付録１　　電磁的記録への記録方式（ASN.1 構造とオブジェクト識別子）

１　Explicitly Tagged Module

```
MOJCMPRegistration { 1 2 392 100300 1 4 41 }

DEFINITIONS EXPLICIT TAGS ::=
BEGIN

IMPORTS

    CertReqMessages, GeneralName, registeredCorporationInfo
    FROM MOJCRMFRegistration { 1 2 392 100300 1 4 42 };

PKIMessage ::= SEQUENCE {
      header          PKIHeader,
      body            PKIBody
}

PKIHeader ::= SEQUENCE {
      pvno              INTEGER     { ietf-version2 (1) },
      sender            GeneralName,
      recipient         GeneralName
}

PKIBody ::= CHOICE {
      ir      [0]  CertReqMessages  --Initialization Request
}

AlgorithmIdentifier     ::=  SEQUENCE {
   algorithm            ALGORITHM-ID.&id({SupportedAlgorithms}),
   parameters           ALGORITHM-ID.&Type({SupportedAlgorithms}
                                          { @algorithm }) OPTIONAL }

ALGORITHM-ID ::= CLASS {
     &id    OBJECT IDENTIFIER UNIQUE,
     &Type  OPTIONAL
}
WITH SYNTAX { OID &id [PARMS &Type] }

SupportedAlgorithms ALGORITHM-ID ::= { ..., -- extensible
                                       rsaPublicKey |
                                       rsaSHA-256 |
                                       sha256Identifier }

rsaPublicKey ALGORITHM-ID ::= { OID rsaEncryption PARMS NULL }
```

```
rsaSHA-256 ALGORITHM-ID ::= { OID sha256WithRSAEncryption PARMS NULL }

sha256Identifier ALGORITHM-ID ::= { OID id-SHA256 PARMS NULL }

pkcs-1 OBJECT IDENTIFIER ::= {
     iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) 1 }

rsaEncryption OBJECT IDENTIFIER ::=  { pkcs-1 1 }
-- subjectPublicKey syntax
RSAPublicKey  ::=  SEQUENCE  {
     modulus INTEGER,       -- n
     publicExponent INTEGER -- e
}

sha256WithRSAEncryption OBJECT IDENTIFIER  ::=  { pkcs-1 11 }

id-SHA256 OBJECT IDENTIFIER ::= {
     joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101)
     csor(3) nistalgorithm(4) hashalgs(2) 1 }

SubjectPublicKeyInfo ::= SEQUENCE{
   algorithm           AlgorithmIdentifier,
   subjectPublicKey    BIT STRING }

Extensions ::= SEQUENCE SIZE (1..MAX) OF Extension

Extension ::= SEQUENCE {
   extnId           EXTENSION.&id ({ExtensionSet}),
   critical         BOOLEAN DEFAULT FALSE,
   extnValue        OCTET STRING }

ExtensionSet  EXTENSION  ::= { registeredCorporationInfo }

EXTENSION ::= CLASS {
   &id         OBJECT IDENTIFIER UNIQUE,
   &ExtnType }
WITH SYNTAX  {
   SYNTAX              &ExtnType
   IDENTIFIED BY       &id }

AttributeTypeAndValue ::= SEQUENCE {
        type    ATTRIBUTE.&id ({SupportedAttributes}),
        value   ATTRIBUTE.&Type ({SupportedAttributes}{@type})}

Name ::= CHOICE {
        rdnSequence  RDNSequence
}

RDNSequence ::= SEQUENCE OF RelativeDistinguishedName

RelativeDistinguishedName ::= SET SIZE (1 .. MAX) OF AttributeTypeAndValue

ID ::= OBJECT IDENTIFIER

ATTRIBUTE ::= CLASS {
        &Type,
        &id     OBJECT IDENTIFIER UNIQUE }
WITH SYNTAX {
        WITH SYNTAX &Type
        ID          &id }

SupportedAttributes ATTRIBUTE ::= {
        commonName | organizationName |   -- RelativeDistinguishedName attributes
        timeLimit | suspensionSecretCode  -- regInfo attributes
}

commonName ATTRIBUTE ::= {
        WITH SYNTAX       DirectoryString {ub-common-name}
        ID                id-at-commonName }

organizationName ATTRIBUTE ::= {
        WITH SYNTAX       DirectoryString {ub-organization-name}
        ID                id-at-organizationName }

DirectoryString { INTEGER:maxSize } ::= CHOICE {
        printableString        PrintableString (SIZE (1..maxSize)),
        utf8String             UTF8String (SIZE(1..maxSize))
}

timeLimit ATTRIBUTE ::= {
        WITH SYNTAX       TimeLimit
        ID                id-registeredcert-mg-effectiveTimeLimit }

TimeLimit ::= OCTET STRING

suspensionSecretCode ATTRIBUTE ::= {
        WITH SYNTAX       SuspensionSecretCode
        ID                id-registeredcert-mg-suspensionSecretCode }

SuspensionSecretCode ::= SEQUENCE {
      hashAlg                    AlgorithmIdentifier,
      hashedSecretCode            OCTET STRING
}

id-at OBJECT IDENTIFIER ::= {joint-iso-ccitt(2) ds(5) 4}

id-at-commonName              OBJECT IDENTIFIER ::= {id-at 3}
id-at-organizationName        OBJECT IDENTIFIER ::= {id-at 10}

id-registeredcert OBJECT IDENTIFIER ::= { 1 2 392 100300 1 }

id-registeredcert-mg OBJECT IDENTIFIER ::= { id-registeredcert 2 }

id-registeredcert-mg-effectiveTimeLimit OBJECT IDENTIFIER ::= { id-registeredcert-mg 104 }
id-registeredcert-mg-suspensionSecretCode OBJECT IDENTIFIER ::= {
    id-registeredcert-mg 105 }

ub-common-name               INTEGER ::= 64
ub-organization-name         INTEGER ::= 64

END
```

２　Implicitly Tagged Module

```
MOJCRMFRegistration { 1 2 392 100300 1 4 42 }

DEFINITIONS IMPLICIT TAGS ::=
BEGIN

IMPORTS
        AttributeTypeAndValue, AlgorithmIdentifier, Name,
        SubjectPublicKeyInfo, Extensions, DirectoryString, EXTENSION
        FROM MOJCMPRegistration { 1 2 392 100300 1 4 41 };


CertReqMessages ::= SEQUENCE SIZE (1..MAX) OF CertReqMsg

CertReqMsg ::= SEQUENCE {
    certReq   CertRequest,
    pop       ProofOfPossession  OPTIONAL,
    regInfo   SEQUENCE SIZE(1..MAX) OF AttributeTypeAndValue }

CertRequest ::= SEQUENCE {
    certReqId     INTEGER,
    certTemplate  CertTemplate }

CertTemplate ::= SEQUENCE {
    subject      [5] Name                  OPTIONAL,
    publicKey    [6] SubjectPublicKeyInfo,
    extensions   [9] Extensions
}

ProofOfPossession ::= CHOICE {
    signature         [1] POPOSigningKey }

POPOSigningKey ::= SEQUENCE {
    algorithmIdentifier   AlgorithmIdentifier,
    signature             BIT STRING }

GeneralName ::= CHOICE {
        directoryName               [4] Name
}

registeredCorporationInfo EXTENSION ::= {
        SYNTAX  RegisteredCorporationInfoSyntax
        IDENTIFIED BY id-registeredcert-pe-registeredCorporationInfo }

RegisteredCorporationInfoSyntax ::= SEQUENCE {
  corporateName               [0] EXPLICIT DirectoryString{ub-corporate-name},
  corporateAddress            [2] EXPLICIT DirectoryString{ub-corporate-address},
  representativeDirectorName  [3] EXPLICIT DirectoryString
                                           {ub-representative-director-name},
  representativeDirectorTitle [4] EXPLICIT DirectoryString
                                           {ub-representative-director-title}
}

id-registeredcert OBJECT IDENTIFIER ::= { 1 2 392 100300 1 }

id-registeredcert-pe OBJECT IDENTIFIER ::=  { id-registeredcert 1 }

id-registeredcert-pe-registeredCorporationInfo OBJECT IDENTIFIER ::= {
     id-registeredcert-pe 3 }

ub-corporate-name                 INTEGER ::= 128
ub-corporate-address              INTEGER ::= 128
ub-representative-director-name   INTEGER ::= 126
ub-representative-director-title  INTEGER ::= 128

END
```

付録２　電子証明書の方式（ASN.１構造とオブジェクト識別子）

１　Explicitly Tagged Module

```
MOJCorpCertExplicit { 1 2 392 100300 1 4 1 }
DEFINITIONS EXPLICIT TAGS ::=
BEGIN

-- EXPORTS ALL --

IMPORTS
        authorityKeyIdentifier, subjectKeyIdentifier, keyUsage,
        privateKeyUsagePeriod, certificatePolicies,
        basicConstraints, authorityInfoAccess, jCertificatePolicies,
        registrar, registeredCorporationInfo, crlDistributionPoints
        FROM MOJCorpCertImplicit { 1 2 392 100300 1 4 2 };


Certificate  ::=  SIGNED { TBSCertificate }

TBSCertificate  ::=  SEQUENCE {
   version                [0]   Version,
   serialNumber                 CertificateSerialNumber,
   signature                    AlgorithmIdentifier,
   issuer                       Name,
   validity                     Validity,
   subject                      Name,
   subjectPublicKeyInfo         SubjectPublicKeyInfo,
   extensions             [3]   Extensions  }

Version  ::=  INTEGER { v3(2) }

CertificateSerialNumber ::= INTEGER

Validity ::= SEQUENCE {
   notBefore          Time,
   notAfter           Time }

Time ::= CHOICE {
        utcTime        UTCTime
}

SubjectPublicKeyInfo ::= SEQUENCE{
   algorithm          AlgorithmIdentifier,
   subjectPublicKey   BIT STRING }

Extensions ::= SEQUENCE SIZE (1..MAX) OF Extension
```

```
Extension ::= SEQUENCE {
   extnId           EXTENSION.&id ({ExtensionSet}),
   critical         BOOLEAN DEFAULT FALSE,
   extnValue        OCTET STRING }

ExtensionSet  EXTENSION  ::= { authorityKeyIdentifier |
                               subjectKeyIdentifier |
                               keyUsage |
                               privateKeyUsagePeriod |
                               certificatePolicies |
                               basicConstraints |
                               authorityInfoAccess |
                               jCertificatePolicies |
                               registrar |
                               registeredCorporationInfo |
                               crlDistributionPoints }

EXTENSION ::= CLASS {
   &id         OBJECT IDENTIFIER UNIQUE,
   &ExtnType }
WITH SYNTAX  {
   SYNTAX              &ExtnType
   IDENTIFIED BY       &id }

SIGNED { ToBeSigned } ::= SEQUENCE {
     toBeSigned  ToBeSigned,
     algorithm   AlgorithmIdentifier,
     signature   BIT STRING
}

AlgorithmIdentifier  ::=  SEQUENCE {
   algorithm           ALGORITHM-ID.&id({SupportedAlgorithms}),
   parameters          ALGORITHM-ID.&Type({SupportedAlgorithms}
                                          { @algorithm}) OPTIONAL }

ALGORITHM-ID ::= CLASS {
     &id    OBJECT IDENTIFIER UNIQUE,
     &Type  OPTIONAL
}
WITH SYNTAX { OID &id [PARMS &Type] }

SupportedAlgorithms ALGORITHM-ID ::= { ..., -- extensible
                                       rsaPublicKey |
                                       rsaSHA-256 }

rsaPublicKey ALGORITHM-ID ::= { OID rsaEncryption PARMS NULL }
```

```
rsaSHA-256 ALGORITHM-ID ::= { OID sha256WithRSAEncryption PARMS NULL }

pkcs-1 OBJECT IDENTIFIER ::= {
     iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) 1 }

rsaEncryption OBJECT IDENTIFIER ::=  { pkcs-1 1 }

-- subjectPublicKey syntax
RSAPublicKey  ::=  SEQUENCE  {
     modulus INTEGER, -- n
     publicExponent INTEGER -- e
}

sha256WithRSAEncryption OBJECT IDENTIFIER  ::=  { pkcs-1 11 }

AttributeTypeAndValue ::= SEQUENCE {
        type    ATTRIBUTE.&id ({SupportedAttributes}),
        value   ATTRIBUTE.&Type ({SupportedAttributes}{@type})}

Name ::= CHOICE {
        rdnSequence  RDNSequence
}

RDNSequence ::= SEQUENCE OF RelativeDistinguishedName

RelativeDistinguishedName ::= SET SIZE (1 .. MAX) OF AttributeTypeAndValue

ID ::= OBJECT IDENTIFIER

ATTRIBUTE ::= CLASS {
        &Type,
        &id     OBJECT IDENTIFIER UNIQUE }
WITH SYNTAX {
        WITH SYNTAX &Type
        ID          &id }

SupportedAttributes ATTRIBUTE ::= {
        commonName | countryName | organizationName | organizationalUnitName }

commonName ATTRIBUTE ::= {
        WITH SYNTAX        DirectoryString {ub-common-name}
        ID                 id-at-commonName }

countryName ATTRIBUTE ::= {
        WITH SYNTAX        PrintableString (SIZE (2))
        ID                  id-at-countryName }

organizationName ATTRIBUTE ::= {
        WITH SYNTAX        DirectoryString {ub-organization-name}
        ID                  id-at-organizationName }

organizationalUnitName ATTRIBUTE ::= {
        WITH SYNTAX        DirectoryString {ub-organizational-unit-name}
        ID                  id-at-organizationalUnitName }

id-at OBJECT IDENTIFIER ::= {joint-iso-ccitt(2) ds(5) 4}

id-at-commonName              OBJECT IDENTIFIER ::= {id-at 3}
id-at-countryName             OBJECT IDENTIFIER ::= {id-at 6}
id-at-organizationName        OBJECT IDENTIFIER ::= {id-at 10}
id-at-organizationalUnitName  OBJECT IDENTIFIER ::= {id-at 11}

DirectoryString { INTEGER:maxSize } ::= CHOICE {
        printableString         PrintableString (SIZE (1..maxSize)),
        utf8String              UTF8String (SIZE(1..maxSize))
}

ub-common-name                INTEGER ::= 64
ub-organization-name          INTEGER ::= 64
ub-organizational-unit-name   INTEGER ::= 64

END
```

２　Implicitly Tagged Module

```
MOJCorpCertImplicit { 1 2 392 100300 1 4 2 }

DEFINITIONS IMPLICIT TAGS ::=
BEGIN

--EXPORTS ALL --

IMPORTS
        Name, CertificateSerialNumber, DirectoryString, EXTENSION
        FROM MOJCorpCertExplicit { 1 2 392 100300 1 4 1 };


authorityKeyIdentifier EXTENSION ::= {
        SYNTAX          AuthorityKeyIdentifier
        IDENTIFIED BY   id-ce-authorityKeyIdentifier }
```

```
AuthorityKeyIdentifier ::= SEQUENCE {
    keyIdentifier              [0] KeyIdentifier,
    authorityCertIssuer        [1] GeneralNames,
    authorityCertSerialNumber  [2] CertificateSerialNumber }

KeyIdentifier ::= OCTET STRING

GeneralNames ::= SEQUENCE SIZE (1..MAX) OF GeneralName

GeneralName ::= CHOICE {
        directoryName               [4] Name,
        uniformResourceIdentifier   [6] IA5String
}

subjectKeyIdentifier EXTENSION ::= {
        SYNTAX          SubjectKeyIdentifier
        IDENTIFIED BY   id-ce-subjectKeyIdentifier }

SubjectKeyIdentifier ::= KeyIdentifier

keyUsage EXTENSION ::= {
        SYNTAX KeyUsage
        IDENTIFIED BY id-ce-keyUsage }

KeyUsage ::= BIT STRING {
        digitalSignature     (0),
        nonRepudiation       (1),
        keyEncipherment      (2),
        dataEncipherment     (3),
        keyAgreement         (4),
        keyCertSign          (5),
        cRLSign              (6)
        encipherOnly         (7),
        decipherOnly         (8)
}

privateKeyUsagePeriod EXTENSION ::= {
        SYNTAX  PrivateKeyUsagePeriod
        IDENTIFIED BY { id-ce-privateKeyUsagePeriod } }

PrivateKeyUsagePeriod ::= SEQUENCE {
        notBefore       [0]     GeneralizedTime,
        notAfter        [1]     GeneralizedTime }

certificatePolicies EXTENSION ::= {
        SYNTAX  CertificatePoliciesSyntax
        IDENTIFIED BY id-ce-certificatePolicies }

CertificatePoliciesSyntax ::=
                SEQUENCE SIZE (1..MAX) OF PolicyInformation

PolicyInformation ::= SEQUENCE {
        policyIdentifier   CertPolicyId,
        policyQualifiers   SEQUENCE SIZE (1..MAX) OF PolicyQualifierInfo }

CertPolicyId ::= OBJECT IDENTIFIER

PolicyQualifierInfo ::= SEQUENCE {
        policyQualifierId       CERT-POLICY-QUALIFIER.&id
                                    ({SupportedPolicyQualifiers}),
        qualifier               CERT-POLICY-QUALIFIER.&Qualifier
                                    ({SupportedPolicyQualifiers}
                                    {@policyQualifierId}) }

SupportedPolicyQualifiers CERT-POLICY-QUALIFIER ::= { noticeToUser }

CERT-POLICY-QUALIFIER ::= CLASS {
        &id             OBJECT IDENTIFIER UNIQUE,
        &Qualifier  }
WITH SYNTAX {
        POLICY-QUALIFIER-ID     &id
        QUALIFIER-TYPE          &Qualifier }

noticeToUser CERT-POLICY-QUALIFIER ::= {
     POLICY-QUALIFIER-ID id-qt-unotice  QUALIFIER-TYPE   UserNotice }

UserNotice ::= SEQUENCE {
     noticeRef        NoticeReference,
     explicitText     DisplayText }

NoticeReference ::= SEQUENCE {
     organization     DisplayText,
     noticeNumbers    SEQUENCE OF INTEGER }

DisplayText ::= CHOICE {
     visibleString    VisibleString  (SIZE (1..200)),
     utf8String       UTF8String     (SIZE (1..200)) }

basicConstraints EXTENSION ::= {
        SYNTAX  BasicConstraintsSyntax
        IDENTIFIED BY id-ce-basicConstraints }
```

```
BasicConstraintsSyntax ::= SEQUENCE {
        cA           BOOLEAN DEFAULT FALSE
}

authorityInfoAccess EXTENSION ::= {
        SYNTAX  AuthorityInfoAccessSyntax
        IDENTIFIED BY id-pe-authorityInfoAccess }

AuthorityInfoAccessSyntax  ::=
        SEQUENCE SIZE (1..MAX) OF AccessDescription

AccessDescription  ::=  SEQUENCE {
        accessMethod         OBJECT IDENTIFIER,
        accessLocation       GeneralName  }

jCertificatePolicies EXTENSION ::= {
        SYNTAX  JCertificatePoliciesSyntax
        IDENTIFIED BY id-registeredcert-pe-jCertificatePolicies }

JCertificatePoliciesSyntax ::= SEQUENCE SIZE (1..MAX) OF PolicyInformation

registrar EXTENSION ::= {
        SYNTAX  RegistrarSyntax
        IDENTIFIED BY id-registeredcert-pe-registrar }

RegistrarSyntax ::= DirectoryString{ub-registrar}

registeredCorporationInfo EXTENSION ::= {
        SYNTAX  RegisteredCorporationInfoSyntax
        IDENTIFIED BY id-registeredcert-pe-registeredCorporationInfo }

RegisteredCorporationInfoSyntax ::= SEQUENCE {
  corporateName               [0] EXPLICIT DirectoryString{ub-corporate-name},
  registeredNumber            [1] EXPLICIT PrintableString,
  corporateAddress            [2] EXPLICIT DirectoryString{ub-corporate-address},
  representativeDirectorName  [3] EXPLICIT DirectoryString
                                           {ub-representative-director-name},
  representativeDirectorTitle [4] EXPLICIT DirectoryString
                                           {ub-representative-director-title},
  registryOffice              [6] EXPLICIT DirectoryString{ub-registry-office}
}

crlDistributionPoints EXTENSION ::= {
        SYNTAX  CRLDistPointsSyntax
        IDENTIFIED BY id-ce-cRLDistributionPoints }
```

```
CRLDistPointsSyntax ::= SEQUENCE SIZE (1..MAX) OF DistributionPoint

DistributionPoint ::= SEQUENCE {
        distributionPoint       [0]     DistributionPointName }

DistributionPointName ::= CHOICE {
        fullName                [0]     GeneralNames }

id-ce   OBJECT IDENTIFIER ::= {joint-iso-ccitt(2) ds(5) 29}

id-ce-subjectKeyIdentifier         OBJECT IDENTIFIER ::= {id-ce 14}
id-ce-keyUsage                     OBJECT IDENTIFIER ::= {id-ce 15}
id-ce-privateKeyUsagePeriod        OBJECT IDENTIFIER ::= {id-ce 16}
id-ce-basicConstraints             OBJECT IDENTIFIER ::= {id-ce 19}
id-ce-cRLDistributionPoints        OBJECT IDENTIFIER ::= {id-ce 31}
id-ce-certificatePolicies          OBJECT IDENTIFIER ::= {id-ce 32}
id-ce-authorityKeyIdentifier       OBJECT IDENTIFIER ::= {id-ce 35}

id-pkix OBJECT IDENTIFIER ::=
  { iso(1) identified-organization(3) dod(6) internet(1) security(5) mechanisms(5) pkix(7) }

id-pe   OBJECT IDENTIFIER ::= { id-pkix 1 }

id-pe-authorityInfoAccess OBJECT IDENTIFIER ::= { id-pe 1 }

id-qt   OBJECT IDENTIFIER ::= { id-pkix 2 }

id-qt-unotice  OBJECT IDENTIFIER ::=  { id-qt 2 }

id-ad   OBJECT IDENTIFIER ::= { id-pkix 48 }

id-ad-ocsp      OBJECT IDENTIFIER ::= { id-ad 1 }

id-registeredcert OBJECT IDENTIFIER ::= { 1 2 392 100300 1 }

id-registeredcert-pe OBJECT IDENTIFIER ::=  { id-registeredcert 1 }

id-registeredcert-pe-jCertificatePolicies OBJECT IDENTIFIER ::= {id-registeredcert-pe 1 }
id-registeredcert-pe-registrar OBJECT IDENTIFIER ::= { id-registeredcert-pe 2 }
id-registeredcert-pe-registeredCorporationInfo OBJECT IDENTIFIER ::= {
     id-registeredcert-pe 3 }

ub-registrar                      INTEGER ::= 128
ub-corporate-name                 INTEGER ::= 128
ub-corporate-address              INTEGER ::= 128
ub-representative-director-name   INTEGER ::= 126
```

```
ub-representative-director-title  INTEGER ::= 128
ub-registry-office                INTEGER ::= 128

END
```

付録３　電子証明書の送信の方式（送受信電文の ASN.1 構造とオブジェクト識別子）

１　Explicitly Tagged Module

```
MOJCMPCertReq { 1 2 392 100300 1 4 11 }

DEFINITIONS EXPLICIT TAGS ::=
BEGIN

IMPORTS

      Certificate
             FROM MOJCorpCertExplicit { 1 2 392 100300 1 4 1 }

      GeneralName, CertReqMessages, EncryptedValue
             FROM MOJCRMFCertReq { 1 2 392 100300 1 4 12 };

AlgorithmIdentifier ::= SEQUENCE {
   algorithm           ALGORITHM-ID.&id ({SupportedAlgorithms}),
   parameters          ALGORITHM-ID.&Type ({SupportedAlgorithms}
                                         { @algorithm }) OPTIONAL }

ALGORITHM-ID ::= CLASS {
     &id    OBJECT IDENTIFIER UNIQUE,
     &Type  OPTIONAL
}
WITH SYNTAX { OID &id [PARMS &Type] }

SupportedAlgorithms ALGORITHM-ID ::= { ..., -- extensible
                                       rsaPublicKey |
                                       rsaSHA-256 |
                                       des-EDE3-CBC |
                                       des-EDE3-CBC-NoParms |
                                       sha256Identifier }

rsaPublicKey ALGORITHM-ID ::= { OID rsaEncryption PARMS NULL }

rsaSHA-256 ALGORITHM-ID ::= { OID sha256WithRSAEncryption PARMS NULL }

des-EDE3-CBC ALGORITHM-ID ::= { OID dES-EDE3-CBC PARMS CBCParameter }
```

```
des-EDE3-CBC-NoParms ALGORITHM-ID ::= { OID dES-EDE3-CBC PARMS NULL }

sha256Identifier ALGORITHM-ID ::= { OID id-SHA256 PARMS NULL }


pkcs-1 OBJECT IDENTIFIER ::= {
     iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) 1 }

rsaEncryption OBJECT IDENTIFIER ::=  { pkcs-1 1 }
-- Public key syntax
RSAPublicKey  ::=  SEQUENCE  {
     modulus INTEGER, -- n
     publicExponent INTEGER -- e
}

sha256WithRSAEncryption OBJECT IDENTIFIER  ::=  { pkcs-1 11 }

dES-EDE3-CBC OBJECT IDENTIFIER  ::=  {
     iso(1) member-body(2) us(840) rsadsi(113549) encryptionAlgorithm(3) 7 }
-- dES-EDE3-CBC parameters
CBCParameter ::= IV
IV ::= OCTET STRING (SIZE (8..8))

id-SHA256 OBJECT IDENTIFIER ::= {
     joint-iso-itu-t(2) country(16) us(840) organization(1)
     gov(101) csor(3) nistalgorithm(4) hashalgs(2) 1 }


InfoTypeAndValue ::= SEQUENCE {
      infoType       INFORMATION-ID.&id({InfoSet}),
      infoValue      INFORMATION-ID.&Type({InfoSet}{@infoType})
}

INFORMATION-ID ::= CLASS {
     &id    OBJECT IDENTIFIER UNIQUE,
     &Type
}
WITH SYNTAX { SYNTAX &Type IDENTIFIED BY &id }

InfoSet INFORMATION-ID ::= { genmInfoReqContent |
                             genpInfoResContent
}

genmInfoReqContent INFORMATION-ID ::= {
  SYNTAX         GenmInfoReqContent
  IDENTIFIED BY  id-registeredcert-mg-genminforeq
}

genpInfoResContent INFORMATION-ID ::= {
  SYNTAX         GenmInfoReqContent
  IDENTIFIED BY  id-registeredcert-mg-genpinfores
}

PKIMessage ::= SEQUENCE {
      header           PKIHeader,
      body             PKIBody,
      protection   [0] PKIProtection OPTIONAL,
      extraCerts   [1] SEQUENCE SIZE (1..MAX) OF Certificate OPTIONAL
}

PKIHeader ::= SEQUENCE {
      pvno                INTEGER     { ietf-version2 (1) },
      sender              GeneralName,
      recipient           GeneralName,
      protectionAlg   [1] AlgorithmIdentifier     OPTIONAL,
      senderKID       [2] KeyIdentifier           OPTIONAL,
      transactionID   [4] OCTET STRING,
      senderNonce     [5] OCTET STRING,
      recipNonce      [6] OCTET STRING            OPTIONAL
}

PKIBody ::= CHOICE {       -- message-specific body elements
      ir      [0]  CertReqMessages,        --Initialization Request
      ip      [1]  CertRepMessage,         --Initialization Response
      genm    [21] GenMsgContent,          --General Message
      genp    [22] GenRepContent,          --General Response
      error   [23] ErrorMsgContent         --Error Message
}

PKIProtection ::= BIT STRING

ProtectedPart ::= SEQUENCE {
      header    PKIHeader,
      body      PKIBody
}

KeyIdentifier ::= OCTET STRING

PKIStatus ::= INTEGER {
      granted                (0),
      -- you got exactly what you asked for
      rejection              (2)
      -- you don't get it, more information elsewhere in the message
}

PKIStatusInfo ::= SEQUENCE {
      status        PKIStatus
}

CertRepMessage ::= SEQUENCE {
      response         SEQUENCE OF CertResponse
}

CertResponse ::= SEQUENCE {
      certReqId           INTEGER,
      status              PKIStatusInfo,
      certifiedKeyPair    CertifiedKeyPair
}

CertifiedKeyPair ::= SEQUENCE {
      certOrEncCert       CertOrEncCert
}

CertOrEncCert ::= CHOICE {
      encryptedCert   [1] EncryptedValue
}

GenMsgContent ::= SEQUENCE OF InfoTypeAndValue

GenRepContent ::= SEQUENCE OF InfoTypeAndValue

ErrorMsgContent ::= SEQUENCE {
      pKIStatusInfo          PKIStatusInfo
}

id-registeredcert OBJECT IDENTIFIER ::= { 1 2 392 100300 1 }

id-registeredcert-mg OBJECT IDENTIFIER ::= { id-registeredcert 2 }

id-registeredcert-mg-genminforeq OBJECT IDENTIFIER ::= { id-registeredcert-mg 21 }
id-registeredcert-mg-genpinfores OBJECT IDENTIFIER ::= { id-registeredcert-mg 22 }

GenmInfoReqContent ::= SEQUENCE OF NegotiationKey

NegotiationKey ::= SEQUENCE {
     symmAlg  AlgorithmIdentifier,
     pubAlg   AlgorithmIdentifier,
     hashAlg  AlgorithmIdentifier
}

GenpInfoResContent ::= SEQUENCE {
     status           PKIStatusInfo,
     negotiationKeys  SEQUENCE OF NegotiationKey OPTIONAL
}

END
```

２　Implicitly Tagged Module

```
MOJCRMFCertReq { 1 2 392 100300 1 4 12 }

DEFINITIONS IMPLICIT TAGS ::=
BEGIN

IMPORTS

        AlgorithmIdentifier
        FROM MOJCMPCertReq { 1 2 392 100300 1 4 11 }

        SubjectPublicKeyInfo, Name, RDNSequence
        FROM MOJCorpCertExplicit { 1 2 392 100300 1 4 1 };

CertReqMessages ::= SEQUENCE SIZE (1..MAX) OF CertReqMsg

CertReqMsg ::= SEQUENCE {
    certReq   CertRequest,
    pop       ProofOfPossession }

CertRequest ::= SEQUENCE {
    certReqId     INTEGER,
    certTemplate  CertTemplate }

CertTemplate ::= SEQUENCE {
    serialNumber  [1] INTEGER,
    publicKey     [6] SubjectPublicKeyInfo }

ProofOfPossession ::= CHOICE {
    signature            [1] POPOSigningKey }

POPOSigningKey ::= SEQUENCE {
    algorithmIdentifier   AlgorithmIdentifier,
    signature             BIT STRING }
```

```
EncryptedValue ::= SEQUENCE {
    symmAlg       [1] AlgorithmIdentifier,
    encSymmKey    [2] BIT STRING,
    keyAlg        [3] AlgorithmIdentifier,
    encValue       BIT STRING }

GeneralName ::= CHOICE {
    directoryName         [4] Name }

END
```

付録４　休止届の送信の方式（送受信電文の ASN.1 構造とオブジェクト識別子）

１　Explicitly Tagged Module

```
MOJCMPSuspReq { 1 2 392 100300 1 4 21 }

DEFINITIONS EXPLICIT TAGS ::=
BEGIN

IMPORTS

      Certificate
      FROM MOJCorpCertExplicit { 1 2 392 100300 1 4 1 }

      GeneralName, EncryptedValue, CertTemplate, CertId, ReasonFlags
      FROM MOJCRMFSuspReq { 1 2 392 100300 1 4 22 };


AlgorithmIdentifier     ::=  SEQUENCE {
   algorithm            ALGORITHM-ID.&id({SupportedAlgorithms}),
   parameters           ALGORITHM-ID.&Type({SupportedAlgorithms}
                                   { @algorithm }) OPTIONAL }

ALGORITHM-ID ::= CLASS {
     &id    OBJECT IDENTIFIER UNIQUE,
     &Type  OPTIONAL
}
WITH SYNTAX { OID &id [PARMS &Type] }

SupportedAlgorithms  ALGORITHM-ID  ::=  { ..., -- extensible
                                          rsaPublicKey |
                                          rsaSHA-256 |
                                          des-EDE3-CBC |
                                          sha256Identifier }
```

```
rsaPublicKey ALGORITHM-ID ::= { OID rsaEncryption PARMS NULL }

rsaSHA-256 ALGORITHM-ID ::= { OID sha256WithRSAEncryption PARMS NULL }

des-EDE3-CBC ALGORITHM-ID ::= { OID dES-EDE3-CBC PARMS CBCParameter }

sha256Identifier ALGORITHM-ID ::= { OID id-SHA256 PARMS NULL }

pkcs-1 OBJECT IDENTIFIER ::= {
     iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) 1 }

rsaEncryption OBJECT IDENTIFIER ::=  { pkcs-1 1 }
-- subjectPublicKey syntax
RSAPublicKey  ::=  SEQUENCE  {
     modulus INTEGER, -- n
     publicExponent INTEGER -- e
}

sha256WithRSAEncryption OBJECT IDENTIFIER  ::=  { pkcs-1 11 }

dES-EDE3-CBC OBJECT IDENTIFIER  ::=  {
     iso(1) member-body(2) us(840) rsadsi(113549) encryptionAlgorithm(3) 7 }
-- dES-EDE3-CBC parameters
CBCParameter ::= IV
     IV ::= OCTET STRING (SIZE (8..8))

id-SHA256 OBJECT IDENTIFIER ::= {
     joint-iso-itu-t(2) country(16) us(840) organization(1)
     gov(101) csor(3) nistalgorithm(4) hashalgs(2) 1 }

InfoTypeAndValue ::= SEQUENCE {
      infoType       INFORMATION-ID.&id({InfoSet}),
      infoValue      INFORMATION-ID.&Type({InfoSet}{@infoType})
}

INFORMATION-ID ::= CLASS {
     &id    OBJECT IDENTIFIER UNIQUE,
     &Type
}
WITH SYNTAX { SYNTAX &Type IDENTIFIED BY &id }

InfoSet INFORMATION-ID ::= { genmSuspReqContent |
                             genpSuspResContent |
                             genmInfoReqContent |
                             genpInfoResContent
}

genmSuspReqContent INFORMATION-ID ::= {
  SYNTAX        GenmSuspReqContent
  IDENTIFIED BY id-registeredcert-mg-genmsuspreq
}

genpSuspResContent INFORMATION-ID ::= {
  SYNTAX        GenpSuspResContent
  IDENTIFIED BY id-registeredcert-mg-genpsuspres
}

genmInfoReqContent INFORMATION-ID ::= {
  SYNTAX        GenmInfoReqContent
  IDENTIFIED BY id-registeredcert-mg-genminforeq
}

genpInfoResContent INFORMATION-ID ::= {
  SYNTAX        GenpInfoResContent
  IDENTIFIED BY id-registeredcert-mg-genpinfores
}

PKIMessage ::= SEQUENCE {
      header           PKIHeader,
      body             PKIBody,
      protection   [0] PKIProtection OPTIONAL,
      extraCerts   [1] SEQUENCE SIZE (1..MAX) OF Certificate OPTIONAL
}

PKIHeader ::= SEQUENCE {
      pvno                 INTEGER     { ietf-version2 (1) },
      sender               GeneralName,
      recipient            GeneralName,
      protectionAlg    [1] AlgorithmIdentifier     OPTIONAL,
      senderKID        [2] KeyIdentifier           OPTIONAL,
      transactionID    [4] OCTET STRING,
      senderNonce      [5] OCTET STRING,
      recipNonce       [6] OCTET STRING            OPTIONAL
}

PKIBody ::= CHOICE {       -- message-specific body elements
      genm    [21] GenMsgContent,          --General Message
      genp    [22] GenRepContent,          --General Response
      error   [23] ErrorMsgContent         --Error Message
}
```

```
PKIProtection ::= BIT STRING

ProtectedPart ::= SEQUENCE {
      header    PKIHeader,
      body      PKIBody
}

PKIStatus ::= INTEGER {
      granted                (0),
      -- you got exactly what you asked for
      rejection              (2)
      -- you don't get it, more information elsewhere in the message
}

KeyIdentifier ::= OCTET STRING

PKIStatusInfo ::= SEQUENCE {
      status        PKIStatus
}

GenMsgContent ::= SEQUENCE OF InfoTypeAndValue

GenRepContent ::= SEQUENCE OF InfoTypeAndValue

ErrorMsgContent ::= SEQUENCE {
      pKIStatusInfo          PKIStatusInfo
}

id-registeredcert OBJECT IDENTIFIER ::= { 1 2 392 100300 1 }

id-registeredcert-mg OBJECT IDENTIFIER ::= { id-registeredcert 2 }

id-registeredcert-mg-genmsuspreq OBJECT IDENTIFIER ::= { id-registeredcert-mg 1 }
id-registeredcert-mg-genpsuspres OBJECT IDENTIFIER ::= { id-registeredcert-mg 2 }
id-registeredcert-mg-genminforeq OBJECT IDENTIFIER ::= { id-registeredcert-mg 21 }
id-registeredcert-mg-genpinfores OBJECT IDENTIFIER ::= { id-registeredcert-mg 22 }

GenmSuspReqContent ::= SEQUENCE {
      certDetails           CertTemplate,
      revocationReason      ReasonFlags,
      suspensionReasonCode  INTEGER,
      suspensionDetail      EncryptedValue  -- encrypted SuspData
}
-- SuspData is made with conection of
-- "suspensionSecretCode" (which is password without tag and length) and
-- hashed "header PKIHeader".
-- The max size of Suspdata is 84Bytes. (Max64Bytes + 20Bytes).

GenpSuspResContent ::= SEQUENCE {
      status         PKIStatusInfo,
      -- status information of suspension results
      revCert        CertId
      -- IDs for which revocation was requested (same order as status)
}

GenmInfoReqContent ::= SEQUENCE OF NegotiationKey

NegotiationKey ::= SEQUENCE {
     symmAlg  AlgorithmIdentifier,
     pubAlg   AlgorithmIdentifier,
     hashAlg  AlgorithmIdentifier  }

GenpInfoResContent ::= SEQUENCE {
     status           PKIStatusInfo,
     negotiationKeys  SEQUENCE OF NegotiationKey OPTIONAL  }

END
```

２　Implicitly Tagged Module

```
MOJCRMFSuspReq { 1 2 392 100300 1 4 22 }

DEFINITIONS IMPLICIT TAGS ::=
BEGIN

IMPORTS
        AlgorithmIdentifier
        FROM MOJCMPSuspReq { 1 2 392 100300 1 4 21 }

        SubjectPublicKeyInfo, Name, RDNSequence
        FROM MOJCorpCertExplicit { 1 2 392 100300 1 4 1 };


CertTemplate ::= SEQUENCE {
    serialNumber [1] INTEGER,
    issuer       [3] Name }

EncryptedValue ::= SEQUENCE {
    keyAlg        [3] AlgorithmIdentifier,
    encValue       BIT STRING }

CertId ::= SEQUENCE {
    issuer           GeneralName,
    serialNumber     INTEGER }

GeneralName ::= CHOICE {
        directoryName               [4] Name  }

ReasonFlags ::= BIT STRING {
        certificateHold         (6) }

END
```

付録５　電子証明書に係る証明及びその請求の方式（送受信電文の ASN.1 構造とオブジェクト識別子）

```
MOJOCSP { 1 2 392 100300 1 4 31 }

DEFINITIONS EXPLICIT TAGS ::=
BEGIN

IMPORTS

          Certificate
          FROM MOJCorpCertExplicit { 1 2 392 100300 1 4 1 };


OCSPRequest ::=  SEQUENCE {
    tbsRequest                  TBSRequest  }

TBSRequest ::= SEQUENCE {
    version             [0] EXPLICIT Version DEFAULT v1,
    requestList             SEQUENCE OF Request,
    requestExtensions   [2] EXPLICIT Extensions  -- nonce
}

Version ::= INTEGER  { v1(0) }

Request ::= SEQUENCE {
    reqCert                     CertID,
    singleRequestExtensions    [0] EXPLICIT Extensions OPTIONAL  -- confirmationTime
}

CertID ::= SEQUENCE {
    hashAlgorithm      AlgorithmIdentifier,
    issuerNameHash     OCTET STRING, -- Hash of Issuer's DN
    issuerKeyHash      OCTET STRING, -- Hash of Issuers public key
    serialNumber       CertificateSerialNumber }

CertificateSerialNumber ::= INTEGER

OCSPResponse ::= SEQUENCE {
   responseStatus         OCSPResponseStatus,
   responseBytes          [0] EXPLICIT ResponseBytes OPTIONAL }

OCSPResponseStatus ::= ENUMERATED {
    successful            (0),      --Response has valid confirmations
    malformedRequest      (1)       --Illegal confirmation request
}

ResponseBytes ::=       SEQUENCE {
    responseType   RESPONSE.&Type({SupportedResponses}),
    response       RESPONSE.&Value({SupportedResponses}{ @responceType })
}

RESPONSE ::= CLASS {
     &Type    OBJECT IDENTIFIER UNIQUE,
     &Value
}
WITH SYNTAX { SYNTAX &Value IDENTIFIED BY &Type }

SupportedResponses RESPONSE ::= { ..., -- extensible
                                  basicOCSPResponse }

basicOCSPResponse RESPONSE ::= {
  SYNTAX          BasicOCSPResponse
  IDENTIFIED BY   id-pkix-ocsp-basic
}

BasicOCSPResponse ::= SEQUENCE {
    tbsResponseData      ResponseData,
    signatureAlgorithm   AlgorithmIdentifier,
    signature            BIT STRING,
    certs                [0] EXPLICIT SEQUENCE OF Certificate }

ResponseData ::= SEQUENCE {
    version              [0] EXPLICIT Version DEFAULT v1,
    responderID              ResponderID,
    producedAt               GeneralizedTime,
    responses                SEQUENCE OF SingleResponse,
    responseExtensions   [1] EXPLICIT Extensions   -- nonce
}
```

```
ResponderID ::= CHOICE {
    byKey    [2] KeyHash }

KeyHash ::= OCTET STRING -- SHA-1 hash of responder's public key
                        -- (excluding the tag and length fields)

SingleResponse ::= SEQUENCE {
    certID                      CertID,
    certStatus                  CertStatus,
    thisUpdate                  GeneralizedTime,
    singleExtensions     [1]    EXPLICIT Extensions
                                -- confirmationTime and ocspStatusCode
}

CertStatus ::= CHOICE {
    good               [0]     IMPLICIT NULL,
    revoked            [1]     IMPLICIT RevokedInfo,
    unknown            [2]     IMPLICIT UnknownInfo }

RevokedInfo ::= SEQUENCE {
    revocationTime             GeneralizedTime,
    revocationReason   [0]     EXPLICIT CRLReason }

UnknownInfo ::= NULL

AlgorithmIdentifier     ::=  SEQUENCE {
   algorithm           ALGORITHM-ID.&id({SupportedAlgorithms}),
   parameters          ALGORITHM-ID.&Type({SupportedAlgorithms}
                                   { @algorithm }) }

ALGORITHM-ID ::= CLASS {
     &id    OBJECT IDENTIFIER UNIQUE,
     &Type
}
WITH SYNTAX { OID &id PARMS &Type }

SupportedAlgorithms ALGORITHM-ID ::= { ..., -- extensible
                                       rsaSHA-256 }

rsaSHA-256 ALGORITHM-ID ::= { OID sha256WithRSAEncryption PARMS NULL }

pkcs-1 OBJECT IDENTIFIER ::= {
     iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) 1 }

sha256WithRSAEncryption OBJECT IDENTIFIER  ::=  { pkcs-1 11 }
```

```
CRLReason ::= ENUMERATED {
        keyCompromise           (1),
        cACompromise            (2),
        affiliationChanged      (3),
        cessationOfOperation    (5),
        certificateHold         (6)
}

Extensions ::= SEQUENCE SIZE (1..MAX) OF Extension

Extension ::= SEQUENCE {
   extnId           EXTENSION.&id ({ExtensionSet}),
   critical         BOOLEAN DEFAULT FALSE,
   extnValue        OCTET STRING }

ExtensionSet  EXTENSION  ::= { ..., -- extensible
                               nonce |
                               confirmationTime |
                               ocspStatusCode
}

EXTENSION ::= CLASS {
   &id         OBJECT IDENTIFIER UNIQUE,
   &ExtnType
}
WITH SYNTAX  {
   SYNTAX         &ExtnType
   IDENTIFIED BY  &id }

nonce EXTENSION ::= {
        SYNTAX  Nonce
        IDENTIFIED BY id-pkix-ocsp-nonce }

Nonce ::= OCTET STRING

confirmationTime EXTENSION ::= {
        SYNTAX  ConfirmationTime
        IDENTIFIED BY id-registeredcert-mg-confirmationtime }

ConfirmationTime ::= GeneralizedTime

ocspStatusCode EXTENSION ::= {
        SYNTAX  OcspStatusCode
        IDENTIFIED BY id-registeredcert-mg-ocspstatuscode }

OcspStatusCode ::= INTEGER
```

```
id-pkix OBJECT IDENTIFIER ::=
  { iso(1) identified-organization(3) dod(6) internet(1) security(5) mechanisms(5) pkix(7) }

id-ad   OBJECT IDENTIFIER ::= { id-pkix 48 }

id-ad-ocsp      OBJECT IDENTIFIER ::= { id-ad 1 }

id-pkix-ocsp                OBJECT IDENTIFIER ::= { id-ad-ocsp }
id-pkix-ocsp-basic          OBJECT IDENTIFIER ::= { id-pkix-ocsp 1 }
id-pkix-ocsp-nonce          OBJECT IDENTIFIER ::= { id-pkix-ocsp 2 }
-- The nonce will be identified by the object identifier id-pkix-ocsp-nonce,
-- while the extnValue is the value of the nonce.

id-registeredcert OBJECT IDENTIFIER ::= { 1 2 392 100300 1 }

id-registeredcert-mg OBJECT IDENTIFIER ::= { id-registeredcert 2 }

id-registeredcert-mg-confirmationtime OBJECT IDENTIFIER ::= { id-registeredcert-mg 102 }
id-registeredcert-mg-ocspstatuscode OBJECT IDENTIFIER ::= { id-registeredcert-mg 103 }

END
```

付録６　失効リストの公開（ASN.1構造とオブジェクト識別子）

１　Explicitly Tagged Module

```
MOJCorpCertCRLExplicit { 1 2 392 100300 1 4 51 }

DEFINITIONS EXPLICIT TAGS ::=
BEGIN

-- EXPORTS ALL --

IMPORTS
        AlgorithmIdentifier, Name, Time, CertificateSerialNumber
        FROM MOJCorpCertExplicit { 1 2 392 100300 1 4 1 }
        authorityKeyIdentifier
        FROM MOJCorpCertImplicit { 1 2 392 100300 1 4 2 };
        cRLNumber, issuingDistributionPoint, reasonCode
        FROM MOJCorpCertCRLImplicit { 1 2 392 100300 1 4 52 };

CertificateList ::=  SIGNED { TBSCertList }

TBSCertList ::= SEQUENCE {
   version                     Version,
   signature                   AlgorithmIdentifier,
   issuer                      Name,
   thisUpdate                  Time,
   nextUpdate                  Time,
   revokedCertificates         SEQUENCE OF SEQUENCE {
      userCertificate             CertificateSerialNumber,
      revocationDate              Time,
      crlEntryExtensions          EntryExtensions },
   crlExtensions         [0]   CRLExtensions }

Version ::= INTEGER { v2(1) }

EntryExtensions ::= SEQUENCE SIZE (1..MAX) OF EntryExtension

EntryExtension ::= SEQUENCE {
   extnId            EXTENSION.&id ({EntryExtensionSet}),
   critical          BOOLEAN DEFAULT FALSE,
   extnValue         OCTET STRING }

EntryExtensionSet  EXTENSION  ::= { reasonCode }

CRLExtensions ::= SEQUENCE SIZE (1..MAX) OF CRLExtension

CRLExtension ::= SEQUENCE {
   extnId          EXTENSION.&id ({CRLExtensionSet}),
   critical        BOOLEAN DEFAULT FALSE,
   extnValue       OCTET STRING }

CRLExtensionSet  EXTENSION  ::= { authorityKeyIdentifier |
                                  cRLNumber |
                                  issuingDistributionPoint }

EXTENSION ::= CLASS {
   &id         OBJECT IDENTIFIER UNIQUE,
   &ExtnType }
WITH SYNTAX  {
   SYNTAX               &ExtnType
   IDENTIFIED BY        &id }

SIGNED { ToBeSigned } ::= SEQUENCE {
     toBeSigned  ToBeSigned,
     algorithm   AlgorithmIdentifier,
     signature   BIT STRING
}
```

```
END
```

２　Implicitly Tagged Module

```
MOJCorpCertCRLImplicit { 1 2 392 100300 1 4 52 }

DEFINITIONS IMPLICIT TAGS ::=
BEGIN

--EXPORTS ALL --

IMPORTS
        DistributionPointName
        FROM MOJCorpCertImplicit { 1 2 392 100300 1 4 2 }
        CRLReason
        FROM MOJOCSP { 1 2 392 100300 1 4 31 };
        EXTENSION
        FROM MOJCorpCertCRLExplicit { 1 2 392 100300 1 4 51 }

reasonCode EXTENSION ::= {
        SYNTAX          CRLReason
        IDENTIFIED BY   id-ce-cRLReason }

cRLNumber EXTENSION ::= {
        SYNTAX          CRLNumber
        IDENTIFIED BY   id-ce-cRLNumber }

CRLNumber ::= INTEGER (0..MAX)

issuingDistributionPoint EXTENSION ::= {
        SYNTAX          IssuingDistributionPoint
        IDENTIFIED BY   id-ce-issuingDistributionPoint }

IssuingDistributionPoint ::= SEQUENCE {
        distributionPoint          [0] DistributionPointName,
        onlyContainsUserCerts      [1] BOOLEAN DEFAULT FALSE,
        onlyContainsCACerts        [2] BOOLEAN DEFAULT FALSE }

id-ce   OBJECT IDENTIFIER ::= {joint-iso-ccitt(2) ds(5) 29}

id-ce-cRLNumber                OBJECT IDENTIFIER ::= {id-ce 20}
id-ce-cRLReason                OBJECT IDENTIFIER ::= {id-ce 21}
id-ce-issuingDistributionPoint OBJECT IDENTIFIER ::= {id-ce 28}

END
```