SCMS MANAGER™
SECURITY CREDENTIAL MANAGEMENT SYSTEM

ELECTOR TECHNICAL SPECIFICATIONS

VERSION 1.1

January 20, 2020

Author(s)
SCMS Manager

© 2019-2020 SCMS Manager LLC - All Rights Reserved

TABLE OF CONTENTS

© 2019-2020 SCMS Manager LLC - All Rights Reserved

SCMS MANAGER ELECTOR TECHNICAL SPECIFICATIONS

1.0 ELECTOR TECHNICAL SPECIFICATIONS OVERVIEW

This document is a companion to the SCMS Manager Elector Policy Specification. While that document provides the high-level policies and procedures, this document provides detailed technical specifications for implementing those policies and procedures.

The design philosophy is to rely heavily on the CAMP specifications for the management of Electors. Where possible those specifications are used unchanged, where necessary they are updated, and, mostly, this document specifies additional details that were left open in the CAMP specifications.

2.0 ELECTOR DESIGN SELECTIONS

This section contains various design selections that were made in the SCMS Manager implementation of Electors.

2.1 NUMBER OF ELECTORS

The CAMP specification leaves open the design decision of the number of electors and does not detail any approach for dealing with changing the number of electors. The SCMS Manager shall fix the number electors at five (5). While flexibility is always desirable, because the Electors form the most basic root of trust, there is no authority that can be trusted to indicate a change in the number of Electors and any attempt to propagate such a change through the ecosystem could lead to confusion. So, for improved security, the SCMS manager is choosing to fix the number of electors.

2.2 TBSCTL MESSAGES

SCMS Manager shall use the Ballot concept as specified by CAMP, but updated per IEEE P1609.2.1/D9 as shown in Appendix A. Specifically, "'Ballot" is now replaced by "tbsCTL", the to-be-signed Certificate Trust List which contains the information to be signed by an Elector.

The term tbsCTL can be applied to this message with or without any signatures applied. For clarity, where necessary, the term Unsigned tbsCTL shall be used to indicate a tbsCTL that has not yet been signed by any Electors, the term Signed tbsCTL shall be used to indicate a tbsCTL that has been signed by an Elector, and the term MultiSignedCTL shall be used to indicate a tbsCTL where all necessary Elector signatures have been combined into a single CTL message ready to be sent to the Root CAs.

© 2019-2020 SCMS Manager LLC - All Rights Reserved

2.3 DIGITAL SIGNATURES

CAMP leaves open the possibility of Electors choosing different signature mechanisms and key lengths. But since End Entities (EE) are required to implement signature validation for any and all mechanisms selected, SCMS Manager shall require that all Electors use the Elliptic Curve Digital Signature Algorithm (ECDSA) as specified by NIST in FIPS 186-4 and further specified IEEE Std 1363-2000 using the NIST P-384 curve. This is the same digital signature algorithm specification that is used for IEEE P1609.2.1/D9 message signatures.

2.4 NUMBER OF SIGNATURES REQUIRED

Only a majority of Electors is required to make a MultiSignedCTL. Since the total number of Electors is five (5), any three (3) are necessary and sufficient to make a MultiSignedCTL. Therefore, when the SCMS Manager sends a tbsCTL out to the Electors, after the first three valid responses are received, the SCMS manager can create the MultiSignedCTL and send it to the Root CAs. While there is no need to wait for the other two responses, the SCMS Manager should still require the additional Electors to respond within the required timeframe and validate the responses for correctness. This is required to confirm that all five (5) Electors are still fully functional and available for future service.

2.5 QUANTUM-SAFE CRYPTOGRAPHY

Quantum-safe (QS) cryptography algorithms have not yet been standardized by NIST. So, the SCMS Manager has not selected a QS method for use at this time. When, and if, such an upgrade is necessary, the SCMS manager shall specify a QS method for the electors to use. During a transition period, Electors shall be required to sign all tbsCTL with both methods and the SCMS manager shall make available to Root CAs MultiSignedCTL signed using both methods. Once all system components are capable of validating the new QS signatures, the old method can be removed.

While the most likely need for an upgrade to the signature method will be the need for QS methods, it is possible other cryptographic developments may create the need for an update. The SCMS manager recommends that all system components keep in mind Crypto-agility and be prepared to receive updates that may be necessary – not only for digital signatures but for all cryptographic algorithms.

2.6 DISTRIBUTION OF SCMS MANAGER MESSAGES

IEEE P1609.2.1/D9 specifies that an addition to the list of Electors is done through the distribution of a new Global Certificate Chain File (GCCF) by the SCMS Manager to the Root CAs. It also specifies that the removal of an Elector is done through the CRL. The GCCF is signed by each SCMS' Policy Generator. This signature is validated upon receipt by an End Entity (EE). Upon validation of the signature of the Policy Generator, the EE will then validate the Elector signatures on the MultiSignedCTL for any new Roots or Electors in the GCCF. Once those signatures are validated the EE will then place the new Elector(s) or Root(s) in its local signature store. Once that has occurred the Policy Generator signed message is no longer required. When a properly signed message is received (through the CRL or other means) to remove an Elector or Root, the EE checks that the Elector signatures in the MutliSignedCTL are valid and then removes that Elector or Root from its local store.

2.7 EE PROTECTION OF ELECTOR PUBLIC KEYS

The root of trust for the entire Connected Vehicle (CV) system is the five (5) Elector public keys. Using those public keys Root CAs are installed. There is no requirement to keep them secret, in fact they should be published as widely as possible so that users/devices/CAs can validate that they are using the correct ones. To that end, SCMS Manager shall publish them, with a SHA-384 fingerprint, on its website www.scmsmanager.org,

It is critical that all EEs and users of these public keys prevent them from being substituted with different public keys. The process by which Electors and Root CAs are removed and installed must be carefully protected against misuse by an adversary. Secure processors are often equipped with tools for protecting secret and private keys against disclosure, and while protection against disclosure is not a problem for the Elector public keys, the must be afforded the highest protection against improper replacement. Please consult the SCMS Manager website for specific requirements for EE security.

3.0 INTERNATIONAL INTEROPERABILITY

As of this writing, the United States and Canada appear to be on a path to adopt compatible systems for Connected Vehicles (CVs). Assuming that Root CAs operate in the same way in both countries, this still leaves open the question of how Roots CAs will be authorized in each country. There are two primary possibilities:

1) Canada could adopt the US SCMS Manager. Then the SCMS Manager would authorize Roots for operation in Canada just as it does in the US. The same Elector public keys would be imbedded in all EEs in both the US and Canada. This would lead to complete interoperability between the US and Canada.

2) Canada may choose to set up its own SCMS Manager and its own electors. In that case, Canadian Electors would be embedded in Canadian vehicles and US Electors would be embedded in US vehicles. Without further action, CV messages exchanged between US and Canadian vehicles would not be recognized as properly signed. In this case, we recommend that agreements be made between the US and Canada to have the SCMS Manager in each country authorize the Roots from the other country. In this scenario, each country maintains its own sovereignty with respect to which Roots are permitted to operate within its own borders, while still maintaining full interoperability between the two countries.

If disputes arise between the US and Canada on which Roots should be authorized, it is possible that certificates issued by some CAs will not work upon crossing the border. We would expect such disputes to be rare and would be resolved by discussions between the two countries as to whether or not the CA in question maintains an acceptable security posture. CVs desiring to operate in both countries will need to use CAs that are not in dispute.

© 2019-2020 SCMS Manager LLC - All Rights Reserved

```
--
--  Licensed under the Apache License, Version 2.0 (the "License");
--  you may not use this file except in compliance with the License.
--  You may obtain a copy of the License at
--
--     http://www.apache.org/licenses/LICENSE-2.0
--
--  Unless required by applicable law or agreed to in writing, software
--  distributed under the License is distributed on an "AS IS" BASIS,
--  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
--  See the License for the specific language governing permissions and
--  limitations under the License.
--
------------------------------------------------------------------------------
--
-- The structures in this file define the protocol for messages for the
-- management of certificates of SCMS components, namely root CAs and
-- Electors.  It is compliant with IEEE P1609.2.1/D9.
--
------------------------------------------------------------------------------



--#
-- @brief This is the parent structure for all SCMS component certificate
--       management structures.
-- @class ScmsComponentCertificateManagementPDU
-- @param electorEndorsement contains the endorsement by an elector to add or
--                     remove an elector or a root CA.
-- @param electorBallot      contains the ballot consisting of the certificate
--                     of an elector or a root CA and the corresponding
--                     signed endorsements by one or more electors.
-- @param compositeCrl       contains zero or more SecuredCrl as defined in
--                     IEEE 1609.2, and zero or more ballot for removing
--                     electors or root CAs.
-- @param certificateChain   contains a collection of certificates maintained
--                     by an RA for the convenience of its clients.
-- @see CompositeCrl, ElectorBallot, ElectorEndorsement
 CertManagementPdu ::= CHOICE {
  compositeCrl                CompositeCrl,
  certificateChain            CertificateChain,
  multiSignedCertificateTrustList MultiSignedCertificateTrustList,
  tbsCtlSignature             ToBeSignedCtlSignature,
  ...
 }

MultiSignedCertificateTrustList ::= SEQUENCE {
  type       IEEE-1609-2-1-CTL.&type({Ieee1609dot2dot1Ctls}),
  tbsCtl     IEEE-1609-2-1-CTL.&TbsCtl({Ieee1609dot2dot1Ctls}{@.type}),
  unsigned   IEEE-1609-2-1-CTL.&UnsignedCtlMaterial({Ieee1609dot2dot1Ctls}{@.type}),
  signatures  SEQUENCE  (SIZE(1…MAX)) OF CtlSignatureSpdu
}

IEEE-1609-2-1-MSCTL ::= CLASS {
  &type       Ieee1609dot2dot1CtlType,
  &TbsCtl,
  &UnsignedCtlMaterial
 } WITH SYNTAX {&TbsCtl IDENTIFIED BY &type USING &UnsignedCtlMaterial}
 Ieee1609dot2dot1Ctls IEEE-1609-2-1-MSCTL ::= {
  { FullIeeeTbsCtl IDENTIFIED BY fullIeeeCtl USING SequenceOfCertificate},
  ...
 }

 Ieee1609dot2dot1MsctlType ::= INTEGER (0..255)
fullIeeeCtl            Ieee1609dot2dot1MsctlType ::= 1

 FullIeeeTbsCtl ::= SEQUENCE {
  type           Ieee1609dot2dot1MsctlType(fullIeeeCtl),
  electorGroupId  ElectorGroupId,
  sequenceNumber  CtlSequenceNumber,
  effectiveDate   Time32,
  electorApprove  SEQUENCE OF CtlElectorEntry,
  electorRemove   SEQUENCE OF CtlElectorEntry,
  rootCaApprove   SEQUENCE OF CtlRootCaEntry,
  rootCaRemove    SEQUENCE OF CtlRootCaEntry,
  ...
 }

ElectorGroupId ::=   OCTET STRING(SIZE(8))
CtlSequenceNumber ::= INTEGER(0..65535)
CtlElectorEntry ::= HashedId48
CtlRootCaEntry  ::= HashedId32

 ToBeSignedCtlSignature ::= SEQUENCE {
  electorGroupId ElectorGroupId,
  ctlType       Ieee1609dot2dot1MsctlType,
  sequenceNumber CtlSequenceNumber,
  tbsCtlHash     HashedId48
 }

 CtlSignatureSpdu ::= Ieee1609Dot2Data-Signed {
  ScmsPdu-Scoped {
   CertManagementPdu (WITH COMPONENTS {
    tbsCtlSignature
   })
  },
  SecurityMgmtPsid
 }

 MultiSignedCertificateTrustListSpdu ::= Ieee1609Dot2Data-Unsecured {
  ScmsPdu-Scoped {
   CertManagementPdu (WITH COMPONENTS {
    multiSignedCertificateTrustList
   })
  }
 }
```

APPENDIX B – ELECTOR CERTIFICATE FORMAT (ASN.1)

```
--
```

```
--  Licensed under the Apache License, Version 2.0 (the "License");
--  you may not use this file except in compliance with the License.
--  You may obtain a copy of the License at
--
--     http://www.apache.org/licenses/LICENSE-2.0
--
--  Unless required by applicable law or agreed to in writing, software
--  distributed under the License is distributed on an "AS IS" BASIS,
--  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
--  See the License for the specific language governing permissions and
--  limitations under the License.

--------------------------------------------------------------------------------
-- CERT-PROFILE
--
-- Defines the certificate structure for each component certificate in SCMS.
--
-- This file is part of the SCMS protocol developed by CAMP VSC5
-- It depends on the IEEE 1609.2 protocol specification
--------------------------------------------------------------------------------

-- @namespace IEEE1609dot2-electors
IEEE1609dot2-electors {iso(1) identified-organization(3) ieee(111)
standards-association-numbered-series-standards(2) wave-stds(1609)
dot2(2) base(1)}

  DEFINITIONS AUTOMATIC TAGS ::= BEGIN

EXPORTS ALL;

IMPORTS
  ExplicitCertificate,
  PsidSsp,
  SequenceOfPsidSsp
FROM IEEE1609dot2 {iso(1) identified-organization(3) ieee(111)
    standards-association-numbered-series-standards(2) wave-stds(1609)
    dot2(2) base(1) schema(1)}

  ElectorCertExpiration,
  SecurityMgmtPsid  -- PSID = 0x23
FROM Ieee1609dot2ScmsBaseTypes {iso(1) identified-organization(3) ieee(111)
                standards-association-numbered-series-standards(2) wave-stds(1609)  dot2(2)
                scms (2) interfaces(1) base-types (2)}

;

-- @brief This data type defines an Elector's certificate structure and its
--        contents generated by the SCMS Manager. An Elector does not provide
--        a public key (encryptionKey).
--        Elector certificates are not part of the PKI hierarchy of the SCMS, meaning
--        they do not sign certificates. They are used primarily for root CA
--        certificate management, including adding and removing a Root CA, by signing management
--        messages. Elector certificates are self signed.
-- @class ElectorCertificate
-- @param issuer     contains the hash of the certificate contents generated by
--                   the SCMS Manager.
-- @param toBeSigned contains certificate content that is self signed by the
--                   Elector.
-- @see IssuerIdentifier

ElectorCertificate ::= ExplicitCertificate (WITH COMPONENTS { ...,
  issuer (WITH COMPONENTS {self (sha384)}),
  toBeSigned (WITH COMPONENTS { ...,
    id (WITH COMPONENTS {
      name
    }),
    cracaId('000000'H),
    crlSeries(0),
    region ABSENT,
    assuranceLevel ABSENT,
    appPermissions (SequenceOfPsidSsp (SIZE(1)) (CONSTRAINED BY {
      PsidSsp (WITH COMPONENTS {
        psid (SecurityMgmtPsid), -- PSID = 0x23
        ssp --OER encoding of ScmsSsp indicating ElectorSsp
      })
    })),
    certIssuePermissions ABSENT,
    certRequestPermissions ABSENT,
    canRequestRollover ABSENT,
    encryptionKey ABSENT,
    verifyKeyIndicator (WITH COMPONENTS {
      verificationKey (WITH COMPONENTS {
          ecdsaNistP384 (WITH COMPONENTS {
            compressed-y-0, compressed-y-1
          }) |
          ecdsaBrainpoolP384r1(WITH COMPONENTS {
            compressed-y-0, compressed-y-1
          })
      })
    })
  })
 signature(WITH COMPONENTS{ecdsaNistP384Signature, ecdsaBrainpoolP384r1Signature})
})

--#
-- @brief This is the parent structure that encompasses all Service Specific
--        Permission (SSP) structures defined in IEEE 1609.2.1.
-- @class ScmsSsp
-- @param elector contains the SSP defined for an Elector.
-- @param root    contains the SSP defined for a Root CA.
-- @param pg      contains the SSP defined for a Policy Generator.
-- @param ica     contains the SSP defined for an Intermediate CA.
-- @param eca     contains the SSP defined for an Enrollment CA.
-- @param aca     contains the SSP defined for an Authorization CA.
-- @param crl     contains the SSP defined for a CRL Signer.
-- @param dcm     contains the SSP defined for a Device Configuration Manager.
-- @param la      contains the SSP defined for a Linkage Authority.
-- @param lop     contains the SSP defined for a Location Obscurer Proxy.
-- @param ma      contains the SSP defined for a Misbehavior Authority.
-- @param ra      contains the SSP defined for a Registration Authority.
  ScmsSsp ::= CHOICE {
    elector  ElectorSsp,
    root     RootCaSsp,
    pg       PgSsp,
    ica      IcaSsp,
    eca      EcaSsp,
    aca      AcaSsp,
    crl      CrlSignerSsp,
    dcm      DcmSsp,
    la       LaSsp,
    lop      LopSsp,
    ma       MaSsp,
    ra       RaSsp,
    ...
  }

--#
-- @brief This structure defines the SSP for an Elector.
-- @class ElectorSsp
-- @param version contains the current version of the data type.
-- @see Uint8
  ElectorSsp ::= SEQUENCE {
    version  Uint8 (2),
    ...
  }

END
```