Internet Draft Denis Pinkas, Integris
draft-ietf-pkix-dpv-dpd-00.txt

July, 2001

Expires in six months

Delegated Path Validation and Delegated Path Discovery Protocols <draft-ietf-pkix-dpv-dpd-00.txt>

Status of this memo

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC 2026.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

1. Abstract

This document specifies two protocols. The first one, called Delegated Path Validation (DPV) can be used to fully delegate a path validation processing to an DPV server, according to a set of rules called a validation policy.

The second one, called Delegated Path Discovery (DPD) can be used to obtain from a DPD server all the information needed (e.g. leaf certificates, CA certificates, full CRLs, delta-CRLs, OCSP responses) to locally validate a certificate according to a set of rules called a path validation criteria. This provides a single protocol to collect at one time all data elements that normally require different protocols.

It also defines an optional protocol allowing to define the set of rules to validate a certificate or to build a path for a certificate.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT",

"SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document (in uppercase, as shown) are to be interpreted as described in [RFC2119].

2. Delegated Path Validation Protocol Overview

The Delegated Path Validation (DPV) protocol allows to validate one certificate for a time T and according to a set of rules, called a validation policy. The validation policies may be known a priori by the DPV server or may be defined by the DPV client. When the validation policy is not a priori known by the DPV server, the client may define it using an additional protocol. The support of that additional protocol is optional.

In this way, the validation protocol always refers to a validation policy defined both by an OID and a hash in order to make sure that the definition of the validation policy is as intended.

The time T may be close to the present time or a time in the past.

When it is a time close to the present time, the server MUST do its best efforts to perform the validation (validation may not be possible if the required data may not be collected). This is called an initial validation.

The support of validation in the past, using some data previously captured at the time of initial verification is optional.

When supported, the server MUST be able to use the data that is provided by the requester which may be the validation data that has been previously returned when making an initial validation. This is called a re-validation.

In order to obtain the revocation status information of any certificate from the certification path, the DPV server MAY use, as appropriate and accordingly to the validation policy, any combination of OCSP requests, CRLs or delta-CRLs.

3. Validation Policy

A validation policy is a set of rules against which the validation of the certificate is performed. In order to succeed, one valid path (i.e. none of the certificates from the path must be revoked) must be found between a leaf certificate and a trust anchor.

A trust anchor is defined as a public key for a given CA name and valid during some time interval, a set of Certification Policy constraints and a set of naming constraints. The use of a self-signed certificate allows to specify at the same time: the public key to be used, the CA name and the validity period of the root key. Additional constrains MAY be included in the self-signed certificate.

Additional conditions that apply to the certificates from the chain,

in particular to the leaf certificate MAY also be specified in the validation policy.

An optional protocol allows to define the validation policy. See section 8.

4. Initial validation and re-validation

The same validation protocol may be used either for:

1) initial validation or for,
2) re-validation.

4.1 Initial validation

The initial validation is performed according to the validation policy. Additional conditions MAY be locally added, e.g. in order to test the presence of some extensions in the certificate chain.

If the client does not specify in its request the validation policy to be used, the server will indicate in the response the one that has been used. In such a case, the client MUST verify that the one selected is appropriate for its use.

The status of the response may be one out of four types:

1) the certificate is valid according to the validation policy,

2) the certificate is not valid according to the validation policy,

3) the certificate is conditionally valid according to the validation policy. A path has been able to be constructed, however one or more revocation status information are missing. If another request could made later on, the certificate could be determined as valid.

4) the server cannot determine the validity (e.g. a path cannot be constructed).

In most instances, in order to be able to prove to a third party that the check has correctly be done, the client will only require a signed response. In other cases, the client will need to get all the data that has been collected during the validation so that the test can be redone again using the same information, in a subsequent re-validation. In such a case the server will need to return that information, called validation data.

The validation data can be rather long since it consists of a certification path and its associated revocation status information for each element from the path.

If the validation data was directly part of the signed response, then the response could be rather long if needed to be stored. For that

reason, only the unambiguous references to the certification path and the revocation status information are optionally returned, and only the hash of it is part of the signed response.

In addition the actual values of the validation data (certificates, CRLs, delta-CRLs or OCSP responses) may also be returned and only the hash of it is part of the signed response.

4.2 Re-validation

Re-validation is performed according to a validation policy.

The client MUST specify in its request the validation policy to be used.

The requestor MUST specify the certificate to be validated and MUST provide previous returned references of the certification path, but may need to provide as well previous returned values of the certification path.

5. Description the DPV protocol

A DPV request may be optionally signed and contains the following

data:

-- protocol version
-- optionally, the identification of the requestor
-- identification of the certificate to validate
-- optionally, the validation policy to be used
-- whether or not the references of the path should be returned
-- whether or not the values of the path should be returned
-- whether or not the references of the path should be time-stamped
-- optionally, useful certificates that can be used by the server
-- optionally, useful revocation information that can be used by the server
-- optionally, previous returned references for re-validation
-- optionally, previous returned values for re-validation
-- a nonce to allow replay protection, when the client has no clock
-- optional extensions

Upon receipt of a request, a DPV Responder determines if:

1. the message is well formed;

2. the responder is configured to provide the requested service; and

3. the request contains the information needed by the responder.

If any one of the prior conditions are not met, the DPV responder produces an error message; otherwise, it returns a definitive response.

A DPV response contains a major status, followed by a signed response

in case of "success" and optionally followed with path references and path values.

The signed response includes the following data:

```
-- protocol version,
-- an optional identification of the requestor,
-- the validation status,
-- identification of the certificate that has been validated,
-- the reference of the validation policy that has been used
-- the validation time,
-- the response time,
-- an optional hash of the path references, that may include
   a sequence of time-stamps,
-- an optional hash of the path values,
-- an optional nonce to detect replays,
-- a field for future extensions.
```

5.2. Detailed Protocol

The ASN.1 syntax imports terms defined in [RFC2459] and in [ES-F]. For signature calculation, the data to be signed is encoded using the ASN.1 distinguished encoding rules (DER) [X.690].

ASN.1 EXPLICIT tagging is used as a default unless specified otherwise.

The terms imported from elsewhere are: Extensions, CertificateSerialNumber, SubjectPublicKeyInfo, Name, AlgorithmIdentifier, CRLReason, CompleteCertificateRefs, CompleteRevocationRefs.

5.2.1. Request

This section specifies the ASN.1 specification for a request. The actual formatting of the message could vary depending on the transport mechanism used (HTTP, SMTP, LDAP, etc.).

```
DPVRequest     ::=     SEQUENCE {
    tbsRequest                TBSRequest,
    optionalSignature    [0]  EXPLICIT Signature       OPTIONAL }

TBSRequest      ::=     SEQUENCE {
    version              [0]  EXPLICIT Version           DEFAULT v1,
    requestorName        [1]  EXPLICIT GeneralName       OPTIONAL,
    certToValidate            CertOrCertRef,
    valPolicyRef              ValPolicyRef               OPTIONAL,
    validationTime            GeneralizedTime            OPTIONAL,
    returnRefs           [2]  BOOLEAN Default            FALSE,
    returnValues         [3]  BOOLEAN Default            FALSE,
    timeStampRefs        [4]  BOOLEAN Default            FALSE,
    usefulCerts               UsefulCerts                OPTIONAL,
```

```
       usefulRevoc               UsefulRevoc               OPTIONAL,
       previousReturnedRefs      PathReferences            OPTIONAL,
       nonce                     INTEGER                   OPTIONAL,
       requestExtensions    [5]  EXPLICIT Extensions       OPTIONAL }
```

```
   Signature       ::=     SEQUENCE {
       signatureAlgorithm        AlgorithmIdentifier,
       signature                 BIT STRING,
       certs                [0]  EXPLICIT SEQUENCE OF Certificate
                                 OPTIONAL }

   Version ::=                   INTEGER  {  v1(0) }

   CertOrCertRef ::=  CHOICE {
       certificate          [1]  Certificate,
       certRef              [2]  OtherCertID }

   ValPolicyRef ::=   SEQUENCE{
     valPolicyID                 ValPolicyID,
     valPolicyHashAlg            AlgorithmIdentifier,
     valPolicyHash               ValPolicyHash,
     valPolLocation              ValPolLocations     OPTIONAL }

   ValPolicyID :: = CHOICE {
       policybyOId               OBJECT IDENTIFIER,
       policybyURN               NAME }

ValPolicyHash ::= OCTET STRING
```

The value for valPolicyHash SHALL be computed on the hash of the DER encoding of ValidationPolicyDef when the policy is locally defined or of the definition of the policy when it is externally defined.

```
   ValPolLocations :: = SEQUENCE OF Name

   UsefulCerts ::= CHOICE {
       certificateSet                  CertificateSet,
       completeCertificateRefs         CompleteCertificateRefs }

   UsefulRevoc ::= CHOICE {
       certificateRevocationLists      CertificateRevocationLists,
       completeRevocationRefs          CompleteRevocationRefs }

   PathReferences :: = SEQUENCE {
       completeCertificateRefs         CompleteCertificateRefs,
       completeRevocationRefs          CompleteRevocationRefs }

   PathValues :: = SEQUENCE {
       certificateValues               CertificateValues,
       revocationValues                RevocationValues }
```

version allows to identify the version of the protocol.

requestorName is an optional field that allows to identify the

requestor. It is used when the request is signed. In that case the name of the requestor must match one of the names found in the certificate.

certToValidate is the certificate to validate. It is either the actual value of the certificate or an unambiguous reference of that certificate: the issuer name and the serial number of the certificate, and a hash value of the certificate in order to guard against compromission of CA keys. Structures like ESSCertID can be used as a reference.

validationPolicyRef is a reference to the validation policy to be used. If the field is missing, the server will indicate which pre-defined policy has been used. It is composed of an OID or a URN, the hash algorithm to be used to compute the hash value of the policy and the hash value of the policy.

validationTime is the time for which the validation should be performed. If that field is missing, then the current time is assumed.

returnRefs is an indicator to ask the server to return the references of the path (both the references of the certificates used and the references of the revocation information used).

returnValues is an indicator to ask the server to return the values of the path (both the values of the certificates used and the values of the revocation information used).

timeStampRefs is an indicator to ask the server to return a time-stamped version of the returnRefs. The number of time-stamps and the names of the TSAs are indicated in the validation Policy. This provides a protection in case a key from a CA, CRL Issuer or OCSP responder would be compromised after the date of issue of the time-stamps.

usefulCerts is a set of certificates, usually obtained through the application protocol that may be useful for the server to build a path.

usefulRevoc is a set of revocation information, usually obtained through the application protocol that may be useful for the server to make sure that no element from the path is revoked.

previousReturnedRefs is used to avoid to return previous returned paths.

nonce is a unique number (e.g. a large pseudo random number) generated by the client that may be used to detect replay.

requestExtensions is a way to allow additional elements to be added later on, if needed.

5.2.2. Response Syntax

This section specifies the ASN.1 specification for a confirmation

response. The actual formatting of the message could vary depending on the transport mechanism used (HTTP, SMTP, LDAP, etc.).

An DPV response at a minimum consists of a dPVStatus field indicating the processing status of the prior request. If the value of dPVStatus is one of the error conditions, tbsResponse and optionalSignature are not set.

```
DPVResponse     ::=     SEQUENCE {
    dPVStatus                   DPVResponseStatus,
    tbsResponse                 TBSResponse           OPTIONAL,
    optionalSignature   [0]     EXPLICIT Signature    OPTIONAL,
    pathReferences              CertPathRefs          OPTIONAL,
    pathValues                  CertPathValues        OPTIONAL }

CertPathRefs :: = SEQUENCE {
    pathReferences            SEQUENCE OF PathReferences,
    timeStamping              SEQUENCE OF TimeStampToken OPTIONAL }

CertPathValues ::= SEQUENCE OF PathValues

DPVResponseStatus ::= ENUMERATED {
    successful            (0),  -- Request was understood
    malformedRequest      (1),  -- malformed request
    internalError         (2),  -- internal error in issuer
    tryLater              (3),  -- try again later
                                -- (4) is not used
    sigRequired           (5),  -- must sign the request
    unauthorized          (6)   -- request unauthorized }

TBSResponse       ::= SEQUENCE {
   tbsResponseData      DPVResponseData,
   signatureAlgorithm   AlgorithmIdentifier,
   signature            BIT STRING,
   certs                [0] EXPLICIT SEQUENCE OF Certificate
                            OPTIONAL }
```

The value for signature SHALL be computed on the hash of the DER encoding of DPVResponseData.

```
DPVResponseData      ::=     SEQUENCE {
    version             [0]     EXPLICIT Version      DEFAULT v1,
    requestorName       [1]     EXPLICIT GeneralName  OPTIONAL,
    validationStatus            ValidationStatus
    certProcessed               CertOrCertRef,
    valPolicyRef                ValPolicyRef,
    validationTime              GeneralizedTime,
    producedAt                  GeneralizedTime,
    pathReferencesHash          OCTET STRING          OPTIONAL,
    pathValuesHash              OCTET STRING          OPTIONAL,
    nonce                       INTEGER               OPTIONAL,
```

```
       responseExtensions  [2]     EXPLICIT Extensions   OPTIONAL }
```

The value for returnedRefsHash SHALL be computed on the hash of the DER encoding of CertPathRefs.

The value for returnedValuesHash SHALL be computed on the hash of the DER encoding of CertPathValues.

```
   ValidationStatus ::= CHOICE {
       valid               [0]     IMPLICIT NULL,
       invalid             [1]     IMPLICIT RevokedInfo,
       conditionallyValid  [2]     IMPLICIT TryLater,
       unknown             [3]     IMPLICIT UnknownInfo }

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

   TryLater ::= GeneralizedTime

   UnknownInfo ::= NULL -- this can be replaced with an enumeration
```

The response MUST be integrity protected. Thus the signature MUST be used if the request was successful.

The various parameters from the DPVResponseData are the following:

version allows to identify the version of the protocol.

requestorName is a copy of the field present in the request, if that field was present.

validationStatus indicates the validity of the certificate according to the validation policy.

When the response indicates "valid", this means that the certificate is valid according to the validation policy.

When the response indicates "invalid", this means that the certificate is invalid according to the validation policy.

When the response indicates "conditionallyValid", this means that either some revocation information is currently missing or that the one element of the certification path is currently "on hold". In that case the server indicates when another attempt could be done.

When the response indicates "unknown", this means that information is missing to build a path between the leaf certificate and a trusted anchor. This may also be, when the reference to the certificate is used, because the server is unable to get the actual value of the leaf certificate.

certProcessed is the certificate to validate. It is a copy of the
field present in the request.

validationPolicy is the validation policy that has been used. It is a copy of the field present in the request, if that field was present. If the field was not present, it is the object identifier that has been used to perform the validation.

validationTime is the time for which the validation should be performed. If that field was present in the request it is a copy of the field present in the request. If that field was not present then the current time is being assumed and that time is identical to the next field: producedAt.

producedAt is the time at which the response has been formed.

pathReferencesHash is a hash computed over the references of the path (both the references of the certificates used and the references of the revocation information used). It may also include a sequence of time-stamps, if this has been requested in the request. Since only the hash is included in the signature, this allows to keep signatures short and does not mandate to know the values of the references of the path to verify the dPVResponseStatus from the response.

pathValuesHash is a hash computed over the values of the path (both the values of the certificates used and the values of the revocation information used). Since only the hash is included in the signature, this allows to keep signatures short and does not mandate to know the values of the values of the path to verify the dPVResponseStatus from the response.

nonce is a pseudo random number generated by the client that may be used to detect replay.

requestExtensions is a way to allow additional elements to be added later on, if needed.

6. Delegated Path Discovery Protocol Overview

The Delegated Path Discovery (DPD) protocol allows to obtain information that can be used to locally validate one certificate for the current time and according to a path validation criteria. The path validation criteria may either be a sequence of self-signed certificates or a reference to a validation policy.

None, one or several certification paths may be returned. Each path consists of a sequence of certificates, starting with the certificate to validate and ending with one self-signed certificate. In addition, the revocation information associated with each path may also be returned.

The client may indicate the maximum number of certification paths that

MUST be returned, provided that they may be found. If the number is not specified, that number is defaulted to one.

The paths that are returned may need to match some additional local controls done by the client, e.g. verifying some certificate extensions.

If the paths that are returned do not mach the local conditions, then the number of number of certification paths to be returned can be extended, by augmenting this value.

The server may use a local cache to avoid to search again the same elements, but is not mandated to maintain any local state information from any previous request.

Path discovery is performed according to the path validation criteria.

The status of the response may be one out of three types:

1) one or more certification paths could be found according to the path validation criteria, with partial or full revocation information present.

2) one or more certification paths could be found according to the path validation criteria, with no revocation information present.

3) no certification path could be found according to the path validation criteria,

The information that is returned consists of one or more certification paths and its associated revocation status information for each element from the path.

6.1. Description the DPD protocol

A DPD request may be optionally signed and contains the following data:

-- protocol version
-- identification of the certificate for the path discovery
-- the path discovery criteria to be used
-- optionally, the number of certification paths to be returned
-- whether or not the revocation data should be returned
-- optionally, useful certificates that can be used by the server
-- optionally, useful revocation information that can be used by the server
-- a nonce to allow replay protection, when the client has no clock
-- optional extensions

Upon receipt of a request, a DPD Responder determines if:

1. the message is well formed;

2. the responder is configured to provide the requested service; and
3. the request contains the information needed by the responder.

If any one of the prior conditions are not met, the DPD responder produces an error message; otherwise, it returns a definitive response.

A DPD response contains a major status, followed by a response in case of "success" and with path values. The response includes the following data:

```
-- protocol version,
-- the discovery status,
-- identification of the certificate under path discovery,
-- the path discovery criteria that has been used
-- the response time,
-- the path values,
-- an optional limit for the number of certification paths
   that have been returned
-- an optional nonce to detect replays,
-- a field for future extensions.
```

The response SHOULD be protected by a data origin authentication service combined with a data integrity service. A TLS protection or an IPSEC protection may be appropriate.

6.2. Detailed Protocol

The ASN.1 syntax imports terms defined in [RFC2459] and in [ES-F].

ASN.1 EXPLICIT tagging is used as a default unless specified otherwise.

The terms imported from elsewhere are: Extensions, CertificateSerialNumber, SubjectPublicKeyInfo, Name, AlgorithmIdentifier, CRLReason, CompleteCertificateRefs, CompleteRevocationRefs.

6.2.1. Request

This section specifies the ASN.1 specification for a request. The actual formatting of the message could vary depending on the transport mechanism used (HTTP, SMTP, LDAP, etc.).

```
DPDRequest     ::=     SEQUENCE {
    version              [0]  EXPLICIT Version           DEFAULT v1,
    certUnderDiscovery        CertOrCertRef,
    pathDiscoveryCriteria     PathDiscoveryCriteria,
    pathsNumber               INTEGER    DEFAULT               1,
    returnRevocInfo           BOOLEAN    DEFAULT            TRUE,
    usefulCerts               UsefulCerts               OPTIONAL,
    usefulRevoc               UsefulRevoc               OPTIONAL,
    nonce                     INTEGER                   OPTIONAL,
```

```
       requestExtensions    [2]  EXPLICIT Extensions        OPTIONAL }

   Version ::=                  INTEGER  {  v1(0) }
```

```
   CertOrCertRef ::=  CHOICE {
       certificate          [1]  Certificate,
       certRef              [2]  OtherCertID }

   PathDiscoveryCriteria :: = CHOICE {
       trustpoints               SEQUENCE OF Certificate
                                 -- self-signed certificates
       valpolref                 ValPolicyRef }
```

version allows to identify the version of the protocol.

certUnderDiscovery is the certificate for which one or more path should be formed. It is either the actual value of the certificate or an unambiguous reference of that certificate: the issuer name and the serial number of the certificate, and a hash value of the certificate in order to guard against compromission of CA keys. Structures like ESSCertID can be used as a reference.

pathDiscoveryCriteria is either a sequence of self-signed root certificates or a reference to a validation policy to be used. This field MUST be present.

pathsNumber indicates the number of paths to be returned. The default value is one.

returnRevocInfo indicates whether or not revocation information should be returned. By default, revocation information is returned.

usefulCerts is a set of certificates, usually obtained through the application protocol that may be useful for the server to build a path.

usefulRevoc is a set of revocation information, usually obtained through the application protocol that may be useful for the server to make sure that no element from the path is revoked.

nonce is a unique number (e.g. a large pseudo random number) generated by the client that may be used to detect replay.

requestExtensions is a way to allow additional elements to be added later on, if needed.

6.2.2. Response Syntax

This section specifies the ASN.1 specification for a confirmation response. The actual formatting of the message could vary depending on the transport mechanism used (HTTP, SMTP, LDAP, etc.).

An DPD response at a minimum consists of a responseStatus field indicating the processing status of the prior request. If the value of responseStatus is one of the error conditions, discoveryResponse

is not present.

```
   DPDResponse     ::=     SEQUENCE {
       responseStatus              DPDResponseStatus,
       discoveryResponse           DiscoveryResponse         OPTIONAL }

   DPDResponseStatus ::= ENUMERATED {
       successful            (0),  -- request was understood
       malformedRequest      (1),  -- malformed request
       internalError         (2),  -- internal error in issuer
       tryLater              (3),  -- try again later
                                   -- (4) is not used
                                   -- (5) is not used
       unauthorized          (6)   -- request unauthorized }

   DiscoveryResponse ::= SEQUENCE {
       version             [0]     EXPLICIT Version      DEFAULT v1,
       discoveryStatus             DiscoveryStatus
       certUnderDiscovery          CertOrCertRef,
       pathDiscoveryCriteria       PathDiscoveryCriteria,
       producedAt                  GeneralizedTime,
       pathsDiscovered             PathsDiscovered       OPTIONAL,
       pathsNumberLimit            INTEGER               OPTIONAL,
       nonce                       INTEGER               OPTIONAL,
       responseExtensions  [2]     EXPLICIT Extensions   OPTIONAL }

   DiscoveryStatus ::= CHOICE {
     certswithRevoc        [0]     IMPLICIT NULL,
     certsNoRevoc          [1]     IMPLICIT NULL,
     noPathFound           [2]     IMPLICIT NULL }

   PathsDiscovered::= SEQUENCE OF PathValues
```

The various parameters from the DiscoveryResponse are the following:

version allows to identify the version of the protocol.

discoveryStatus indicates the result of the discovery according to path validation criteria.

When the response indicates "certswithRevoc", this means at least one path has been found, with partial or full revocation information is present.

When the response indicates "certsNoRevoc", this means that at least one path has been found, but no revocation information is present.

When the response indicates "noPathFound", this means that the no path has been able to be found.

certUnderDiscovery is the certificate for which one or more path should be formed. It is a copy of the field present in the request.

pathDiscoveryCriteria is either a sequence of self-signed root certificates or a reference to a validation policy to be used. It is a copy of the field present in the request.

producedAt is the time at which the response has been formed.

pathsDiscovered is a sequence of PathValues. Each sequence corresponds to one path.

pathsNumberLimit is an optional element that is used by the server to indicate that the number of paths that have been returned has been limited by the server to this value. Rules with a lower granularity should be used for further requests.

nonce is a pseudo random number generated by the client that may be used to detect replay.

requestExtensions is a way to allow additional elements to be added later on, if needed.

7. Components for a validation policy

A validation policy is build from three components:

1. Certificate requirements,
2. Revocation requirements,
3. Optional Time-Stamping requirements.

The ASN.1 construct for a validation policy definition is as follows:

```
ValPolicyDef :: =   SEQUENCE OF PolicyRequirement

PolicyRequirement :: = SEQUENCE {
    certificateRequirements     CertificateTrustPoint,
    revocRequirements           RevocRequirements,
    timeStampingRequirements    TimeStampingRequirements OPTIONAL }
```

7.1. Certificate Requirements

The certificateRequirements identifies a self-signed root certificate used to start (or end) certificate path processing and the initial conditions for certificate path validation as defined RFC 2459 [7] section 4.

```
CertificateTrustPoint ::= SEQUENCE {
    trustpoint               Certificate,
                            -- self-signed certificate
    pathLenConstraint    [0] PathLenConstraint   OPTIONAL,
    acceptablePolicySet  [1] AcceptablePolicySet OPTIONAL,
                            -- if not present "any policy"
```

```
       nameConstraints      [2] NameConstraints     OPTIONAL,
       policyConstraints    [3] PolicyConstraints   OPTIONAL }
```

The trustPoint field gives the self signed certificate for the CA that is used as the trust point for the start of certificate path processing.

The pathLenConstraint field gives the maximum number of CA certificates that may be in a certification path following the trustpoint. A value of zero indicates that only the given trustpoint certificate and an end-entity certificate may be used. If present, the pathLenConstraint field must be greater than or equal to zero. Where pathLenConstraint is not present, there is no limit to the allowed length of the certification path.

```
PathLenConstraint    ::=   INTEGER (0..MAX)
```

The acceptablePolicySet field identifies the initial set of certificate policies, any of which are acceptable under the signature policy.

```
AcceptablePolicySet ::= SEQUENCE OF CertPolicyId

CertPolicyId ::= OBJECT IDENTIFIER
```

The nameConstraints field indicates a name space within which all subject names in subsequent certificates in a certification path must be located. Restrictions may apply to the subject distinguished name or subject alternative names. Restrictions apply only when the specified name form is present. If no name of the type is in the certificate, the certificate is acceptable.

Restrictions are defined in terms of permitted or excluded name subtrees. Any name matching a restriction in the excludedSubtrees field is invalid regardless of information appearing in the permittedSubtrees.

```
    NameConstraints ::= SEQUENCE {
           permittedSubtrees       [0]     GeneralSubtrees OPTIONAL,
           excludedSubtrees        [1]     GeneralSubtrees OPTIONAL }

      GeneralSubtrees ::= SEQUENCE SIZE (1..MAX) OF GeneralSubtree

      GeneralSubtree ::= SEQUENCE {
           base                    GeneralName,
           minimum         [0]     BaseDistance DEFAULT 0,
           maximum         [1]     BaseDistance OPTIONAL }

      BaseDistance ::= INTEGER (0..MAX)
```

The policyConstraints extension constrains path processing in two ways. It can be used to prohibit policy mapping or require that each certificate in a path contain an acceptable policy identifier.

The policyConstraints field, if present specifies requirement for explicit indication of the certificate policy and/or the constraints on policy mapping.

```
   PolicyConstraints ::= SEQUENCE {
        requireExplicitPolicy           [0] SkipCerts OPTIONAL,
        inhibitPolicyMapping            [1] SkipCerts OPTIONAL }

SkipCerts ::= INTEGER (0..MAX)
```

If the inhibitPolicyMapping field is present, the value indicates the number of additional certificates that may appear in the path (including the trustpoint's self certificate) before policy mapping is no longer permitted. For example, a value of one indicates that policy mapping may be processed in certificates issued by the subject of this certificate, but not in additional certificates in the path.

If the requireExplicitPolicy field is present, subsequent certificates must include an acceptable policy identifier. The value of requireExplicitPolicy indicates the number of additional certificates that may appear in the path (including the trustpoint's self certificate) before an explicit policy is required. An acceptable policy identifier is the identifier of a policy required by the user of the certification path or the identifier of a policy which has been declared equivalent through policy mapping.

7.2. Revocation Requirements

The RevocRequirements specifies minimum requirements for revocation information, obtained through CRLs, delta-CRLs or OCSP responses, to be used in checking the revocation status of certificates. This ASN1 structure is used to define policy for validating the certificate.

```
   RevocRequirements ::= SEQUENCE {
       endCertRevReq       RevReq,
       caCerts             RevReq }
```

Certificate revocation requirements are specified in terms of checks required on:

* endCertRevReq: end certificates (i.e. the signers certificate, the attribute certificate or the timestamping authority certificate).

* caCerts: CA certificates.

```
   RevReq ::= BIT STRING {
           crlCheck           (0),
           ocspCheck          (1),
           deltaCrlCheck      (2) }
```

Bits in RevReq are used as follows:

The crlCheck bit is asserted when checks are to be done against

full CRLs (or full Authority revocation lists).

The ocspCheck bit is asserted when the revocation status check is to be done using the OCSP protocol (i.e. RFC 2560).

The deltaCrlCheck bit is asserted when checks are to be done against delta CRLs and relevant associated full CRLs (or full Authority revocation lists).

When more than one bit is set, then any of the indicated sources of revocation information MUST be used.

When a single bit is set, then the indicated source of revocation information MUST be used.

When no bit is set, then no revocation status check SHALL be done.

7.3. Time-Stamping requirements

The TimeStampingRequirements identifies the number of time stamps to be used over the references from the path and the name of the TSAs to be used.

```
TimeStampingRequirements ::= SEQUENCE OF GeneralName
```

8. Validation policy definition protocol

In order to validate a certificate, it is needed to refer to a validation policy through an OID or a URI. This reference can be temporary defined by the requester. This protocol allows to define a validation policy. It is up to server to decide how long a temporary defined validation policy should be kept in memory.

8.1. Request

The ASN.1 construct for the request is as follows:

```
VPDefRequest ::=     SEQUENCE {
       tbsRequest                  TbsVPDefRequest,
       optionalSignature   [0]     EXPLICIT Signature       OPTIONAL }

   TbsVPDefRequest      ::=   SEQUENCE {
       version             [0]     EXPLICIT Version         DEFAULT v1,
       requestorName       [1]     EXPLICIT GeneralName     OPTIONAL,
       valPolicyId                 ValPolicyID,
       valPolicyDef                ValPolicyDef,
       requestExtensions   [2]     EXPLICIT Extensions      OPTIONAL }
```

8.2. Response

The ASN.1 construct for the response is as follows:

```
   VPDefResponse ::=     SEQUENCE {
       vPDefStatus                 VPDefResponseStatus,
       tbsResponse                 TbsVPDefResponse        OPTIONAL }
```

```
   VPDefResponseStatus ::= ENUMERATED {
       successful            (0),  -- Request was understood
       malformedRequest      (1),  -- malformed request
```

```
       internalError         (2),  -- internal error in issuer
       tryLater              (3),  -- try again later
                                   -- (4) is not used

       sigRequired           (5),  -- must sign the request
       unauthorized          (6)   -- request unauthorized }

   TbsDefResponse       ::= SEQUENCE {
      tbsResponseData               VPDefResponseData,
      signatureAlgorithm            AlgorithmIdentifier   OPTIONAL,
      signature                     BIT STRING            OPTIONAL,
      certs                 [0]     EXPLICIT SEQUENCE OF Certificate
                                    OPTIONAL }
```

The value for signature SHALL be computed on the hash of the DER encoding of VPDefResponseData.

```
VPDefResponseData :: = SEQUENCE {
       responseStatus               VPDefResponseStatus,
       validationPolicyRef          ValidationPolicyRef     OPTIONAL }

vPDefResponseStatus ::= CHOICE {
       registered           [0]     IMPLICIT NULL,
       notregistered        [1]     IMPLICIT NULL }
```

When both the request is understood (i.e. there is no syntax error) and the validation policy can be correctly interpreted (i.e. there is no semantic error) then a validation policy reference is returned.

The identifier of the policy may be specific to the request or may be an identifier that has already been attributed as the result of a previous request with an identical definition.

9. Security considerations

To be provided.

10. Acknowledgments

To be provided.

11. References

[PKIX-1]

Internet X.509 Public Key Infrastructure.
Certificate and CRL Profile. RFC 2459
R. Housley, W. Ford, W. Polk, D. Solo.

or its successor as soon as it can be referenced.

[OCSP]

X.509 Internet Public Key Infrastructure.
Online Certificate Status Protocol û OCSP. RFC 2560
M. Myers, R. Ankney, A. Malpani, S. Galperin, C. Adams.

[TSP]

Internet X.509 Public Key Infrastructure
Time-Stamp Protocol (TSP). RFC YYYY
C. Adams, P. Cain, D. Pinkas, R. Zuccherato.

[ES-F]

Electronic Signature Formats for long term electronic signatures
D. Pinkas, J. Ross, N. Pope. RFC 3126 (to be published).

[CMS]

Cryptographic Message Syntax. RFC 2630.
R. Housley June 1999.

[ISO-X509]

ISO/IEC 9594-8/ITU-T Recommendation X.509, "Information Technology - Open Systems Interconnection: The Directory: Authentication Framework," 1997 edition. (Pending publication of 1997 edition, use 1993 edition with the following amendment applied: Final Text of Draft Amendment DAM 1 to ISO/IEC 9594-8 on Certificate Extensions, June 1996.)

12. Authors' addresses

Denis Pinkas
Integris, Bull.
68, Route de Versailles
78434 Louveciennes CEDEX
FRANCE
e-mail: Denis.Pinkas@bull.net

13. Full Copyright Statement

Annex A (normative): ASN.1 Definitions

To be provided.