                  X.509 Internet Public Key Infrastructure
                  Online Certificate Status Protocol - OCSP

Status of this Memo

   This document specifies an Internet standards track protocol for the
   Internet community, and requests discussion and suggestions for
   improvements. Please refer to the current edition of the "Internet
   Official Protocol Standards" (STD 1) for the standardization state
   and status of this protocol. Distribution of this memo is unlimited.

1. Abstract

   This document specifies a protocol useful in determining the current
   status of a digital certificate without requiring CRLs. Additional
   mechanisms addressing PKIX operational requirements are specified in
   separate documents.

   An overview of the protocol is provided in section 2. Functional
   requirements are specified in section 4. Details of the protocol are
   in section 5. We cover security issues with the protocol in section
   6. Appendix A defines OCSP over HTTP, appendix B accumulates ASN.1
   syntactic elements and appendix C specifies the mime types for the
   messages.

   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. Protocol Overview

   In lieu of or as a supplement to checking against a periodic CRL, it
   may be necessary to obtain timely information regarding the
   revocation status of a certificate (cf. [RFC2459], Section 3.3).
   Examples include high-value funds transfer or large stock trades.

   The Online Certificate Status Protocol (OCSP) enables applications to
   determine the (revocation) state of an identified certificate. OCSP
   may be used to satisfy some of the operational requirements of
   providing more timely revocation information than is possible with
   CRLs and may also be used to obtain additional status information. An
   OCSP client issues a status request to an OCSP responder and suspends
   acceptance of the certificate in question until the responder
   provides a response.

   This protocol specifies the data that needs to be exchanged between
   an application checking the status of a certificate and the server
   providing that status.

2.1 Request

   An OCSP request contains the following data:

   -- protocol version
   -- service request
   -- target certificate identifier
   -- optional extensions which MAY be processed by the OCSP Responder

   Upon receipt of a request, an OCSP 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 OCSP responder
   produces an error message; otherwise, it returns a definitive
   response.

2.2 Response

   OCSP responses can be of various types. An OCSP response consists of
   a response type and the bytes of the actual response. There is one
   basic type of OCSP response that MUST be supported by all OCSP
   servers and clients. The rest of this section pertains only to this
   basic response type.

All definitive response messages SHALL be digitally signed. The key
used to sign the response MUST belong to one of the following:

-- the CA who issued the certificate in question
-- a Trusted Responder whose public key is trusted by the requester
-- a CA Designated Responder (Authorized Responder) who holds a
   specially marked certificate issued directly by the CA, indicating
   that the responder may issue OCSP responses for that CA

A definitive response message is composed of:

-- version of the response syntax
-- name of the responder
-- responses for each of the certificates in a request
-- optional extensions
-- signature algorithm OID
-- signature computed across hash of the response

The response for each of the certificates in a request consists of

-- target certificate identifier
-- certificate status value
-- response validity interval
-- optional extensions

This specification defines the following definitive response
indicators for use in the certificate status value:

-- good
-- revoked
-- unknown

The "good" state indicates a positive response to the status inquiry.
At a minimum, this positive response indicates that the certificate
is not revoked, but does not necessarily mean that the certificate
was ever issued or that the time at which the response was produced
is within the certificate's validity interval. Response extensions
may be used to convey additional information on assertions made by
the responder regarding the status of the certificate such as
positive statement about issuance, validity, etc.

The "revoked" state indicates that the certificate has been revoked
(either permanantly or temporarily (on hold)).

The "unknown" state indicates that the responder doesn't know about
the certificate being requested.

2.3 Exception Cases

   In case of errors, the OCSP Responder may return an error message.
   These messages are not signed. Errors can be of the following types:

   -- malformedRequest
   -- internalError
   -- tryLater
   -- sigRequired
   -- unauthorized

   A server produces the "malformedRequest" response if the request
   received does not conform to the OCSP syntax.

   The response "internalError" indicates that the OCSP responder
   reached an inconsistent internal state. The query should be retried,
   potentially with another responder.

   In the event that the OCSP responder is operational, but unable to
   return a status for the requested certificate, the "tryLater"
   response can be used to indicate that the service exists, but is
   temporarily unable to respond.

   The response "sigRequired" is returned in cases where the server
   requires the client sign the request in order to construct a
   response.

   The response "unauthorized" is returned in cases where the client is
   not authorized to make this query to this server.

2.4 Semantics of thisUpdate, nextUpdate and producedAt

   Responses can contain three times in them - thisUpdate, nextUpdate
   and producedAt. The semantics of these fields are:

   - thisUpdate: The time at which the status being indicated is known
                 to be correct
   - nextUpdate: The time at or before which newer information will be
                 available about the status of the certificate
   - producedAt: The time at which the OCSP responder signed this
                 response.

   If nextUpdate is not set, the responder is indicating that newer
   revocation information is available all the time.

2.5 Response Pre-production

   OCSP responders MAY pre-produce signed responses specifying the
   status of certificates at a specified time. The time at which the
   status was known to be correct SHALL be reflected in the thisUpdate
   field of the response. The time at or before which newer information
   will be available is reflected in the nextUpdate field, while the
   time at which the response was produced will appear in the producedAt
   field of the response.

2.6 OCSP Signature Authority Delegation

   The key that signs a certificate's status information need not be the
   same key that signed the certificate. A certificate's issuer
   explicitly delegates OCSP signing authority by issuing a certificate
   containing a unique value for extendedKeyUsage in the OCSP signer's
   certificate. This certificate MUST be issued directly to the
   responder by the cognizant CA.

2.7 CA Key Compromise

   If an OCSP responder knows that a particular CA's private key has
   been compromised, it MAY return the revoked state for all
   certificates issued by that CA.

3. Functional Requirements

3.1 Certificate Content

   In order to convey to OCSP clients a well-known point of information
   access, CAs SHALL provide the capability to include the
   AuthorityInfoAccess extension (defined in [RFC2459], section 4.2.2.1)
   in certificates that can be checked using OCSP. Alternatively, the
   accessLocation for the OCSP provider may be configured locally at the
   OCSP client.

   CAs that support an OCSP service, either hosted locally or provided
   by an Authorized Responder, MUST provide for the inclusion of a value
   for a uniformResourceIndicator (URI) accessLocation and the OID value
   id-ad-ocsp for the accessMethod in the AccessDescription SEQUENCE.

   The value of the accessLocation field in the subject certificate
   defines the transport (e.g. HTTP) used to access the OCSP responder
   and may contain other transport dependent information (e.g. a URL).

3.2 Signed Response Acceptance Requirements

   Prior to accepting a signed response as valid, OCSP clients SHALL
   confirm that:

   1. The certificate identified in a received response corresponds to
   that which was identified in the corresponding request;

   2. The signature on the response is valid;

   3. The identity of the signer matches the intended recipient of the
   request.

   4. The signer is currently authorized to sign the response.

   5. The time at which the status being indicated is known to be
   correct (thisUpdate) is sufficiently recent.

   6. When available, the time at or before which newer information will
   be available about the status of the certificate (nextUpdate) is
   greater than the current time.

4. Detailed Protocol

   The ASN.1 syntax imports terms defined in [RFC2459]. 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

4.1 Requests

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

4.1.1 Request Syntax

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

   TBSRequest   ::=   SEQUENCE {

```
        version               [0]     EXPLICIT Version DEFAULT v1,
        requestorName         [1]     EXPLICIT GeneralName OPTIONAL,
        requestList                   SEQUENCE OF Request,
        requestExtensions     [2]     EXPLICIT Extensions OPTIONAL }

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

    Version         ::=                INTEGER  {  v1(0) }

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

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

issuerNameHash is the hash of the Issuer's distinguished name. The
hash shall be calculated over the DER encoding of the issuer's name
field in the certificate being checked. issuerKeyHash is the hash of
the Issuer's public key. The hash shall be calculated over the value
(excluding tag and length) of the subject public key field in the
issuer's certificate. The hash algorithm used for both these hashes,
is identified in hashAlgorithm. serialNumber is the serial number of
the certificate for which status is being requested.

4.1.2 Notes on the Request Syntax

The primary reason to use the hash of the CA's public key in addition
to the hash of the CA's name, to identify the issuer, is that it is
possible that two CAs may choose to use the same Name (uniqueness in
the Name is a recommendation that cannot be enforced). Two CAs will
never, however, have the same public key unless the CAs either
explicitly decided to share their private key, or the key of one of
the CAs was compromised.

Support for any specific extension is OPTIONAL. The critical flag
SHOULD NOT be set for any of them. Section 4.4 suggests several
useful extensions. Additional extensions MAY be defined in
additional RFCs. Unrecognized extensions MUST be ignored (unless they
have the critical flag set and are not understood).

The requestor MAY choose to sign the OCSP request. In that case, the
signature is computed over the tbsRequest structure. If the request
is signed, the requestor SHALL specify its name in the requestorName
field. Also, for signed requests, the requestor MAY include
certificates that help the OCSP responder verify the requestor's
signature in the certs field of Signature.

4.2 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.).

4.2.1 ASN.1 Specification of the OCSP Response

An OCSP 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, responseBytes are
not set.

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

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

The value for responseBytes consists of an OBJECT IDENTIFIER and a
response syntax identified by that OID encoded as an OCTET STRING.

```
ResponseBytes ::=         SEQUENCE {
   responseType   OBJECT IDENTIFIER,
   response       OCTET STRING }
```

For a basic OCSP responder, responseType will be id-pkix-ocsp-basic.

```
id-pkix-ocsp           OBJECT IDENTIFIER ::= { id-ad-ocsp }
id-pkix-ocsp-basic     OBJECT IDENTIFIER ::= { id-pkix-ocsp 1 }
```

   OCSP responders SHALL be capable of producing responses of the id-
   pkix-ocsp-basic response type. Correspondingly, OCSP clients SHALL be
   capable of receiving and processing responses of the id-pkix-ocsp-
   basic response type.

   The value for response SHALL be the DER encoding of
   BasicOCSPResponse.

```
   BasicOCSPResponse       ::= SEQUENCE {
      tbsResponseData      ResponseData,
      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 ResponseData.

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

   ResponderID ::= CHOICE {
      byName               [1] Name,
      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,
      nextUpdate        [0]   EXPLICIT GeneralizedTime OPTIONAL,
      singleExtensions  [1]   EXPLICIT Extensions OPTIONAL }

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

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

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

4.2.2 Notes on OCSP Responses

4.2.2.1 Time

   The thisUpdate and nextUpdate fields define a recommended validity
   interval. This interval corresponds to the {thisUpdate, nextUpdate}
   interval in CRLs. Responses whose nextUpdate value is earlier than
   the local system time value SHOULD be considered unreliable.
   Responses whose thisUpdate time is later than the local system time
   SHOULD be considered unreliable. Responses where the nextUpdate value
   is not set are equivalent to a CRL with no time for nextUpdate (see
   Section 2.4).

   The producedAt time is the time at which this response was signed.

4.2.2.2 Authorized Responders

   The key that signs a certificate's status information need not be the
   same key that signed the certificate. It is necessary however to
   ensure that the entity signing this information is authorized to do
   so. Therefore, a certificate's issuer MUST either sign the OCSP
   responses itself or it MUST explicitly designate this authority to
   another entity. OCSP signing delegation SHALL be designated by the
   inclusion of id-kp-OCSPSigning in an extendedKeyUsage certificate
   extension included in the OCSP response signer's certificate. This
   certificate MUST be issued directly by the CA that issued the
   certificate in question.

   id-kp-OCSPSigning OBJECT IDENTIFIER ::= {id-kp 9}

   Systems or applications that rely on OCSP responses MUST be capable
   of detecting and enforcing use of the id-ad-ocspSigning value as
   described above. They MAY provide a means of locally configuring one
   or more OCSP signing authorities, and specifying the set of CAs for
   which each signing authority is trusted. They MUST reject the
   response if the certificate required to validate the signature on the
   response fails to meet at least one of the following criteria:

   1. Matches a local configuration of OCSP signing authority for the
   certificate in question; or

   2. Is the certificate of the CA that issued the certificate in
   question; or

   3. Includes a value of id-ad-ocspSigning in an ExtendedKeyUsage
   extension and is issued by the CA that issued the certificate in
   question."

Additional acceptance or rejection criteria may apply to either the
response itself or to the certificate used to validate the signature
on the response.

4.2.2.2.1 Revocation Checking of an Authorized Responder

Since an Authorized OCSP responder provides status information for
one or more CAs, OCSP clients need to know how to check that an
authorized responder's certificate has not been revoked. CAs may
choose to deal with this problem in one of three ways:

- A CA may specify that an OCSP client can trust a responder for the
lifetime of the responder's certificate. The CA does so by including
the extension id-pkix-ocsp-nocheck. This SHOULD be a non-critical
extension. The value of the extension should be NULL. CAs issuing
such a certificate should realized that a compromise of the
responder's key, is as serious as the compromise of a CA key used to
sign CRLs, at least for the validity period of this certificate. CA's
may choose to issue this type of certificate with a very short
lifetime and renew it frequently.

id-pkix-ocsp-nocheck OBJECT IDENTIFIER ::= { id-pkix-ocsp 5 }

- A CA may specify how the responder's certificate be checked for
revocation. This can be done using CRL Distribution Points if the
check should be done using CRLs or CRL Distribution Points, or
Authority Information Access if the check should be done in some
other way. Details for specifying either of these two mechanisms are
available in [RFC2459].

- A CA may choose not to specify any method of revocation checking
for the responder's certificate, in which case, it would be up to the
OCSP client's local security policy to decide whether that
certificate should be checked for revocation or not.

4.3 Mandatory and Optional Cryptographic Algorithms

Clients that request OCSP services SHALL be capable of processing
responses signed used DSA keys identified by the DSA sig-alg-oid
specified in section 7.2.2 of [RFC2459]. Clients SHOULD also be
capable of processing RSA signatures as specified in section 7.2.1 of
[RFC2459]. OCSP responders SHALL support the SHA1 hashing algorithm.

4.4 Extensions

This section defines some standard extensions, based on the extension
model employed in X.509 version 3 certificates see [RFC2459]. Support
for all extensions is optional for both clients and responders. For

each extension, the definition indicates its syntax, processing
performed by the OCSP Responder, and any extensions which are
included in the corresponding response.

4.4.1 Nonce

The nonce cryptographically binds a request and a response to prevent
replay attacks. The nonce is included as one of the requestExtensions
in requests, while in responses it would be included as one of the
responseExtensions. In both the request and the response, the nonce
will be identified by the object identifier id-pkix-ocsp-nonce, while
the extnValue is the value of the nonce.

    id-pkix-ocsp-nonce   OBJECT IDENTIFIER ::= { id-pkix-ocsp 2 }

4.4.2 CRL References

It may be desirable for the OCSP responder to indicate the CRL on
which a revoked or onHold certificate is found. This can be useful
where OCSP is used between repositories, and also as an auditing
mechanism. The CRL may be specified by a URL (the URL at which the
CRL is available), a number (CRL number) or a time (the time at which
the relevant CRL was created). These extensions will be specified as
singleExtensions. The identifier for this extension will be id-pkix-
ocsp-crl, while the value will be CrlID.

    id-pkix-ocsp-crl   OBJECT IDENTIFIER ::= { id-pkix-ocsp 3 }

    CrlID ::= SEQUENCE {
        crlUrl   [0]   EXPLICIT IA5String OPTIONAL,
        crlNum   [1]   EXPLICIT INTEGER OPTIONAL,
        crlTime   [2]   EXPLICIT GeneralizedTime OPTIONAL }

For the choice crlUrl, the IA5String will specify the URL at which
the CRL is available. For crlNum, the INTEGER will specify the value
of the CRL number extension of the relevant CRL. For crlTime, the
GeneralizedTime will indicate the time at which the relevant CRL was
issued.

4.4.3 Acceptable Response Types

An OCSP client MAY wish to specify the kinds of response types it
understands. To do so, it SHOULD use an extension with the OID id-
pkix-ocsp-response, and the value AcceptableResponses. This
extension is included as one of the requestExtensions in requests.
The OIDs included in AcceptableResponses are the OIDs of the various
response types this client can accept (e.g., id-pkix-ocsp-basic).

```
id-pkix-ocsp-response  OBJECT IDENTIFIER ::= { id-pkix-ocsp 4 }

AcceptableResponses ::= SEQUENCE OF OBJECT IDENTIFIER
```

As noted in section 4.2.1, OCSP responders SHALL be capable of
responding with responses of the id-pkix-ocsp-basic response type.
Correspondingly, OCSP clients SHALL be capable of receiving and
processing responses of the id-pkix-ocsp-basic response type.

4.4.4 Archive Cutoff

An OCSP responder MAY choose to retain revocation information beyond
a certificate's expiration. The date obtained by subtracting this
retention interval value from the producedAt time in a response is
defined as the certificate's "archive cutoff" date.

OCSP-enabled applications would use an OCSP archive cutoff date to
contribute to a proof that a digital signature was (or was not)
reliable on the date it was produced even if the certificate needed
to validate the signature has long since expired.

OCSP servers that provide support for such historical reference
SHOULD include an archive cutoff date extension in responses. If
included, this value SHALL be provided as an OCSP singleExtensions
extension identified by id-pkix-ocsp-archive-cutoff and of syntax
GeneralizedTime.

```
id-pkix-ocsp-archive-cutoff  OBJECT IDENTIFIER ::= { id-pkix-ocsp 6 }

ArchiveCutoff ::= GeneralizedTime
```

To illustrate, if a server is operated with a 7-year retention
interval policy and status was produced at time t1 then the value for
ArchiveCutoff in the response would be (t1 - 7 years).

4.4.5 CRL Entry Extensions

All the extensions specified as CRL Entry Extensions - in Section 5.3
of [RFC2459] - are also supported as singleExtensions.

4.4.6 Service Locator

An OCSP server may be operated in a mode whereby the server receives
a request and routes it to the OCSP server which is known to be
authoritative for the identified certificate. The serviceLocator
request extension is defined for this purpose. This extension is
included as one of the singleRequestExtensions in requests.

```
    id-pkix-ocsp-service-locator OBJECT IDENTIFIER ::= { id-pkix-ocsp 7 }

    ServiceLocator ::= SEQUENCE {
        issuer    Name,
        locator   AuthorityInfoAccessSyntax OPTIONAL }
```

Values for these fields are obtained from the corresponding fields in
the subject certificate.

5. Security Considerations

For this service to be effective, certificate using systems must
connect to the certificate status service provider. In the event such
a connection cannot be obtained, certificate-using systems could
implement CRL processing logic as a fall-back position.

A denial of service vulnerability is evident with respect to a flood
of queries. The production of a cryptographic signature significantly
affects response generation cycle time, thereby exacerbating the
situation. Unsigned error responses open up the protocol to another
denial of service attack, where the attacker sends false error
responses.

The use of precomputed responses allows replay attacks in which an
old (good) response is replayed prior to its expiration date but
after the certificate has been revoked. Deployments of OCSP should
carefully evaluate the benefit of precomputed responses against the
probability of a replay attack and the costs associated with its
successful execution.

Requests do not contain the responder they are directed to. This
allows an attacker to replay a request to any number of OCSP
responders.

The reliance of HTTP caching in some deployment scenarios may result
in unexpected results if intermediate servers are incorrectly
configured or are known to possess cache management faults.
Implementors are advised to take the reliability of HTTP cache
mechanisms into account when deploying OCSP over HTTP.

6. References

   [RFC2459] Housley, R., Ford, W., Polk, W. and D. Solo, "Internet
             X.509 Public Key Infrastructure Certificate and CRL
             Profile", RFC 2459, January 1999.

   [HTTP]   Fielding, R., Gettys, J., Mogul, J., Frystyk, H. and T.
             Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC
             2068, January 1997.

   [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
             Requirement Levels", BCP 14, RFC 2119, March 1997.

   [URL]   Berners-Lee, T., Masinter, L. and M. McCahill, "Uniform
             Resource Locators (URL)", RFC 1738, December 1994.

   [X.690]   ITU-T Recommendation X.690 (1994) | ISO/IEC 8825-1:1995,
             Information Technology - ASN.1 encoding rules:
             Specification of Basic Encoding Rules (BER), Canonical
             Encoding Rules (CER) and Distinguished Encoding Rules
             (DER).

7. Authors' Addresses

Michael Myers
VeriSign, Inc.
1350 Charleston Road
Mountain View, CA 94043

EMail: mmyers@verisign.com

Rich Ankney
CertCo, LLC
13506 King Charles Dr.
Chantilly, VA 20151

EMail: rankney@erols.com

Ambarish Malpani
ValiCert, Inc.
1215 Terra Bella Ave.
Mountain View, CA 94043

Phone: 650.567.5457
EMail: ambarish@valicert.com

Slava Galperin
My CFO, Inc.
1945 Charleston Road
Mountain View, CA

EMail: galperin@mycfo.com

Carlisle Adams
Entrust Technologies
750 Heron Road, Suite E08
Ottawa, Ontario
K1V 1A7
Canada

EMail: cadams@entrust.com

Appendix A.

A.1 OCSP over HTTP

   This section describes the formatting that will be done to the
   request and response to support HTTP.

A.1.1 Request

   HTTP based OCSP requests can use either the GET or the POST method to
   submit their requests. To enable HTTP caching, small requests (that
   after encoding are less than 255 bytes), MAY be submitted using GET.
   If HTTP caching is not important, or the request is greater than 255
   bytes, the request SHOULD be submitted using POST. Where privacy is
   a requirement, OCSP transactions exchanged using HTTP MAY be
   protected using either TLS/SSL or some other lower layer protocol.

   An OCSP request using the GET method is constructed as follows:

   GET {url}/{url-encoding of base-64 encoding of the DER encoding of
   the OCSPRequest}

   where {url} may be derived from the value of AuthorityInfoAccess or
   other local configuration of the OCSP client.

   An OCSP request using the POST method is constructed as follows: The
   Content-Type header has the value "application/ocsp-request" while
   the body of the message is the binary value of the DER encoding of
   the OCSPRequest.

A.1.2 Response

   An HTTP-based OCSP response is composed of the appropriate HTTP
   headers, followed by the binary value of the DER encoding of the
   OCSPResponse. The Content-Type header has the value
   "application/ocsp-response". The Content-Length header SHOULD specify
   the length of the response. Other HTTP headers MAY be present and MAY
   be ignored if not understood by the requestor.

Appendix B. OCSP in ASN.1

OCSP DEFINITIONS EXPLICIT TAGS::=

BEGIN

IMPORTS

      -- Directory Authentication Framework (X.509)
            Certificate, AlgorithmIdentifier, CRLReason
            FROM AuthenticationFramework { joint-iso-itu-t ds(5)
                    module(1) authenticationFramework(7) 3 }

-- PKIX Certificate Extensions
            AuthorityInfoAccessSyntax
          FROM PKIX1Implicit88 {iso(1) identified-organization(3)
                  dod(6) internet(1) security(5) mechanisms(5) pkix(7)
                  id-mod(0) id-pkix1-implicit-88(2)}

          Name, GeneralName, CertificateSerialNumber, Extensions,
           id-kp, id-ad-ocsp
            FROM PKIX1Explicit88 {iso(1) identified-organization(3)
                  dod(6) internet(1) security(5) mechanisms(5) pkix(7)
                  id-mod(0) id-pkix1-explicit-88(1)};

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

TBSRequest   ::=   SEQUENCE {
    version   [0] EXPLICIT Version DEFAULT v1,
    requestorName   [1] EXPLICIT GeneralName OPTIONAL,
    requestList   SEQUENCE OF Request,
    requestExtensions   [2] EXPLICIT Extensions OPTIONAL }

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

Version ::= INTEGER { v1(0) }

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

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

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

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

ResponseBytes ::=        SEQUENCE {
    responseType   OBJECT IDENTIFIER,
    response       OCTET STRING }

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

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

ResponderID ::= CHOICE {
   byName   [1] Name,
   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,
```

```
    nextUpdate          [0]     EXPLICIT GeneralizedTime OPTIONAL,
    singleExtensions    [1]     EXPLICIT Extensions OPTIONAL }

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

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

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

ArchiveCutoff ::= GeneralizedTime

AcceptableResponses ::= SEQUENCE OF OBJECT IDENTIFIER

ServiceLocator ::= SEQUENCE {
    issuer    Name,
    locator   AuthorityInfoAccessSyntax }

-- Object Identifiers

id-kp-OCSPSigning         OBJECT IDENTIFIER ::= { id-kp 9 }
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 }
id-pkix-ocsp-crl         OBJECT IDENTIFIER ::= { id-pkix-ocsp 3 }
id-pkix-ocsp-response    OBJECT IDENTIFIER ::= { id-pkix-ocsp 4 }
id-pkix-ocsp-nocheck     OBJECT IDENTIFIER ::= { id-pkix-ocsp 5 }
id-pkix-ocsp-archive-cutoff  OBJECT IDENTIFIER ::= { id-pkix-ocsp 6 }
id-pkix-ocsp-service-locator OBJECT IDENTIFIER ::= { id-pkix-ocsp 7 }


END
```

Appendix C. MIME registrations

C.1 application/ocsp-request

   To: ietf-types@iana.org
   Subject: Registration of MIME media type application/ocsp-request

   MIME media type name: application

   MIME subtype name: ocsp-request

   Required parameters: None

   Optional parameters: None

   Encoding considerations: binary

   Security considerations: Carries a request for information. This
   request may optionally be cryptographically signed.

   Interoperability considerations: None

   Published specification: IETF PKIX Working Group Draft on Online
   Certificate Status Protocol - OCSP

   Applications which use this media type: OCSP clients

   Additional information:

      Magic number(s): None
      File extension(s): .ORQ
      Macintosh File Type Code(s): none

   Person & email address to contact for further information:
   Ambarish Malpani <ambarish@valicert.com>

   Intended usage: COMMON

   Author/Change controller:
   Ambarish Malpani <ambarish@valicert.com>

C.2 application/ocsp-response

   To: ietf-types@iana.org
   Subject: Registration of MIME media type application/ocsp-response

   MIME media type name: application

MIME subtype name: ocsp-response

Required parameters: None

Optional parameters: None
Encoding considerations: binary

Security considerations: Carries a cryptographically signed response

Interoperability considerations: None

Published specification: IETF PKIX Working Group Draft on Online
Certificate Status Protocol - OCSP

Applications which use this media type: OCSP servers

Additional information:

Magic number(s): None
File extension(s): .ORS
Macintosh File Type Code(s): none

Person & email address to contact for further information:
Ambarish Malpani <ambarish@valicert.com>

Intended usage: COMMON

Author/Change controller:
Ambarish Malpani <ambarish@valicert.com>

Full Copyright Statement

Acknowledgement