_SequenceUpdate = 0x0000015C
TPM_CC_Sign = 0x0000015D
TPM_CC_Unseal = 0x0000015E
TPM_CC_PolicySigned = 0x00000160
TPM_CC_ContextLoad = 0x00000161
TPM_CC_ContextSave = 0x00000162
TPM_CC_ECDH_KeyGen = 0x00000163
TPM_CC_EncryptDecrypt = 0x00000164
TPM_CC_FlushContext = 0x00000165
TPM_CC_LoadExternal = 0x00000167
TPM_CC_MakeCredential = 0x00000168
TPM_CC_NV_ReadPublic = 0x00000169
TPM_CC_PolicyAuthorize = 0x0000016A
TPM_CC_PolicyAuthValue = 0x0000016B
TPM_CC_PolicyCommandCode = 0x0000016C
TPM_CC_PolicyCounterTimer = 0x0000016D
TPM_CC_PolicyCpHash = 0x0000016E
TPM_CC_PolicyLocality = 0x0000016F
TPM_CC_PolicyNameHash = 0x00000170
TPM_CC_PolicyOR = 0x00000171
TPM_CC_PolicyTicket = 0x00000172
TPM_CC_ReadPublic = 0x00000173
TPM_CC_RSA_Encrypt = 0x00000174
TPM_CC_StartAuthSession = 0x00000176
TPM_CC_VerifySignature = 0x00000177
TPM_CC_ECC_Parameters = 0x00000178
TPM_CC_FirmwareRead = 0x00000179
TPM_CC_GetCapability = 0x0000017A
TPM_CC_GetRandom = 0x0000017B
TPM_CC_GetTestResult = 0x0000017C
TPM_CC_Hash = 0x0000017D
TPM_CC_PCR_Read = 0x0000017E
TPM_CC_PolicyPCR = 0x0000017F
TPM_CC_PolicyRestart = 0x00000180
TPM_CC_ReadClock = 0x00000181
TPM_CC_PCR_Extend = 0x00000182
TPM_CC_PCR_SetAuthValue = 0x00000183
TPM_CC_NV_Certify = 0x00000184

; TPM Structure Tags (6.9)

$TPM_ST /= TPM_ST_RSP_COMMAND
$TPM_ST /= TPM_ST_NULL
$TPM_ST /= TPM_ST_NO_SESSIONS
```



```
$TPM_ST /= TPM_ST_SESSIONS
$TPM_ST /= TPM_ST_reserved
$TPM_ST /= TPM_ST_ATTEST_NV
$TPM_ST /= TPM_ST_ATTEST_COMMAND_AUDIT
$TPM_ST /= TPM_ST_ATTEST_SESSION_AUDIT
$TPM_ST /= TPM_ST_ATTEST_CERTIFY
$TPM_ST /= TPM_ST_ATTEST_QUOTE
$TPM_ST /= TPM_ST_ATTEST_TIME
$TPM_ST /= TPM_ST_ATTEST_CREATION
$TPM_ST /= TPM_ST_CREATION
$TPM_ST /= TPM_ST_VERIFIED
$TPM_ST /= TPM_ST_AUTH_SECRET
$TPM_ST /= TPM_ST_HASHCHECK
$TPM_ST /= TPM_ST_AUTH_SIGNED
$TPM_ST /= TPM_ST_FU_MANIFEST

TPM_ST_RSP_COMMAND = 0x00C4
TPM_ST_NULL = 0X8000
TPM_ST_NO_SESSIONS = 0x8001
TPM_ST_SESSIONS = 0x8002
TPM_ST_reserved = 0x8003 / 0x8004 ; see table 19
TPM_ST_ATTEST_NV = 0x8014
TPM_ST_ATTEST_COMMAND_AUDIT = 0x8015
TPM_ST_ATTEST_SESSION_AUDIT = 0x8016
TPM_ST_ATTEST_CERTIFY = 0x8016
TPM_ST_ATTEST_QUOTE = 0x8018
TPM_ST_ATTEST_TIME = 0x8019
TPM_ST_ATTEST_CREATION = 0x801A
TPM_ST_CREATION = 0x8021
TPM_ST_VERIFIED = 0x8022
TPM_ST_AUTH_SECRET = 0x8023
TPM_ST_HASHCHECK = 0x8024
TPM_ST_AUTH_SIGNED = 0x8025
TPM_ST_FU_MANIFEST = 0x8029

; Capabilities (6.12)

$TPM_CAP /= TPM_CAP_FIRST
$TPM_CAP /= TPM_CAP_ALGS
$TPM_CAP /= TPM_CAP_HANDLES
$TPM_CAP /= TPM_CAP_COMMANDS
$TPM_CAP /= TPM_CAP_PP_COMMANDS
$TPM_CAP /= TPM_CAP_AUDIT_COMMANDS
$TPM_CAP /= TPM_CAP_PCRS
$TPM_CAP /= TPM_CAP_TPM_PROPERTIES
$TPM_CAP /= TPM_CAP_PCR_PROPERTIES
$TPM_CAP /= TPM_CAP_ECC_CURVES
$TPM_CAP /= TPM_CAP_AUTH_POLICIES
$TPM_CAP /= TPM_CAP_LAST
$TPM_CAP /= TPM_CAP_VENDOR_PROPERTY

TPM_CAP_FIRST = 0x00000000
TPM_CAP_ALGS = 0x00000000
TPM_CAP_HANDLES = 0x00000001
TPM_CAP_COMMANDS = 0x00000002
TPM_CAP_PP_COMMANDS = 0x00000003
```



```
TPM_CAP_AUDIT_COMMANDS = 0x00000004
TPM_CAP_PCRS = 0x00000005
TPM_CAP_TPM_PROPERTIES = 0x00000006
TPM_CAP_PCR_PROPERTIES = 0x00000007
TPM_CAP_ECC_CURVES = 0x00000008
TPM_CAP_AUTH_POLICIES = 0x00000009
TPM_CAP_LAST = 0x00000009
TPM_CAP_VENDOR_PROPERTY = 0x00000100

; Property Tags (6.13)

$TPM_PT /= TPM_PT_NONE
$TPM_PT /= PT_GROUP
$TPM_PT /= PT_FIXED
$TPM_PT /= TPM_PT_FAMILY_INDICATOR
$TPM_PT /= TPM_PT_LEVEL
$TPM_PT /= TPM_PT_REVISION
$TPM_PT /= TPM_PT_DAY_OF_YEAR
$TPM_PT /= TPM_PT_YEAR
$TPM_PT /= TPM_PT_MANUFACTURER
$TPM_PT /= TPM_PT_VENDOR_STRING_1
$TPM_PT /= TPM_PT_VENDOR_STRING_2
$TPM_PT /= TPM_PT_VENDOR_STRING_3
$TPM_PT /= TPM_PT_VENDOR_STRING_4
$TPM_PT /= TPM_PT_VENDOR_TPM_TYPE
$TPM_PT /= TPM_PT_FIRMWARE_VERSION_1
$TPM_PT /= TPM_PT_FIRMWARE_VERSION_2
$TPM_PT /= TPM_PT_INPUT_BUFFER
$TPM_PT /= TPM_PT_HR_TRANSIENT_MIN
$TPM_PT /= TPM_PT_HR_PERSISTENT_MIN
$TPM_PT /= TPM_PT_HR_LOADED_MIN
$TPM_PT /= TPM_PT_ACTIVE_SESSIONS_MAX
$TPM_PT /= TPM_PT_PCR_COUNT
$TPM_PT /= TPM_PT_PCR_SELECT_MIN
$TPM_PT /= TPM_PT_CONTEXT_GAP_MAX
$TPM_PT /= TPM_PT_skipped
$TPM_PT /= TPM_PT_NV_COUNTERS_MAX
$TPM_PT /= TPM_PT_NV_INDEX_MAX
$TPM_PT /= TPM_PT_MEMORY
$TPM_PT /= TPM_PT_CLOCK_UPDATE
$TPM_PT /= TPM_PT_CONTEXT_HASH
$TPM_PT /= TPM_PT_CONTEXT_SYM
$TPM_PT /= TPM_PT_CONTEXT_SYM_SIZE
$TPM_PT /= TPM_PT_ORDERLY_COUNT
$TPM_PT /= TPM_PT_MAX_COMMAND_SIZE
$TPM_PT /= TPM_PT_MAX_RESPONSE_SIZE
$TPM_PT /= TPM_PT_MAX_DIGEST
$TPM_PT /= TPM_PT_MAX_OBJECT_CONTEXT
$TPM_PT /= TPM_PT_MAX_SESSION_CONTEXT
$TPM_PT /= TPM_PT_PS_FAMILY_INDICATOR
$TPM_PT /= TPM_PT_PS_LEVEL
$TPM_PT /= TPM_PT_PS_REVISION
$TPM_PT /= TPM_PT_PS_DAY_OF_YEAR
$TPM_PT /= TPM_PT_PS_YEAR
$TPM_PT /= TPM_PT_SPLIT_MAX
$TPM_PT /= TPM_PT_TOTAL_COMMANDS
```

```
$TPM_PT /= TPM_PT_LIBRARY_COMMANDS
$TPM_PT /= TPM_PT_VENDOR_COMMANDS
$TPM_PT /= TPM_PT_NV_BUFFER_MAX
$TPM_PT /= TPM_PT_MODES
$TPM_PT /= TPM_PT_MAX_CAP_BUFFER
$TPM_PT /= PT_VAR
$TPM_PT /= TPM_PT_PERMANENT
$TPM_PT /= TPM_PT_STARTUP_CLEAR
$TPM_PT /= TPM_PT_HR_NV_INDEX
$TPM_PT /= TPM_PT_HR_LOADED
$TPM_PT /= TPM_PT_HR_LOADED_AVAIL
$TPM_PT /= TPM_PT_HR_ACTIVE
$TPM_PT /= TPM_PT_HR_ACTIVE_AVAIL
$TPM_PT /= TPM_PT_HR_TRANSIENT_AVAIL
$TPM_PT /= TPM_PT_HR_PERSISTENT
$TPM_PT /= TPM_PT_HR_PERSISTENT_AVAIL
$TPM_PT /= TPM_PT_NV_COUNTERS
$TPM_PT /= TPM_PT_NV_COUNTERS_AVAIL
$TPM_PT /= TPM_PT_ALGORITHM_SET
$TPM_PT /= TPM_PT_LOADED_CURVES
$TPM_PT /= TPM_PT_LOCKOUT_COUNTER
$TPM_PT /= TPM_PT_MAX_AUTH_FAIL
$TPM_PT /= TPM_PT_LOCKOUT_INTERVAL
$TPM_PT /= TPM_PT_LOCKOUT_RECOVERY
$TPM_PT /= TPM_PT_NV_WRITE_RECOVERY
$TPM_PT /= TPM_PT_AUDIT_COUNTER_0
$TPM_PT /= TPM_PT_AUDIT_COUNTER_1

TPM_PT_NONE = 0x00000100
PT_GROUP = 0x00000100
PT_FIXED = 0x00000100
TPM_PT_FAMILY_INDICATOR = 0x00000100
TPM_PT_LEVEL = 0x00000101
TPM_PT_REVISION = 0x00000102
TPM_PT_DAY_OF_YEAR = 0x00000103
TPM_PT_YEAR = 0x00000104
TPM_PT_MANUFACTURER = 0x00000105
TPM_PT_VENDOR_STRING_1 = 0x00000106
TPM_PT_VENDOR_STRING_2 = 0x00000107
TPM_PT_VENDOR_STRING_3 = 0x00000108
TPM_PT_VENDOR_STRING_4 = 0x00000109
TPM_PT_VENDOR_TPM_TYPE = 0x0000010A
TPM_PT_FIRMWARE_VERSION_1 = 0x0000010B
TPM_PT_FIRMWARE_VERSION_2  = 0x0000010C
TPM_PT_INPUT_BUFFER = 0x0000010D
TPM_PT_HR_TRANSIENT_MIN = 0x0000010E
TPM_PT_HR_PERSISTENT_MIN = 0x0000010F
TPM_PT_HR_LOADED_MIN = 0x00000110
TPM_PT_ACTIVE_SESSIONS_MAX = 0x00000111
TPM_PT_PCR_COUNT = 0x00000112
TPM_PT_PCR_SELECT_MIN = 0x00000113
TPM_PT_CONTEXT_GAP_MAX = 0x00000114
TPM_PT_skipped = 0x00000115
TPM_PT_NV_COUNTERS_MAX = 0x00000116
TPM_PT_NV_INDEX_MAX = 0x00000117
TPM_PT_MEMORY = 0x00000118
```

```
TPM_PT_CLOCK_UPDATE = 0x00000119
TPM_PT_CONTEXT_HASH = 0x0000011A
TPM_PT_CONTEXT_SYM = 0x0000011B
TPM_PT_CONTEXT_SYM_SIZE = 0x0000011C
TPM_PT_ORDERLY_COUNT = 0x0000011D
TPM_PT_MAX_COMMAND_SIZE = 0x0000011E
TPM_PT_MAX_RESPONSE_SIZE = 0x0000011F
TPM_PT_MAX_DIGEST = 0x00000120
TPM_PT_MAX_OBJECT_CONTEXT = 0x00000121
TPM_PT_MAX_SESSION_CONTEXT = 0x00000122
TPM_PT_PS_FAMILY_INDICATOR = 0x00000123
TPM_PT_PS_LEVEL = 0x00000124
TPM_PT_PS_REVISION = 0x00000125
TPM_PT_PS_DAY_OF_YEAR = 0x00000126
TPM_PT_PS_YEAR = 0x00000127
TPM_PT_SPLIT_MAX = 0x00000128
TPM_PT_TOTAL_COMMANDS = 0x00000129
TPM_PT_LIBRARY_COMMANDS = 0x0000012A
TPM_PT_VENDOR_COMMANDS = 0x0000012B
TPM_PT_NV_BUFFER_MAX = 0x0000012C
TPM_PT_MODES = 0x0000012D
TPM_PT_MAX_CAP_BUFFER = 0x0000012E
PT_VAR = 0x00000200
TPM_PT_PERMANENT = 0x00000200
TPM_PT_STARTUP_CLEAR = 0x00000201
TPM_PT_HR_NV_INDEX = 0x00000202
TPM_PT_HR_LOADED = 0x00000203
TPM_PT_HR_LOADED_AVAIL = 0x00000204
TPM_PT_HR_ACTIVE = 0x00000205
TPM_PT_HR_ACTIVE_AVAIL = 0x00000206
TPM_PT_HR_TRANSIENT_AVAIL = 0x00000207
TPM_PT_HR_PERSISTENT = 0x00000208
TPM_PT_HR_PERSISTENT_AVAIL = 0x00000209
TPM_PT_NV_COUNTERS = 0x0000020A
TPM_PT_NV_COUNTERS_AVAIL = 0x0000020B
TPM_PT_ALGORITHM_SET = 0x0000020C
TPM_PT_LOADED_CURVES = 0x0000020D
TPM_PT_LOCKOUT_COUNTER = 0x0000020E
TPM_PT_MAX_AUTH_FAIL = 0x0000020F
TPM_PT_LOCKOUT_INTERVAL = 0x00000210
TPM_PT_LOCKOUT_RECOVERY = 0x00000211
TPM_PT_NV_WRITE_RECOVERY = 0x00000212
TPM_PT_AUDIT_COUNTER_0 = 0x00000213
TPM_PT_AUDIT_COUNTER_1 = 0x00000214

; PCR Property Tags (6.13)

$TPM_PT_PCR /= TPM_PT_PCR_FIRST
$TPM_PT_PCR /= TPM_PT_PCR_SAVE
$TPM_PT_PCR /= TPM_PT_PCR_EXTEND_L0
$TPM_PT_PCR /= TPM_PT_PCR_RESET_L0
$TPM_PT_PCR /= TPM_PT_PCR_EXTEND_L1
$TPM_PT_PCR /= TPM_PT_PCR_RESET_L1
$TPM_PT_PCR /= TPM_PT_PCR_EXTEND_L2
$TPM_PT_PCR /= TPM_PT_PCR_RESET_L2
$TPM_PT_PCR /= TPM_PT_PCR_EXTEND_L3
```

```
$TPM_PT_PCR /= TPM_PT_PCR_RESET_L3
$TPM_PT_PCR /= TPM_PT_PCR_EXTEND_L4
$TPM_PT_PCR /= TPM_PT_PCR_RESET_L4
$TPM_PT_PCR /= TPM_PT_PCR_NO_INCREMENT
$TPM_PT_PCR /= TPM_PT_PCR_DRTM_RESET
$TPM_PT_PCR /= TPM_PT_PCR_POLICY
$TPM_PT_PCR /= TPM_PT_PCR_AUTH
$TPM_PT_PCR /= TPM_PT_PCR_LAST

TPM_PT_PCR_FIRST = 0x00000000
TPM_PT_PCR_SAVE = 0x00000000
TPM_PT_PCR_EXTEND_L0 = 0x00000001
TPM_PT_PCR_RESET_L0 = 0x00000002
TPM_PT_PCR_EXTEND_L1 = 0x00000003
TPM_PT_PCR_RESET_L1 = 0x00000004
TPM_PT_PCR_EXTEND_L2 = 0x00000005
TPM_PT_PCR_RESET_L2 = 0x00000006
TPM_PT_PCR_EXTEND_L3 = 0x00000007
TPM_PT_PCR_RESET_L3 = 0x00000008
TPM_PT_PCR_EXTEND_L4 = 0x00000009
TPM_PT_PCR_RESET_L4 = 0x0000000A
TPM_PT_PCR_NO_INCREMENT = 0x00000011
TPM_PT_PCR_DRTM_RESET = 0x00000012
TPM_PT_PCR_POLICY = 0x00000013
TPM_PT_PCR_AUTH = 0x00000014
TPM_PT_PCR_LAST = 0x00000014 ; is this correct? could also be 0x00000214

; Attributes Structures (8)

TPMA_ALGORITHM = 0x0..0x7FF ; effectively UINT32, bits 11-31 are reserved, hence bits 0-10
TPMA_OBJECT = 0x0..0x7FFF; effectively UINT32, bits 19-31 are reserved, hence bits 0-18
TPMA_SESSION = BYTE
TPMA_LOCALITY = BYTE
TPMA_PERMANENT = 0x0..0x7FF ; effectively UINT32, bits 11-31 are reserved, hence bits 0-10
TPMA_STARTUP_CLEAR = UINT32
TPMA_MEMORY = 0x0..0x7 ; effectively UINT32, bits 3-31 are reserved, hence bits 0-2
TPMA_CC = { TPMA_CC-commandIndex,
          TPMA_CC-nv,
          TPMA_CC-extensive,
          TPMA_CC-flushed,
          TPMA_CC-cHandles,
          TPMA_CC-rHandle,
          TPMA_CC-V, ; if this is SET, the pre-defined TPM_CC rule cannot be used reliably
        }
TPMA_MODES = 0x1 ; effectively UINT32, bits 1-31 are reserved, hence bit 0, always set? FIXME

; TPM Interface Types (9)

TPMI_YES_NO = 0x0..0x1 ; effectively BYTE, but actually either 0 or 1
TPMI_DH_OBJECT = TPM_HANDLE
TPMI_DH_PARENT = TPM_HANDLE
TPMI_DH_PERSISTENT = TPM_HANDLE
TPMI_DH_ENTITY = TPM_HANDLE
TPMI_DH_PCR = TPM_HANDLE
TPMI_SH_AUTH_SESSION = TPM_HANDLE
TPMI_SH_HMAC = TPM_HANDLE
```

```
TPMI_SH_POLICY = TPM_HANDLE
TPMI_DH_CONTEXT = TPM_HANDLE
TPMI_RH_HIERARCHY = TPM_HANDLE
TPMI_RH_ENABLES = TPM_HANDLE
TPMI_RH_HIERARCHY_AUTH = TPM_HANDLE
TPMI_RH_PLATFORM = TPM_HANDLE
TPMI_RH_OWNER = TPM_HANDLE
TPMI_RH_ENDORSEMENT = TPM_HANDLE
TPMI_RH_PROVISION = TPM_HANDLE
TPMI_RH_CLEAR = TPM_HANDLE
TPMI_RH_NV_AUTH = TPM_HANDLE
TPMI_RH_LOCKOUT = TPM_HANDLE
TPMI_RH_NV_INDEX = TPM_HANDLE
TPMI_ALG_HASH = $TPM_ALG_ID
TPMI_ALG_ASYM = $TPM_ALG_ID
TPMI_ALG_SYM = $TPM_ALG_ID
TPMI_ALG_SYM_OBJECT = $TPM_ALG_ID
TPMI_ALG_SYM_MODE = $TPM_ALG_ID
TPMI_ALG_KDF = $TPM_ALG_ID
TPMI_ALG_SIG_SCHEME = $TPM_ALG_ID
TPMI_ECC_KEY_EXCHANGE = $TPM_ALG_ID
TPMI_ST_COMMAND = $TPM_ST

; TPM Structures (10)

TPMS_EMPTY = {}

TPMS_ALGORITHM_DESCRIPTION = { TPMS_ALGORITHM_DESCRIPTION-alg,
                    TPMS_ALGORITHM_DESCRIPTION-attributes,
                  }

; Hash/Digest Structures (10.3)

TPMT_HA = { TPMU_HA-hashAlg,
        TPMU_HA-digest,
      }

; Sized Buffers (10.4)

TPM2B_DIGEST = uint16-bytes
TPM2B_DATA = uint16-bytes
TPM2B_NONCE = TPM2B_DIGEST
TPM2B_AUTH = TPM2B_DIGEST
TPM2B_OPERAND = TPM2B_DIGEST
TPM2B_EVENT = oneK-bytes
TPM2B_MAX_BUFFER = uint16-bytes
TPM2B_MAX_NV_BUFFER = uint16-bytes
TPM2B_TIMEOUT = TPM2B_DIGEST
TPM2B_IV = uint16-bytes

; Name Structures (10.5)

TPMU_NAME = TPMU_NAME-digest-entry / TPMU_NAME-handle-entry
TPM2B_NAME = TPMU_NAME
TPMU_NAME-digest-entry = { TPMU_NAME-digest }
TPMU_NAME-handle-entry = { TPMU_NAME-handle }
```

```
; PCR Structures (10.6)

TPMS_PCR_SELECT = { TPMS_PCR_SELECT-pcrSelect }
TPMS_PCR_SELECTION = { TPMS_PCR_SELECTION-hash,
                       TPMS_PCR_SELECTION-pcrSelect,
                     }

; Ticket Structures (10.7)

TPMT_TK_CREATION = { TPMT_TK_CREATION-tag,
                     TPMT_TK-hierarchy,
                     TPMT_TK-digest,
                   }

TPMT_TK_VERIFIED = { TPMT_TK_VERIFIED-tag,
                     TPMT_TK-hierarchy,
                     TPMT_TK-digest,
                   }

TPMT_TK_AUTH = { TPMT_TK_AUTH-tag,
                 TPMT_TK-hierarchy,
                 TPMT_TK-digest,
               }

TPMT_TK_HASHCHECK = { TPMT_TK_HASHCHECK-tag,
                      TPMT_TK-hierarchy,
                      TPMT_TK-digest,
                    }

; Property Structures (10.8)

TPMS_ALG_PROPERTY = { TPMS_ALG_PROPERTY-alg,
                      TPMS_ALG_PROPERTY-algProperties,
                    }

TPMS_TAGGED_PROPERTY = { TPMS_TAGGED_PROPERTY-property,
                         TPMS_TAGGED_PROPERTY-value,
                       }

TPMS_TAGGED_PCR_SELECT = { TPMS_TAGGED_PCR_SELECT-tag,
                           TPMS_TAGGED_PCR_SELECT-pcrSelect,
                         }

TPMS_TAGGED_POLICY = { TPMS_TAGGED_POLICY-handle,
                       TPMS_TAGGED_POLICY-policyHash,
                     }

; Lists (10.9)

TPML_CC = { TPML-count,
            TPML_CC-commandCodes,
          }

TPML_CCA = { TPML-count,
             TPML_CCA-commandAttributes,
```

```
        }

TPML_ALG = { TPML-count,
          TPML_ALG-algorithms,
        }

TPML_HANDLE = { TPML-count,
            TPML_HANDLE-handle,
          }

TPML_DIGEST = { TPML-count, ; minimum here is 2
            TPML_DIGEST-digests,
          }

TPML_DIGEST_VALUES = { TPML-count,
                TPML_DIGEST_VALUES-digests,
              }

TPML_PCR_SELECTION = { TPML-count,
                TPML_PCR_SELECTION-pcrSelections,
              }

TPML_ALG_PROPERTY = { TPML-count,
               TPML_ALG_PROPERTY-algProperties,
             }

TPML_TAGGED_TPM_PROPERTY = { TPML-count,
                    TPML_TAGGED_TPM_PROPERTY-tpmProperty,
                  }

TPML_TAGGED_PCR_PROPERTY = { TPML-count,
                    TPML_TAGGED_PCR_PROPERTY-pcrProperty,
                  }

TPML_ECC_CURVE = { TPML-count,
              TPML_ECC_CURVE-eccCurves,
            }

TPML_TAGGED_POLICY = { TPML-count,
                TPML_TAGGED_POLICY-policies,
              }

; Capability Structures (10.10)

$TPMU_CAPABILITIES /= TPMU_CAPABILITIES-algorithms-entry
$TPMU_CAPABILITIES /= TPMU_CAPABILITIES-handles-entry
$TPMU_CAPABILITIES /= TPMU_CAPABILITIES-command-entry
$TPMU_CAPABILITIES /= TPMU_CAPABILITIES-ppCommands-entry
$TPMU_CAPABILITIES /= TPMU_CAPABILITIES-auditCommands-entry
$TPMU_CAPABILITIES /= TPMU_CAPABILITIES-assignedPCR-entry
$TPMU_CAPABILITIES /= TPMU_CAPABILITIES-tpmProperties-entry
$TPMU_CAPABILITIES /= TPMU_CAPABILITIES-pcrProperties-entry
$TPMU_CAPABILITIES /= TPMU_CAPABILITIES-eccCurves-entry
$TPMU_CAPABILITIES /= TPMU_CAPABILITIES-authPolicies-entry

TPMU_CAPABILITIES-algorithms-entry = { TPMU_CAPABILITIES-capability,
```

```
                    TPMU_CAPABILITIES-algorithms-data,
                  }

TPMU_CAPABILITIES-handles-entry = { TPMU_CAPABILITIES-capability,
                    TPMU_CAPABILITIES-handles-data,
                  }

TPMU_CAPABILITIES-command-entry = { TPMU_CAPABILITIES-capability,
                    TPMU_CAPABILITIES-command-data,
                  }

TPMU_CAPABILITIES-ppCommands-entry = { TPMU_CAPABILITIES-capability,
                    TPMU_CAPABILITIES-ppCommands-data,
                  }

TPMU_CAPABILITIES-auditCommands-entry = { TPMU_CAPABILITIES-capability,
                    TPMU_CAPABILITIES-auditCommands-data,
                  }

TPMU_CAPABILITIES-assignedPCR-entry = { TPMU_CAPABILITIES-capability,
                    TPMU_CAPABILITIES-assignedPCR-data,
                  }

TPMU_CAPABILITIES-tpmProperties-entry = { TPMU_CAPABILITIES-capability,
                    TPMU_CAPABILITIES-tpmProperties-data,
                  }

TPMU_CAPABILITIES-pcrProperties-entry = { TPMU_CAPABILITIES-capability,
                    TPMU_CAPABILITIES-pcrProperties-data,
                  }

TPMU_CAPABILITIES-eccCurves-entry = { TPMU_CAPABILITIES-capability,
                    TPMU_CAPABILITIES-eccCurves-data,
                  }

TPMU_CAPABILITIES-authPolicies-entry = { TPMU_CAPABILITIES-capability,
                    TPMU_CAPABILITIES-authPolicies,
                  }

TPMS_CAPABILITY_DATA = $TPMU_CAPABILITIES

; Clock/Counter Structures (10.11)

TPMS_CLOCK_INFO = { TPMS_CLOCK_INFO-clock,
                    TPMS_CLOCK_INFO-resetCount,
                    TPMS_CLOCK_INFO-restartCount,
                    TPMS_CLOCK_INFO-safe,
                  }

TPMS_TIME_INFO = { TPMS_TIME_INFO-time,
                   TPMS_TIME_INFO-clockInfo
                 }

; Attestation Strucutures (10.12)

TPMS_TIME_ATTEST_INFO = { TPMS_TIME_ATTEST_INFO-time,
```

```
                TPMS_TIME_ATTEST_INFO-firmwareVersion,
              }

TPMS_CERTIFY_INFO = { TPMS_CERTIFY_INFO-name,
              TPMS_CERTIFY_INFO-qualifiedName,
            }

TPMS_QUOTE_INFO = { TPMS_QUOTE_INFO-pcrSelect,
            TPMS_QUOTE_INFO-pcrDigest,
          }

TPMS_COMMAND_AUDIT_INFO = { TPMS_COMMAND_AUDIT_INFO-auditCounter,
              TPMS_COMMAND_AUDIT_INFO-digestAlg,
              TPMS_COMMAND_AUDIT_INFO-auditDigest,
              TPMS_COMMAND_AUDIT_INFO-commandDigest,
            }

TPMS_SESSION_AUDIT_INFO = { TPMS_SESSION_AUDIT_INFO-exclusiveSession,
              TPMS_SESSION_AUDIT_INFO-sessionDigest,
            }

TPMS_CREATION_INFO = { TPMS_CREATION_INFO-objectName,
              TPMS_CREATION_INFO-creationHash,
            }

TPMS_NV_CERTIFY_INFO = { TPMS_NV_CERTIFY_INFO-indexName,
              TPMS_NV_CERTIFY_INFO-offset,
              TPMS_NV_CERTIFY_INFO-nvContents,
            }

TPM2B_ATTEST = { TPM2B_ATTEST-size,
            TPM2B_ATTEST-attestationData,
          }

; Authorization Structures (10.13)

TPMS_AUTH_COMMAND = { TPMS_AUTH_COMMAND-sessionHandle,
              TPMS_AUTH_COMMAND-nonce,
              TPMS_AUTH_COMMAND-sessionAttributes,
              TPMS_AUTH_COMMAND-hmac,
            }

TPMS_AUTH_RESPONSE = { TPMS_AUTH_RESPONSE-nonce,
              TPMS_AUTH_RESPONSE-sessionAttributes,
              TPMS_AUTH_RESPONSE-hmac,
            }

; END OF TCG TPM STRUCTURES CDDL BODY
```


