PKIX Working Group
INTERNET-DRAFT
Expires in six months

M. Branchaud
Xcert
April 1998

Internet Public Key Infrastructure
Caching the Online Certificate Status Protocol
draft-ietf-pkix-ocsp-caching-00.txt

1. Status of this Memo

This document is an Internet-Draft. 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 6 months and may be updated, replaced, or rendered obsolete 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."

To view the entire list of current Internet-Drafts, please check the "1id-abstracts.txt" listing contained in the Internet-Drafts Shadow Directories on ftp.is.co.za (Africa), ftp.nordu.net (Northern Europe), ftp.nis.garr.it (Southern Europe), munnari.oz.au (Pacific Rim), ftp.ietf.org (US East Coast), or ftp.isi.edu (US West Coast).

Distribution of this document is unlimited.

This Internet Draft expires in October 1998.

2. Abstract

The Online Certificate Status Protocol [OCSP] specifies how a client process may obtain certificate status information online from a server. In order for OCSP to scale beyond small communities of users, a method to cache certificate status information at intermediary servers is required.

This document describes the requirements of and assumptions behind OCSP caching, and defines the mechanisms through which such caching can be accomplished.

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].

3. Change Log

This is the first draft of this document.

4. Contents

5. Introduction

OCSP can allow a certificate processing entity to obtain certificate status information directly from a Certification Authority (CA). However, on the global Internet, it is impractical for any entity to directly contact the CAs of each certificate it encounters. For one thing, such behavior would require that the entity securely obtain a copy of every CA's OCSP responder key. For another, CAs would quickly find themselves overwhelmed by the volume of OCSP requests.

One solution to these impracticalities is to employ OCSP servers as intermediaries between the entities and the CAs, and to allow those servers to cache certificate status information. While this allows CAs to distribute their OCSP loads, caching introduces an extra layer of complexity that must be addressed.

In particular, cached data can not be transparently passed to a querying entity. Because of the way certificates are used, it is important for an entity to know how long a status value has been cached. The entity may be much more willing to accept a particular status response if it was created two seconds ago than if it is two weeks old.

The nature of certificate-using systems also requires that they be able to override OCSP caches as needed. An entity's policies for most of its applications may allow it to accept status information that has been cached for, say, less than a day. However, those policies may also dictate that for a particular application it must obtain status that has been cached for no more than 1 hour.

The OCSP caching mechanisms described in this document are designed to permit this kind of flexibility. The following section presents an overview and example of cached OCSP, to illustrate the basic operations expected of OCSP participants. The details of OCSP caching are described in Section 7. Definitions of the OCSP extensions required for caching are presented in Section 8. OCSP's caching closely resembles that defined in HTTP 1.1, and a discussion of the relationship between the two systems is in Section 9.

6. Cached OCSP Overview

This section describes the requirements and assumptions behind cached OCSP by presenting an overview of the protocol. We first define the entities that participate in cached OCSP, then provide an example of the protocol's operation.

6.1 OCSP Entities

Three types of entities participate in the cached OCSP protocol:

- Clients - entities which request a certificate's status;
- Servers - entities which cache status information and respond to clients' requests;
- CAs - the entities with authoritative information about the status of some certificates.

6.1.1 OCSP Clients

OCSP clients request certificate status information from OCSP servers. An OCSP client MAY be another OCSP server or CA, or it MAY be an "end" entity: one that originates a status request. Any client MAY cache a response. Indeed, OCSP servers usually cache the responses they receive, but OCSP's caching mechanisms are flexible enough to allow end entities to also cache responses and use them, for example, in offline operations.

In general, clients that are not themselves OCSP servers SHOULD use only a few, usually one, OCSP server. This is because such clients need to have complete trust in their OCSP server's public key, to the same extent that they would trust a root CA's key. In particular, clients MUST obtain a server's key in a trusted fashion as it would the key of a trusted root CA. When a client obtains a server's key in this way, the client is said to "trust" the server. Clients MAY accept in-band revocation notification of a server's key, however they MUST NOT accept an in-band replacement for a revoked server key. When a client is made aware of the revocation of a server's key, whether through in-band or out-of-band notification, it MUST NOT use that server for OCSP processing until it obtains a new key for the server via an out-of-band channel.

6.1.2 OCSP Servers

OCSP servers reply to OCSP requests. In creating a response, a server MAY used cached information if it is appropriate, or it MAY forward the requests to other OCSP servers.

When an OCSP server receives a query, it first checks its local store of certificate information to see if it can fulfill the request with cached data. This local store is mostly made up of cached information but can also include authoritative information from a co-located CA (see below). If the OCSP server can fulfill the request from its local store, it does so. Otherwise, the server MAY make one or more queries to other OCSP servers that it trusts. If it receives a definitive response from those queries (as opposed to a "status unknown" or error response), it SHOULD store that information in its local cache before returning the response to the client. Note that the client need not have any knowledge of the keys of the other OCSP servers.

6.1.3 OCSP CAs

All OCSP CAs are also OCSP servers and act in an identical fashion except that there is no need for an OCSP CA to "cache" the status of its own certificates. Any CA that supports cached OCSP MUST operate an OCSP server, and that server is said to be "co-located" with that CA. A single OCSP server MAY be co-located with more than one CA. When a caching OCSP server is co-located with a CA, it does not cache certificate status information for that CA's certificates. It instead has access to the CA's own authoritative information about those certificates. That is, the co-located OCSP server is the source of its CA's OCSP information. It is most likely that the co-located OCSP server has direct, local access to, or is even integrated with, the CA's directory service.

Authoritative certificate information consists simply of the certificate and its status. How these two pieces of information are associated is beyond the scope of this document. The certificate's directory entry MAY have a status attribute, or the certificate's status MAY be defined by how the certificate is stored in the directory, or status MAY be determined by some other means. Regardless, a co-located OCSP server MUST have direct and immediate access to a CA's internal certificate status information. The co-located OCSP server MUST NOT receive that information via a CRL or some other periodic means. Note that a server that is not co-located with a CA MAY receive status information from that CA periodically.

This document uses the term "CA" interchangeably with "co-located OCSP server."

6.2 OCSP Example

The best way to understand OCSP caching is to study an example of it in operation. This section presents such an example that depicts the

typical behavior expected of the various OCSP entities. This example

glosses over the finer points of OCSP caching in order to focus on the basic principles of the protocol.

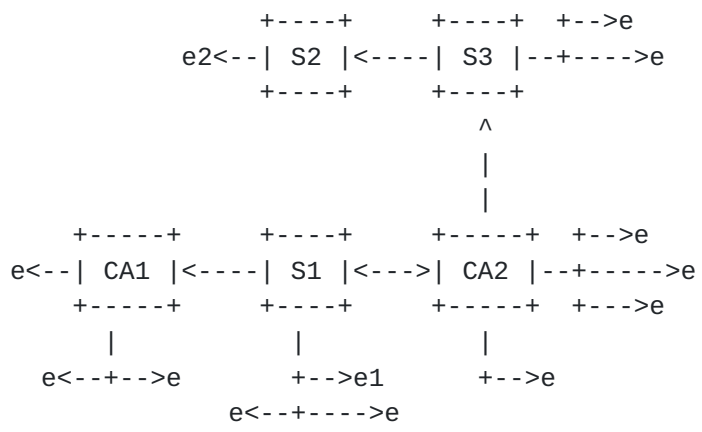

Figure 1 -- OCSP Entity Relationships

Figure 1 depicts the typical relationships among the OCSP entities (end entity clients are represented by a lowercase e, servers by an S and CAs by the letters CA). The arrows indicate the distribution of OCSP public key values: The entity at the source of an arrow has distributed its public key to the entity at the head of an arrow. These OCSP keys are distributed securely, as described above.

OCSP queries flow oppositely to the directions depicted by the arrows. Let's say that e1, an end entity client of S1, wishes to determine the status of a certificate, C1, issued by CA1. Assuming that no entity has cached any information yet, e1 would send a status query to S1. Since S1 has obtained CA1's public key, it "recognizes" the issuer of certificate C1 and sends its own query to that issuer: CA1.

CA1 returns a response containing the certificate's status. S1 can verify the authenticity of the response because it knows CA1's public OCSP key. S1 can now cache and return the response to e1, and e1 can authenticate S1's response because it knows S1's public key. In this way, e1 can obtain C1's status without having direct knowledge of CA1's public OCSP key. If e1 makes another query for the status of C1, S1 can return its cached value if it's acceptable to e1.

Let's add an extra level of indirection and consider how e2, a client of S2, would obtain the status of the a certificate, C2, issued by CA2. When e2 sends its query to S2, the server does not recognize C2's issuer because the server has not securely obtained CA2's public OCSP key. However, the server can be configured to query a number of

default servers when it encounters a certificate that it doesn't recognize. Here, S2 would send a query to S3, and now the process would proceed in the same manner as before, with both S3 and S2 caching the response. In fact, S3 would act no differently than it would if the query came from any other kind of client.

Note that the clients of CA1 can not obtain the status of any certificates issued by CA2 since CA1 does not have a copy of any of the servers' public keys, and so it can not make any external OCSP queries. However, clients of CA2 can obtain the status of CA1's certificates, since CA2 has exchanged public keys with S1 (notice the two-way arrow between S1 and CA2).

As we conclude this section, notice that the OCSP relationships between entities represent the trust relationships for the PKI. That is, OCSP relationships are similar to those established under cross-certification. If one entity trusts another to provide status information, then it can become an OCSP client to that entity by obtaining its public key. Thus, trust relationships are expressed through an explicit decision to trust an entity for status information. Also, an entity can require its consent before it is trusted, by simply refusing to answer any queries from entities that it doesn't recognize.

This discussion glossed over the finer points of OCSP caching. The next section deals with caching, but first note that OCSP caching does not change the mapping between OCSP relationships and trust relationships, nor does it affect the basic OCSP procedures described above. Instead, the caching allows the end-to-end requirement for OCSP messages - that a query must reach the authoritative CA - to be relaxed when appropriate.

7. OCSP Caching

OCSP uses caching to improve performance across the network. Caching certificate status information leads to the possibility of a cached status value becoming invalid but still being accepted as accurate. In an extreme sense, the same could be said of any copy of certificate status held outside the CA (such as a CRL). The value being copied could even become invalid in transit, and so the copy could be invalid before it even exists. Physical limitations, prevent absolutely accurate status information from being instantly available to all concerned parties. Even if such propagation were possible, network bandwidth is a scarce resource that should be used sparingly. OCSP's caching is designed with these constraints in mind to provide the most practical, online propagation of certificate status information. In particular, OCSP caching was designed to meet the following criteria:

- A relying party MUST always be able to override any intermediary caches between it and the CA, so that it MAY (at its discretion) obtain the most up-to-date status information possible. OCSP clients can formulate their requests with a number of cache parameters that allow them to finely tune requests to suit their needs.

- A relying party MUST always be given enough information to determine if a given certificate status value is acceptable for its purpose. To this end, OCSP clients are provided with the amount of time that has elapsed since the status value was generated by the CA. This elapsed time is called the age of the status value.

- A CA MUST always be able to specify how the status for a particular certificate should be cached. OCSP CAs MAY include recommended caching parameters in their replies that other OCSP entities SHOULD observe. Note that entities may be forced to ignore a CA's caching requirements (for example, if the entity can't establish a connection to the CA), but do so at their own peril.

With these criteria in mind, we now describe OCSP caching. First, we define how a server determines if it can respond to a query with a cached value.

7.1 Correctness of Cache Entries

An OCSP cache entry is said to be correct when an OCSP server can use the entry to reply to a request. An OCSP cache entry is correct if it meets either of the following criteria:

1. It is "fresh enough" (see Section 7.2.2). By default, this means that it meets the caching requirements of the client, the server and the CA.;

2. It includes a warning if the cache requirements of the client or the CA are not met.

If an OCSP server receives a response that it would normally forward to a client, and that response is no longer fresh, the server should forward the response to the client without attaching any warnings. An OCSP server SHOULD NOT attempt to revalidate a response that became stale in transit, as this might lead to an infinite loop.

Whenever an OCSP server returns a response that is not "fresh enough" it MUST attach an appropriate warning to the response (see Section 8.3).

7.2 Age Calculations

In order to know if a cache entry is fresh, its current age MUST be compared to its freshness lifetime (the age at which it becomes stale). This section describes how to calculate those two values, and how to determine if an entry is fresh.

7.2.1 Current Age

OCSP entities calculate the current age of a response using the following values:

now: The current local time of the entity performing the age calculation.

producedAt: This value is the time at which the response was generated by the CA. CAs MAY include this value in their responses. When a response is generated from a CRL, the value of producedAt SHOULD be the value of the thisUpdate field of the CRL.

requested_time: This is the receiving entity's local time when it sent the request that triggered this response.

received_time: This is the receiving entity's local time when it received the response.

age_value: This is the value of the age field in the response. This is the current age of the response as calculated by the entity sending the response at the time of transmission. All entities MUST include an age value in any response. OCSP ages are expressed in seconds.

The calculation is outlined below.

```
apparent_age       = max (0, received_time - producedAt)
received_age       = max (apparent_age, age_value)
entry_creation_age = received_age + (received_time - requested_time)
entry_local_age    = now - received_time
current_age        = entry_creation_age + entry_local_age
```

If the response does not include a producedAt value, then the received_age SHOULD be set to age_value and the first two steps above SHOULD be skipped.

Essentially, this calculation provides a conservative estimate of the sum of the number of seconds the response has been resident in each of the caches between the calculating entity and the CA, plus the number of seconds the response has spent in transit.

When an OCSP server sends a response to any entity, it MUST calculate its current_age and include it as the value of the age field in the response. CAs SHOULD provide an age value of 0 in their responses. This helps inquirers to determine if the response comes directly from an authoritative source.

Finally, note that if there is a significant difference between the clocks of the calculating entity and the CA, this calculation may lead to inordinately old age values. For this reason, entities MAY ignore the producedAt value in a response and proceed as if it were not present.

7.2.2 Freshness Calculation

The freshness lifetime of a cache entry is defined by the maxAge value specified by the CA in its authoritative response to a request. If this value is not present in the response, the calculating entity MAY use a locally-defined heuristic to determine the entry's freshness lifetime. For example, an entity might be configured with a 24 hour freshness lifetime for (entries for) email certificates, while it may only have a 5 second freshness lifetime for any aircraft-carrier-purchasing certificates.

Once a freshness lifetime is obtained, the freshness of an entry is calculated by simply comparing its freshness lifetime to its current_age:

```
entry_is_fresh = (freshness_lifetime > current_age)
```

7.2.3 Disambiguating Multiple Responses

A client MAY send out more than one request message to determine the status of a single certificate (e.g. a query MAY be sent to several OCSP servers). Thus, an entity may receive responses from multiple paths. To disambiguate these responses, the client SHOULD use the one with the lowest age value.

7.3 Server Cache Management

Servers are at liberty to manage their caches in any way they see fit. This section merely presents some recommendations that they MAY wish to adopt.

When a server recovers from a crash or is restarted after being down for some reason, it SHOULD erase its cache. This provides the most secure and easiest to implement startup environment. If a server elects to keep its cache data between downtimes, it MUST at least ensure that the current ages of all the entries are appropriately adjusted for the missing time.

Servers MAY elect to refresh their caches periodically. A server may "pull" status information for multiple certificates by including all the certificates in a single "batch" query submitted to the CA. The point here is that a server need not wait for a query to trigger a refresh of a cached value.

7.4 OCSP Cache Entry Validation

When an OCSP query includes a status value for the identified certificate (see Section 8.2), the query is called a "validation". Only CAs MAY reply to validations, and they MUST only reply to validations of certificates for which they are authoritative (a CA, and other servers, MAY still relay the validation to another CA or server, and return their response). OCSP servers MUST NOT use cached data to reply to a validation. A CA MAY reply to a validation with a normal response, or it MAY follow the procedure described in this section.

The purpose of a validation is to provide entities with a fast and efficient method to determine if a certificate's status has changed since it was last obtained. This is often done to see if a stale entry is still accurate (if so, the entry MAY be refreshed). The client includes its current notion of the certificate's status in the query. Servers relay the query to the certificate's CA. When the CA receives this message, it compares the presented status with the certificate's actual current status.

If the statuses match, then the CA MAY return a response consisting of the certID and a responseStatus field with the value "unchanged" (see Section 8.1). The CA MAY include any optional fields or extensions in the response.

If the statuses do not match (i.e. the status and / or time values are different), then the CA MUST respond to the query as if it were a normal query. The CA MUST ignore the presumedStatus value presented in the validation query (see Section 8.2).

An entity that receives a responseStatus value of "unchanged" in response to a validation may recalculate the current age of its cache entry for the response's certificate. It MUST do so by setting received_age equal to the response's age value in the response and skipping the first two steps of the calculation described in Section 7.2.1. If a server is configured to perform this recalculation, and it is going to forward the response of the validation to a client, then it MUST perform the recalculation before relaying the response. That is, the age value of the relayed response MUST be set to the newly calculated current_age.

8. Caching Enhancements to OCSP

This sections presents detailed definitions of the additions and extensions OCSP requires for caching, and further specifies the behavior of the OCSP entities. Most of these enhancements are defined as extensions to the basic OCSP protocol. These extensions are not optional for cached OCSP, and implementations MUST support them as required below.

8.1 The unchanged Certificate Status Value

A new certificate status is defined for cached OCSP: unchanged.

```
OCSPResponseStatus ::= ENUMERATED {
    successful         (0),   -- Response has valid confirmations
    malformedRequest   (1),   -- Illegal confirmation request
    internalError      (2),   -- Internal error in issuer
    tryLater           (3),   -- Try again later
    notFound           (4),   -- Certificate not on record
    certRequired       (5),   -- Must supply certificate
    unchanged          (6)    -- Status has not changed
}
```

The unchanged state is used in response to an OCSP cache entry validation (see Section 7.4). A CA MUST NOT use this state in response to a normal query. When responding to a validation, a CA MAY use this state if the current state of the certificate being validated matches the presumed state sent in the validation.

8.2 The Presumed Status Extension

This extension is used to create validation queries. That is, a query that includes this extension MAY be treated as a validation query. Responses MUST NOT contain this extension. It contains the client's current notion of the certificate's status.

```
presumedStatus EXTENSION ::= {
    SYNTAX PresumedStatusSyntax
    IDENTIFIED BY id-ocsp-presumedStatus
}

PresumedStatusSyntax ::= OCSPResponseStatus

id-ocsp-presumedStatus OBJECT IDENTIFIER ::= TBA
```

OCSP clients MUST NOT use the "unchanged" status value in a validation query. CAs which receive a query with this extension MAY treat the query as a validation (see Section 7.4), unless the

extension's value is

set to "unchanged." In that case the CA MUST NOT treat the query as a validation. Note that queries which include an "unchanged" value in this extension do not conform to the cached OCSP protocol.

This extension SHALL NOT be marked critical.

8.3 The Cache Warnings Extension

This extension contains warnings about the cache status of a certificate status response. This extension SHALL NOT be marked critical. It is defined as follows:

```
cacheWarnings EXTENSION ::= {
    SYNTAX CacheWarningsSyntax
    IDENTIFIED BY id-ocsp-cacheWarnings
}

CacheWarningsSyntax ::= SEQUENCE SIZE (1..MAX) of
SingleCacheWarning

SingleCacheWarning ::= SEQUENCE {
  warning      CacheWarningValue,
  text         UTF8String    OPTIONAL,
  warningData  OCTET STRING  OPTIONAL
}

CacheWarningValue ::= INTEGER {
    stale                   (0),
    revalidationFailed      (1),
    disconnectedOperation   (2)
}

id-ocsp-cacheWarnings OBJECT IDENTIFIER ::= TBA
```

Each warning is assigned a number and MAY include an explanatory text and/or some additional data. Responses MAY include more than one warning. Some warnings MUST be preserved by OCSP servers. That is, when an OCSP server receives a response that contains such a warning, it MUST pass that warning along when it relays the response, whether directly or from its cache, to a client. Warnings that must be preserved are identified in their definitions below.

The following warnings are defined. Other warnings may be defined in other documents.

stale (0): OCSP servers MUST include this warning whenever they return a response using stale cached data. A server MAY add this warning to any response, but MUST NOT remove it until the response is known to be fresh. Servers MUST preserve this warning.

revalidationFailed (1): OCSP servers MUST include this warning if they return a stale response because attempts to revalidate failed. A server MAY add this warning to any response, but MUST NOT remove it until the response is successfully revalidated. Servers MUST preserve this warning.

disconnectedOperation (2): OCSP servers SHOULD include this warning in all responses if the server is aware that it is not connected to the rest of the network. A server MAY conclude it is not connected after a number of network operations fail, or it MAY be told it is not connected by an administrator. Servers MUST preserve this warning. An OCSP server MUST include this warning when it can not reply to a noCache query with authoritative data (either from its own store or from another OCSP server). See the description of the noCache parameter in Section 8.5

8.4 The Cache Status Information Extension

This extension contains information about the age of the status data in the response. It consists of two fields:

age: This is the age, in seconds, of the response when it was sent by an OCSP server. All OCSP servers MUST include a value for age in all their responses. When a co-located OCSP server responds to a request about a certificate for which it is authoritative, it MUST include an age value of 0 in the response. See Section 7.2.1 for details on how the age value is otherwise calculated.

producedAt: This is the local time when the authoritative CA generated the response. Only OCSP servers that are co-located with a CA MAY include this value in a response. An OCSP server MUST NOT create this value in a response for a certificate for which it is not authoritative. An OCSP server that receives and caches a response containing a producedAt value MUST NOT modify or remove it when the response is used to reply to queries.

This extension SHALL NOT be marked critical.

```
  cacheStatusInfo EXTENSION ::= {
      SYNTAX CacheStatusInfoSyntax
      IDENTIFIED BY id-ocsp-cacheStatusInfo
  }
```

```
CacheStatusInfoSyntax ::= SEQUENCE {
     age                 INTEGER,
     producedAt          GeneralTime    OPTIONAL
}

id-ocsp-cacheStatusInfo OBJECT IDENTIFIER ::= TBA
```

8.5 The Request Cache Parameters Extension

The request cache parameters extension allows the client to specify required cache characteristics for the response. Those characteristics are defined by the following fields:

noCache: The client requests that the response be retrieved from the CA, i.e. that any intermediary OCSP servers ignore their caches when replying to this request. A server that receives a request with the noCache parameter MUST NOT reply with cached data. The server MUST either reply with authoritative information, or it MUST forward the request, including the noCache parameter, to another OCSP server. If a server is unable to do either, then it MUST reply with a status value of unknown accompanied by a "disconnected operation" warning.

maxAge: The client requests that the response can come from a cache provided it is no older than maxAge seconds. See Section 7.2 for a description of cache entry age calculations. If a server receives a request that specifies a maxAge of 0 then it MUST attempt to validate its entry (see Section 7.4), or retrieve a fresh response from another OCSP server or, if appropriate, from its authoritative store. Only if those attempts are unsuccessful MAY a server return a cached response older than the specified maxAge, and in doing so the server MUST include a staleness warning with the response (see Section 8.3).

minFresh: The server MAY return a cached response as long as that response it at least minFresh seconds away from becoming stale. See Section 7.2 for a definition of stale.

maxStale: The server MAY return a cached response as long as no more than maxStale seconds have elapsed since the response became stale. See Section 7.2for a definition of stale.

noValidate: The client will accept a stale response from the server. That is, the server MAY return a stale cache entry without first validating it. Note that any stale response MUST always include an appropriate warning (see Section 8.3).

The maxAge parameter MAY be combined with either the minFresh or the

maxStale parameters. When a server receives a request containing either combination, it MUST reply with a cached entry only if that entry satisfies both parameters, or all attempts to retrieve a satisfactory response from other servers are unsuccessful (in which case the server MUST include a warning in the response).

The noCache parameter MUST NOT be combined with any other parameter, while the noValidate flag MAY accompany any other parameter (except noCache).

This extension SHALL NOT be marked critical.

```
requestCacheParameters EXTENSION ::= {
    SYNTAX RequestCacheParametersSyntax
    IDENTIFIED BY id-ocsp-requestCacheParameters
}

RequestCacheParametersSyntax ::= SEQUENCE {
  noCache     BOOLEAN  OPTIONAL,
  maxAge      INTEGER  OPTIONAL,
  minFresh    INTEGER  OPTIONAL,
  maxStale    INTEGER  OPTIONAL,
  noValidate  BOOLEAN  OPTIONAL
}

id-ocsp-requestCacheParameters OBJECT IDENTIFIER ::= TBA
```

8.6 The Response Cache Parameters Extension

This extension allows a CA to specify required caching characteristics for the response. That is, the CA MAY specify how the client's cache (if any) should handle the response.

A server that receives a response containing this extension MUST NOT remove or alter the extension when sending replies based on that response. That is, the server MUST preserve this extension in the same way that some warnings must be preserved (see Section 8.3).

Response cache parameters are defined by the following fields:

noCache: The receiver MUST NOT cache the response at all.

maxAge: The receiver MAY cache the response but MUST consider it stale once the cached entry's current age exceeds maxAge. Responses with a maxAge of 0 MUST be revalidated every time they are used. See Section 7.2.2 for a description of how this field is used in cache entry freshness calculations.

Implementations MUST NOT generate responses that include both noCache and maxAge. The ASN.1 code for this extension precludes this.

This extension SHALL NOT be marked critical.

```
responseCacheParameters EXTENSION ::= {
    SYNTAX ResponseCacheParametersSyntax
    IDENTIFIED BY id-ocsp-responseCacheParameters
}

ResponseCacheParametersSyntax ::= CHOICE {
  noCache   [0]  BOOLEAN,
  maxAge    [1]  INTEGER
}

id-ocsp-responseCachParameters OBJECT IDENTIFIER ::= TBA
```

9. HTTP and OCSP Caching

[
Author's Note -- The issues surrounding an HTTP proxy caching OCSP responses are largely unresolved. Here's a quick list of some problems:

1. HTTP proxies can't sign responses. Therefore, they can't provide a signed record of how old a response is. That may be acceptable in some circumstances (e.g. when non-repudiation of responses isn't required), but if so the response should still be transmitted over a TLS (or SSL) authenticated channel. That is, the client should authenticate the proxy before accepting a cached response from it. The problem here is that HTTP proxies are normally transparent to HTTPS requests.

2. Can an OCSP client really tell an HTTP proxy to refresh its cache on demand (if, say, the client won't accept an entry older than x, but the age of the HTTP proxy's entry is > x)?

3. There are many broken HTTP caching implementations. Blindly relying upon them to perform OCSP caching properly can lead to problems.

Since these and other issues still need to be resolved, this section currently reflects only the author's views of HTTP caching. The fact that what's here may seem somewhat muddled only reflects the need for resolution.
]

Most of OCSP's caching mechanism is borrowed from HTTP 1.1. Readers familiar with [HTTP] will have recognized the similarities. The main differences are:

- OCSP's caching is transport independent (OCSP can be implemented over protocols other than HTTP);

- OCSP cache parameters can be signed to allow for a non-repudiable record of request and response parameters;

- OCSP responses are always origin-authenticated, either via signed responses or SSL/TLS.

This section describes the relationship between OCSP caching parameters and HTTP Cache-control directives.

When OCSP is transported over HTTP 1.1 or higher, the caching parameters of the OCSP messages SHALL take precedence over any Cache-

control directives in the HTTP messages. Where an implementation wishes to use HTTP Cache-control directives when transmitting OCSP messages, it SHOULD ensure that corresponding HTTP directives and OCSP cache parameters, as defined below, have the same values (although see below). HTTP cache-control parameters MUST NOT be used as a replacement for OCSP caching parameters. The following table shows the equivalences between OCSP caching parameters and HTTP 1.1 Cache-control directives.

OCSP Caching Parameter	HTTP 1.1 Cache-control Directive or Header
noCache	no-cache directive
maxAge	max-age directive
minFresh	min-fresh directive
maxStale	max-stale directive
noValidate	only-if-cached directive
producedAt	Date header
age	Age header

The above correlations are not an invitation to blindly use a plain HTTP caching proxy as an OCSP cache. HTTP proxy caching is uneven territory. Many HTTP caches do not function properly, and many also do not support the HTTP 1.1 directives. What is particularly troubling is that transporting OCSP requests over HTTP can lead to the client receiving invalid OCSP responses if there is an intermediate HTTP proxy. Consider the situation in Figure 2.

Figure 2 -- An HTTP Proxy in OCSP

If the client uses HTTP to contact the CA, responses from the CA might be transparently cached by the proxy. If a previous response from the CA had an age value of, say, 10 seconds, a client making a second query (for the same certificate) through the HTTP Proxy might receive the cached 10-second-old response, which may by then be wrong. The client has no way of knowing if the response came from the CA or from the cache. The client's only recourse would be to make sure it never received a cached entry (by, say, using the nonce extension).

Because of these issues, this profile recommends that HTTP caching SHOULD NOT be used for OCSP. When an OCSP message is sent via HTTP, the HTTP no-cache directive SHOULD be used. Although a properly-

functioning HTTP 1.1 proxy MAY be employed as an OCSP cache, implementations MUST NOT assume that an HTTP proxy they're dealing with is functioning properly. Also, note that simply because a directly-connected HTTP cache is functionally correct does not mean that any other caches it contacts will be, and the OCSP client has no way of determining the compliance of any HTTP cache. Note, too, that some caches even ignore the no-cache directive.

At a minimum, if use of an HTTP proxy is unavoidable (say, to transport OCSP across a firewall) then that proxy MUST at least recognize and obey the no-cache directive. However, since an implementation has no way of determining if this is true, it is up to installation administrators to ensure that their HTTP proxies are compliant.

Still, in general, it is better to avoid HTTP caching for OCSP.

10. Security Considerations

Many attacks, such as spoofing, are attenuated by the authentication inherent in the basic OCSP protocol. The main threat that basic OCSP does not address is a denial of service attack. Note that service may be denied not only because of a malicious attacker but also because of more benign sources, such as heavy network traffic.

OCSP clients may require servers to contact other servers (or CAs) to respond to a request. In the event that such contact is impossible, the server MAY reply with cached information even though the client would consider such a response to be stale. When the server responds with data it knows would not be acceptable to a client, the server MUST include one or more of the warnings described in Section 8.3.

Such warnings allow the client to be made aware of the situation, and to take appropriate steps (which will depend on the client's local policies). For example, a client that receives a disconnectedOperation warning in a response MAY notify its user of the age of the response and ask for the user's permission to proceed.

Other security considerations arise when a non-OCSP entity, such as an HTTP proxy, is used to cache OCSP responses. This sitution SHOULD be avoided. For more, see Section 9.

11. Patent Considerations

[Humour impaired: Please plant tongue firmly in cheek before proceeding.]

As of April 1998, the author is unaware of any patents in any jurisdiction that might cover any aspect of the OCSP caching protocol described in this document. It is possible that the publication of this document under the auspices of the IETF will prevent this protocol from being patented in the future.

However the author is neither omniscient nor prescient, and so implementors SHOULD take steps to ensure that they are not infringing on any patents filed in their respective jurisdictions.

Aspects external to this document, such as the basic OCSP protocol, any application of certificates, the use of a computers or a network thereof to do anything, or anything else you might think is a good idea, may already or probably someday will be subject to a patent. Implementors MAY take this as a sign of impending apocalypse.

12. References

[OCSP] M. Myers, A. Malpani, R. Ankney, C. Adams and S. Galperin, "Online Certificate Status Protocol - OCSP" (draft).

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

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

13. Author's Address

Marc Branchaud
Xcert Software Inc.
Suite 1001
701 W. Georgia Street
Vancouver, BC, Canada
V7Y 1C6

Phone: +1 604-640-6227
Fax: +1 604-640-6220
Email: marcnarc@xcert.com

14. Appendix - Collected ASN.1

```
--
-- New OCSPResponseStatus definition
--

OCSPResponseStatus ::= ENUMERATED {
    successful         (0),   -- Response has valid confirmations
    malformedRequest   (1),   -- Illegal confirmation request
    internalError      (2),   -- Internal error in issuer
    tryLater           (3),   -- Try again later
    notFound           (4),   -- Certificate not on record
    certRequired       (5),   -- Must supply certificate
    unchanged          (6)    -- Status has not changed
}

--
-- Presumed Status extension
--

presumedStatus EXTENSION ::= {
    SYNTAX PresumedStatusSyntax
    IDENTIFIED BY id-ocsp-presumedStatus
}

PresumedStatusSyntax ::= OCSPResponseStatus

--
-- Cache Warnings extension
--

cacheWarnings EXTENSION ::= {
    SYNTAX CacheWarningsSyntax
    IDENTIFIED BY id-ocsp-cacheWarnings
}

CacheWarningsSyntax ::= SEQUENCE SIZE (1..MAX) of SingleCacheWarning

SingleCacheWarning ::= SEQUENCE {
  warning      CacheWarningValue,
  text         UTF8String    OPTIONAL,
  warningData  OCTET STRING  OPTIONAL
}

CacheWarningValue ::= INTEGER {
    stale                   (0),
    revalidationFailed      (1),
    disconnectedOperation   (2)
```

```
}

--
-- Cache Status Info extension
--

cacheStatusInfo EXTENSION ::= {
    SYNTAX CacheStatusInfoSyntax
    IDENTIFIED BY id-ocsp-cacheStatusInfo
}

CacheStatusInfoSyntax ::= SEQUENCE {
     age                INTEGER,
     producedAt         GeneralTime    OPTIONAL
}

--
-- Request Cache Parameters extension
--

requestCacheParameters EXTENSION ::= {
    SYNTAX RequestCacheParametersSyntax
    IDENTIFIED BY id-ocsp-requestCacheParameters
}

RequestCacheParametersSyntax ::= SEQUENCE {
  noCache     BOOLEAN  OPTIONAL,
  maxAge      INTEGER  OPTIONAL,
  minFresh    INTEGER  OPTIONAL,
  maxStale    INTEGER  OPTIONAL,
  noValidate  BOOLEAN  OPTIONAL
}

--
-- Response Cache Parameters extension
--

responseCacheParameters EXTENSION ::= {
    SYNTAX ResponseCacheParametersSyntax
    IDENTIFIED BY id-ocsp-responseCacheParameters
}

ResponseCacheParametersSyntax ::= CHOICE {
  noCache   [0]  BOOLEAN,
  maxAge    [1]  INTEGER
}

--
```

```
   -- Collected OIDs
   --

   id-ocsp-presumedStatus OBJECT IDENTIFIER ::= TBA
   id-ocsp-cacheWarnings OBJECT IDENTIFIER ::= TBA
   id-ocsp-cacheStatusInfo OBJECT IDENTIFIER ::= TBA
   id-ocsp-requestCacheParameters OBJECT IDENTIFIER ::= TBA
   id-ocsp-responseCachParameters OBJECT IDENTIFIER ::= TBA
```