                A Protocol for Provisioning Resource Certificates

Abstract

   This document defines a framework for certificate management
   interactions between an Internet Number Resource issuer ("issuer")
   and an Internet Number Resource recipient ("subject") through the
   specification of a protocol for interaction between the two parties.
   The protocol supports the transmission of requests from the subject,
   and corresponding responses from the issuer encompassing the actions
   of certificate issuance, certificate revocation, and certificate
   status information reports. This protocol is intended to be limited
   to the application of Internet Number Resource Certificate management
   and is not intended to be used as part of a more general certificate
   management framework.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

   This document defines a framework for certificate management
   interactions between an Internet Number Resource issuer ("issuer")
   and an Internet Number Resource recipient ("subject") through the
   specification of a protocol for interaction between the two parties.
   The protocol supports the transmission of requests from the subject,
   and corresponding responses from the issuer encompassing the actions
   of certificate issuance, certificate revocation, and certificate
   status information reports. This protocol is intended to be limited
   to the application of Internet Number Resource certificate management
   and is not intended to be used as part of a more general certificate
   management framework.

1.1. Terminology

   Terms used in this document are:

   "Internet Number Resource" (or "resource") used in the context of
      this document to refer to Autonomous System (AS) numbers, IP
      version 4 addresses, and IP version 6 addresses.

   "issuer" used in the context of this document as an entity
      undertaking the role of resource issuer. An "issuer" is a
      Certification Authority (CA), and can issue resource certificates.

   "subject" used in the context of this document as an entity
      undertaking the role of resource recipient who is the subject of a
      resource certificate. A "subject" may be issued with a CA-enabled
      certificate, allowing the entity to also assume the role of an
      "issuer".

   "resource class" a resource class refers to a collection of
      resources that can be certified in a single resource certificate
      by an issuer.

   "server" in the context of this client/server protocol
      specification, the issuer assumes the role of the "server".

   "client" in the context of this client/server protocol
      specification, the subject assumes the role of the "client".

   The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT",
   "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this
   document are to be interpreted as described in [RFC2119].

2. Scope

   This Resource Public Key Infrastructure (RPKI) certificate
   provisioning protocol defines a basic set of interactions that allow
   a subject to request certificate issuance, revocation, and status
   information from the issuer, and for an issuer to maintain an issued
   certificate set that is aligned to the allocation records relating to
   each subject. The issuer's resource allocation database is the
   authoritative source of what resource allocations the issuer may
   certify for a subject.

   A resource recipient (subject) may also undertake the role of a
   resource issuer (issuer).

   This protocol specification does not encompass:

   o signing of objects with keys that are certified by resource
      certificates, nor the issuance of end-entity certificates.

   o the specification of interaction with the issuer's resource
      allocation database, nor the specification of a protocol to manage
      the publication repository.

   o the interactions between client and server that establish
      identities, or the exchange of the certificates and validation
      Public Key Infrastructure (PKI) contexts used in the Cryptographic
      Message Syntax (CMS) [RFC5652] message exchange.

   o the interactions between client and server that allow respective
      local CMS signing time values to be reset to mutually recognized
      values.

3. Protocol Specification

   This RPKI certificate provisioning protocol is expressed as a simple
   request/response interaction, where the client passes a request to
   the server, and the server generates a corresponding response.

   The protocol is implemented as an exchange of messages.

   Messages are passed over an HTTP [RFC2616] end-to-end connection. A
   message exchange commences with the client initiating an HTTP POST
   with content type of "application/rpki-updown" and the message object
   as the body. The server's response is similarly an HTTP response,
   with the message object carried as the body of the response and with
   a response content type of "application/rpki-updown". The content of
   the POST and the server's response are "well-formed" CMS [RFC5652]
   objects, encoded using the Distinguished Encoding Rules (DER) for

ASN.1 [X.509-88], formatted in accordance with the CMS profile
specified in the following section. CMS is used as the signing
format to sign the message object. The CMS message includes an end-
entity (EE) certificate that is to be used to validate the CMS
digital signature (see Section 3.1.1.4); the certificate chain that
is used to validate the EE certificate MAY be included in the CMS
message, and if it is not present it is assumed to have been
communicated between the two entities, through mechanisms not defined
in this specification.

The protocol's request/response interaction is assumed to be
reliable, in that all requests MUST generate a matching response.
The protocol requires sequential operation for each distinct client,
where the server MUST NOT accept a client's request unless it has
generated and sent a response to the client's previous request.
Attempts by the client to initiate multiple requests in parallel
(i.e., multiple concurrent requests with a common sender attribute
(see Section 3.2) in the request) MUST be detected by the server and
rejected with an error response (i.e., an error code 1101 response).

3.1. CMS Profile

   The format of the CMS object is:

          ContentInfo ::= SEQUENCE {
            contentType ContentType,
            content [0] EXPLICIT ANY DEFINED BY contentType }

          ContentType ::= OBJECT IDENTIFIER

   The content-type is the signed-data type of id-data, namely
   id-signedData, OID = 1.2.840.113549.1.7.2. [RFC5652]

3.1.1. SignedData Content Type

   According to the CMS standard [RFC5652], signed-data content types
   are the ASN.1 type SignedData:

    SignedData ::= SEQUENCE {
        version CMSVersion,
        digestAlgorithms DigestAlgorithmIdentifiers,
        encapContentInfo EncapsulatedContentInfo,
        certificates [0] IMPLICIT CertificateSet OPTIONAL,
        crls [1] IMPLICIT RevocationInfoChoices OPTIONAL,
        signerInfos SignerInfos }

      DigestAlgorithmIdentifiers ::= SET OF DigestAlgorithmIdentifier
      SignerInfos ::= SET OF SignerInfo

   Additionally, the SignerInfos set MUST contain only a single
   SignerInfo object.

3.1.1.1. version

   The version is the syntax version number. It MUST be 3,
   corresponding to the signerInfo structure having version number 3.

3.1.1.2. digestAlgorithms

   The digestAlgorithms set contains the Object Identifiers (OID)s of
   the digest algorithm(s) used in signing the encapsulated content.
   This set MUST contain exactly one digest algorithm OID, and the OID
   MUST be selected from those specified in [RFC6485].

3.1.1.3. encapContentInfo

   encapContentInfo is the signed content, consisting of a content type
   identifier and the content itself. The encapContentInfo represents
   the payload of the RPKI certificate provisioning protocol.

      EncapsulatedContentInfo ::= SEQUENCE {
         eContentType ContentType,
         eContent [0] EXPLICIT OCTET STRING OPTIONAL }

      ContentType ::= OBJECT IDENTIFIER

3.1.1.3.1. eContentType

   The eContentType for the RPKI Protocol Message object is defined as
   id-ct-xml, and has the numerical value of 1.2.840.113549.1.9.16.1.28.

      id-smime OBJECT IDENTIFIER ::= { iso(1) member-body(2) us(840)
                              rsadsi(113549) pkcs(1) pkcs9(9) 16 }

      id-ct OBJECT IDENTIFIER ::= { id-smime 1 }

      id-ct-xml OBJECT IDENTIFIER ::= { id-ct 28 }

3.1.1.3.2. eContent

   The content of an RPKI XML Protocol Object consists of a single
   protocol message, structured according to a defined XML schema, as
   defined in subsequent sections of this document. The eContent field
   of the CMS object is formally defined using ASN.1 as:

      RPKIXMLProtocolObject ::= OCTET STRING -- XML encoded message

3.1.1.4. certificates

   This field MUST be present and MUST contain the single EE certificate
   of the key pair whose private key value was used to sign the CMS.
   This MUST NOT be an RPKI certificate, and SHOULD be a certificate
   that is recognized to attest to the identity of the party that
   created the CMS object.

   This field MAY contain CA certificates that a relying party MAY use
   to validate the EE certificate.

3.1.1.5. crls

   This field MUST be present. The contents of the field are specified
   in [RFC5652]. The current Certificate Revocation List (CRL) issued
   by the same CA that issued the EE certificate of the key pair whose
   private key value was used to sign the CMS MUST be present in this
   field. This field MAY contain CRLs issued by other CAs covering CA
   certificates included in the certificates field of the CMS object
   (see Section 3.1.1.4). This field MUST NOT contain any other CRLs.

3.1.1.6. SignerInfo

   SignerInfo is defined in CMS as:

   SignerInfo ::= SEQUENCE {
     version CMSVersion,
     sid SignerIdentifier,
     digestAlgorithm DigestAlgorithmIdentifier,
     signedAttrs [0] IMPLICIT SignedAttributes OPTIONAL,
     signatureAlgorithm SignatureAlgorithmIdentifier,
     signature SignatureValue,
     unsignedAttrs [1] IMPLICIT UnsignedAttributes OPTIONAL }

3.1.1.6.1. version

   The version number MUST be 3, corresponding with the choice of
   SubjectKeyIdentifier for the sid.

3.1.1.6.2. sid

   The sid is defined as:

   SignerIdentifier ::= CHOICE {
     issuerAndSerialNumber IssuerAndSerialNumber,
     subjectKeyIdentifier [0] SubjectKeyIdentifier }

In this profile, the sid MUST be the SubjectKeyIdentifier that
appears in the EE certificate carried in the CMS certificates field.

3.1.1.6.3. digestAlgorithm

The digestAlgorithm MUST consist of the OID of a digest algorithm
that conforms to the RPKI Algorithms and Key Size Profile
specification [RFC6485].

3.1.1.6.4. signedAttrs

The signedAttrs field is defined as:

       SignedAttributes ::= SET SIZE (1..MAX) OF Attribute

       UnsignedAttributes ::= SET SIZE (1..MAX) OF Attribute

       Attribute ::= SEQUENCE {
         attrType OBJECT IDENTIFIER,
         attrValues SET OF AttributeValue }

       AttributeValue ::= ANY

The signedAttr element MUST be present and MUST include the content-
type and message-digest attributes [RFC5652]. If either the signing-
time [RFC5652] attribute or the binary-signing-time attribute
[RFC6019], or both attributes, are present, they MUST also be
included as the SignedAttributes. Other signed attributes MUST NOT
be included.

The signedAttr MUST include only a single instance of any particular
attribute. Additionally, even though the syntax allows for a SET OF
AttributeValue, in this profile, the attrValues MUST consist of only
a single AttributeValue.

3.1.1.6.4.1. Content-Type Attribute

The content-type attribute MUST be present. The attrType OID for the
content-type attribute is 1.2.840.113549.1.9.3.

       id-contentType OBJECT IDENTIFIER ::= { iso(1) member-body(2)
           us(840) rsadsi(113549) pkcs(1) pkcs9(9) 3 }

       ContentType ::= OBJECT IDENTIFIER

The attrValues for the content-type attribute MUST match the
eContentType in the EncapsulatedContentInfo. This OID value is
defined as id-ct-xml and has the numerical value of
1.2.840.113549.1.9.16.1.28.

3.1.1.6.4.2. Message-Digest Attribute

The message-digest attribute MUST be present. The attrType OID for
the message-digest attribute is 1.2.840.113549.1.9.4.

    id-messageDigest OBJECT IDENTIFIER ::= { iso(1) member-body(2)
        us(840) rsadsi(113549) pkcs(1) pkcs9(9) 4 }

    MessageDigest ::= OCTET STRING

The attrValues for the message-digest attribute contains the output
of the digest algorithm applied to the content being signed, as
specified in Section 5.4 of [RFC5652].

3.1.1.6.4.3. Signing-Time Attribute

The signing-time attribute MAY be present. The attrType OID for the
signing-time attribute is 1.2.840.113549.1.9.5.

    id-signingTime OBJECT IDENTIFIER ::= { iso(1) member-body(2)
        us(840) rsadsi(113549) pkcs(1) pkcs9(9) 5 }

    SigningTime ::= Time

    Time ::= CHOICE {
      utcTime UTCTime,
      generalizedTime GeneralizedTime }

The signing-time attribute specifies the time, based on the local
system clock, when the digital signature was applied to the content.

Guidelines regarding the use of UTCTime and GeneralizedTime in the
signing-time attribute can be found in Section 11.3 of [RFC5652].

Either one of the signing-time attribute or the binary-signing-time
attribute, or both attributes, MUST be present. If both the signing-
time and binary-signing-time attributes are present, they MUST both
represent the same underlying time value.

3.1.1.6.4.4. Binary-Signing-Time Attribute

   The binary-signing-time attribute MAY be present. The attrType OID
   for the binary-signing-time attribute is 1.2.840.113549.1.9.16.2.46.

      id-aa-binarySigningTime OBJECT IDENTIFIER ::= { iso(1)
          member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs9(9)
          smime(16) aa(2) 46 }

      BinarySigningTime ::= BinaryTime

      BinaryTime ::= INTEGER (0..MAX)

   The binary-signing-time attribute specifies the time, based on the
   local system clock, when the digital signature was applied to the
   content. The precise definition of the binary-signing-time attribute
   can be found at [RFC6019].

   Either one of the signing-time or the binary-signing-time attributes,
   or both attributes, MUST be present. If both the signing-time and
   binary-signing-time attributes are present, they MUST both represent
   the same underlying time value.

3.1.1.6.5. signatureAlgorithm Attribute

   The signatureAlgorithm MUST conform to the RPKI Algorithms and Key
   Size Profile specification [RFC6485].

3.1.1.6.6. signature Attribute

   The signature value is defined as:

      SignatureValue ::= OCTET STRING

   The signature characteristics are defined by the digest and signature
   algorithms.

3.1.1.6.7. UnsignedAttributes Attribute

   unsignedAttrs MUST be omitted.

3.1.2. CMS Object Validation

   Before a recipient of a CMS signed object can use the content of the
   object, the recipient MUST validate the signed object by verifying
   that all of the following conditions hold. A recipient may perform
   these checks in any order.

1. The CMS object is well formed, such that the signed object syntax
    complies with this specification. In particular, that each of
    the following is true:

    a. The content-type of the CMS object is SignedData (OID
        1.2.840.113549.1.7.2)

    b. The version of the SignedData object is 3.

    c. The certificates field in the SignedData object is present
        and contains one EE certificate, the SubjectKeyIdentifier
        field of which matches the sid field of the SignerInfo
        object.

    d. The crls field in the SignedData object is present.

    e. The version of the SignerInfo is 3.

    f. The signedAttrs field in the SignerInfo object is present and
        contains one each of the content-type attribute (OID
        1.2.840.113549.1.9.3), the message-digest attribute (OID
        1.2.840.113549.1.9.4), and either or both of a single
        instance of the signing-time attribute (OID
        1.2.840.113549.1.9.5) and the binary-signing-time attribute
        (OID 1.2.840.113549.1.9.16.2.46), and no other attributes.

    g. The eContentType in the EncapsulatedContentInfo is an OID
        that matches the attrValue in the content-type attribute and
        has the attrValue of id-ct-xml.

    h. The unsignedAttrs field in the SignerInfo object is omitted.

    i. If both the signing-time attribute and the binary-signing-
        time attribute are present, then their values represent the
        same time.

    j. The digestAlgorithm in the SignedData and SignerInfo objects
        conforms to the RPKI Algorithms and Key Size Profile
        specification [RFC6485].

    k. The signatureAlgorithm in the SignerInfo object conforms to
        the RPKI Algorithms and Key Size Profile specification
        [RFC6485].

    l. The signed object is DER encoded.

   2. The public key of the EE certificate (contained within the CMS
       signed-data object) can be used to successfully verify the
       signature on the signed object.

   3. The EE certificate (contained within the CMS signed-data object)
       is a valid EE certificate. In particular, there exists a valid
       certification path from a trust anchor selected by the recipient
       to this EE certificate.

   4. At the current time, the EE certificate is not revoked. This can
       be determined by confirming that the CRL contained in the crls
       field of the CMS signed data object is a current valid CRL,
       issued by the same CA that issued the EE certificate, and the CRL
       does not list the serial number of the EE certificate.

   5. The time represented by the signing-time attribute or the binary-
       signing-time attribute is greater than or equal to the time value
       passed in previously valid CMS objects that were passed from the
       same originator to this recipient. This signing time value MAY
       lie within the Validity Time of the EE certificate, but the EE
       certificate SHOULD NOT be considered invalid if this is not the
       case when all other checks listed here are passed.

3.1.3. ASN.1 Specification of the CMS Signed Object

   The following is the ASN.1 specification of the CMS signed object
   used by the RPKI provisioning protocol.

       ContentInfo ::= SEQUENCE {
         contentType ContentType,
         content [0] EXPLICIT ANY DEFINED BY contentType }

       ContentType ::= OBJECT IDENTIFIER

       id-smime OBJECT IDENTIFIER ::= { iso(1) member-body(2) us(840)
                               rsadsi(113549) pkcs(1) pkcs9(9) 16 }
       id-ct OBJECT IDENTIFIER ::= { id-smime 1 }

       id-ct-xml OBJECT IDENTIFIER ::= { id-ct 28 }

       RPKIXMLProtocolObject ::= OCTET STRING -- XML encoded message

       id-signedData OBJECT IDENTIFIER ::= { iso(1) member-body(2)
                       us(840) rsadsi(113549) pkcs(1) pkcs7(7) 2 }

```
      SignedData ::= SEQUENCE {
        version CMSVersion,
        digestAlgorithms DigestAlgorithmIdentifiers,
        encapContentInfo EncapsulatedContentInfo,
        certificates [0] IMPLICIT CertificateSet OPTIONAL,
        crls [1] IMPLICIT RevocationInfoChoices OPTIONAL,
        signerInfos SignerInfos }

      DigestAlgorithmIdentifiers ::= SET OF DigestAlgorithmIdentifier

      SignerInfos ::= SET OF SignerInfo

      SignerInfo ::= SEQUENCE {
        version CMSVersion,
        sid SignerIdentifier,
        digestAlgorithm DigestAlgorithmIdentifier,
        signedAttrs [0] IMPLICIT SignedAttributes OPTIONAL,
        signatureAlgorithm SignatureAlgorithmIdentifier,
        signature SignatureValue,
        unsignedAttrs [1] IMPLICIT UnsignedAttributes OPTIONAL }

      SignerIdentifier ::= CHOICE {
        issuerAndSerialNumber IssuerAndSerialNumber,
        subjectKeyIdentifier [0] SubjectKeyIdentifier }

      SignedAttributes ::= SET SIZE (1..MAX) OF Attribute

      Attribute ::= SEQUENCE {
      attrType OBJECT IDENTIFIER,
      attrValues SET OF AttributeValue }

      AttributeValue ::= ANY

      SignatureValue ::= OCTET STRING

      id-contentType OBJECT IDENTIFIER ::= { iso(1) member-body(2)
          us(840) rsadsi(113549) pkcs(1) pkcs9(9) 3 }

      ContentType ::= OBJECT IDENTIFIER

      id-messageDigest OBJECT IDENTIFIER ::= { iso(1) member-body(2)
          us(840) rsadsi(113549) pkcs(1) pkcs9(9) 4 }

      MessageDigest ::= OCTET STRING

      id-signingTime OBJECT IDENTIFIER ::= { iso(1) member-body(2)
          us(840) rsadsi(113549) pkcs(1) pkcs9(9) 5 }
```

```
   SigningTime ::= Time

   Time ::= CHOICE {
     utcTime UTCTime,
     generalizedTime GeneralizedTime }

   id-aa-binarySigningTime OBJECT IDENTIFIER ::= { iso(1)
       member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs9(9)
       smime(16) aa(2) 46 }

   BinarySigningTime ::= BinaryTime

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

3.2. Common Message Format

   The XML template for all messages is informally described as follows
   (the RELAX NG compact form schema that formally describes the
   protocol message objects is contained in Section 3.7):

```
   <?xml version="1.0" encoding="UTF-8"?>
   <message xmlns="http://www.apnic.net/specs/rescerts/up-down/"

           version="1"
           sender="sender name"
           recipient="recipient name"
           type="message type">

   [payload]

   </message>
```

   ----

   version:
      the value of this attribute is the version of this protocol. This
      document describes version 1.

   sender:
      the value of this attribute is the agreed name of the message
      sender, as determined between the client and the server by prior
      arrangement.

recipient:
    the value of this attribute is the agreed name of the message
    recipient, as determined between the client and the server by
    prior arrangement.

type:
    the possible values of this attribute are "list", "list_response",
    "issue", "issue_response", "revoke", "revoke_response", and
    "error_response".

Conforming parsers MUST reject any document with a version number
they do not understand or with any elements or attributes they do not
understand. Servers must generate an error response when receiving
such a request. Clients should generate an error when receiving such
a response.

The encapsulated content of the CMS wrapping is an XML document. The
remainder of this protocol specification omits this CMS wrapper and
only discusses the XML document.

Messages are checked using the following tests:

1. Check that the CMS is well-formed (see test 1 of Section 3.1.2).

2. Check that the XML is well-formed.

3. Check that the XML sender and recipient attributes reference a
    known client and this server's system respectively for a query,
    and the previously addressed server and this client for a
    response.

4. Verify the digital signature using the public key provided in the
    certificate carried in the CMS wrapper (see test 2 of Section
    3.1.2).

5. Validate the CMS-provided certificate using the PKI that has been
    determined by prior arrangement between the client and server
    (see test 3 of Section 3.1.2).

6. Check that the signing time of the CMS is equal to or greater
    than the signing time provided in the most recent previous
    message that this recipient has received from this sender (see
    test 4 of Section 3.1.2).

7. Check that the value of the version number of the message is 1.

These checks SHOULD be applied in the order specified here.

Any errors encountered while checking items 1 through 7 MUST cause a
server to generate an "HTTP 400 Bad Request" response to the HTTP
POST operation. An error in step 7 MUST cause the server to generate
a "Request-Not-Performed" error response. Any errors encountered in
these tests by a client SHOULD cause the client to generate an error.

A server MAY perform flow control on the rate of processed requests.
Requests not processed due to such a flow control constraint MAY
cause the server to generate an "HTTP 503 Service Unavailable"
response. An HTTP 503 response MAY include an HTTP Retry-After:
header as a hint to the client.

3.3. Control - Resource Class Query

This query is used for a client to query a server for a list of all
resources that have been allocated or assigned by the server to the
client. In addition, the server's response will contain a copy of
the current certificates issued by the server's CA where this client
is the certificate's subject.

3.3.1. Resource Class List Query

The value of the message "type" message attribute for this query is:

  type="list"

  Payload:

  [No message payload is defined for this query]

3.3.2. Resource Class List Response

   The value of the message "type" element for this response is:

      type="list_response"

   Payload:

    <class class_name="class name"
        cert_url="url"
        resource_set_as="as resource set"
        resource_set_ipv4="ipv4 resource set"
        resource_set_ipv6="ipv6 resource set"
        resource_set_notafter="datetime"
        suggested_sia_head="[directory uri]" >
        <certificate cert_url="url"
            req_resource_set_as="as resource set"
            req_resource_set_ipv4="ipv4 resource set"
            req_resource_set_ipv6="ipv6 resource set" >
        [certificate]
        </certificate>

        ...

        (repeated for each current certificate where the client
         is the certificate's subject)

        <issuer>[issuer's certificate]</issuer>
        </class>

     ...

     (repeated for each of the issuer's resource class where the
      client has been allocated resources)

   Where the client has been allocated resources from multiple resource
   classes, the response MUST contain multiple class elements that
   correspond to the complete set of the issuer's resource classes where
   the client holds allocated resources. Those issuer's resource
   classes where the client holds no allocated resources MUST NOT be
   included in the response.

   Where the issuer has issued multiple certificates in a resource class
   signed with different keys (as may occur during a staged issuer-key

rollover), only the most recent certificate issued with the currently
"active" issuer's key is to be listed in the response.

Each "class" element describes a set of resources that are certified
within the scope of a single certificate, referring to a single
resource class with a common validation path.

class_name:
   the value of this attribute is the issuer-assigned name of the
   issuer's resource class.

cert_url:
   in the context of a class element, the value of this attribute is
   a pointer to the issuer's CA certificate (i.e., a reference to the
   immediate superior certificate, being the CA-enabled certificate
   where the issuer is the certificate's subject). Its value is a
   comma-separated list of URIs, of which at least one MUST be an
   rsync URI [RFC5781]. Any comma values within a URI MUST be
   escaped ("%2C"). The ordering of the list may be interpreted by
   the client as a relative preference for access methods as
   expressed by the publisher of this certificate.

resource_set_as:
   in the context of a class element, the value of this attribute is
   the set of AS numbers and AS number ranges that the issuer has
   allocated to the client within the scope of this resource class,
   presented in ASCII as a comma-separated list. The list elements
   are decimal integer values and ranges of decimal integers
   specified by the lowest and highest values of the range with a
   hyphen delimiter, using the canonical order as described in
   [RFC3779], without leading zeros, and with no white space or
   punctuation other than the comma and the hyphen range designator
   (e.g., resource_set_as="123,456-789,123456"). If there are no AS
   numbers in this resource class, then the empty AS set is
   represented by a null string value ("") for this attribute.

resource_set_ipv4:
   in the context of a class element, the value of this attribute is
   the set of IPv4 addresses that the issuer has allocated to the
   client within the scope of this resource class. The value is
   presented in ASCII as a comma-separated list of elements. Each
   element is either an address prefix using the notation of <dotted
   quad>/mask length, or a range specified as the lowest and highest
   values of the range in dotted quad notation with a hyphen
   delimiter. The list is presented in canonical order, as described
   in [RFC3779]. The dotted quad notation is without leading zeros,
   and the list contains no white space or punctuation other than the
   period, forward slash, hyphen, and comma (e.g.,

resource_set_ipv4="192.0.2.0/26,192.0.2.66-192.0.2.76"). If there
are no IPv4 addresses in this resource class, the empty IPv4
address set is represented by a null string value ("") for this
attribute.

resource_set_ipv6:
   in the context of a class element, the value of this attribute is
   the set of IPv6 addresses that the issuer has allocated to the
   client within the scope of this resource class. The value is
   presented in ASCII as a comma-separated list of elements. Each
   element is either an address prefix using the notation of <hex
   nibble sequence>/mask length, or a range specified as lowest and
   highest values of the range in hex nibble notation with a hyphen
   delimiter. Trailing zero nibbles are truncated and represented by
   '::'. The list is presented in canonical order, as described in
   [RFC3779]. The hex nibble sequence notation is without leading
   zeros, and the list contains no white space or punctuation other
   than the colon, forward slash, hyphen, and comma, and conforms to
   the canonical format of [RFC5952] (e.g.,
   resource_set_ipv6="2001:db8::/48,2001:db8:2::-2001:db8:5::"). The
   XML Schema data type is
   "http://www.w3.org/TR/xmlschema-2/#hexBinary" and the value is
   case insensitive, with the canonical form being lower case. If
   there are no IPv6 addresses in this resource class, the empty IPv6
   address set is represented by a null string value ("") for this
   attribute.

resource_set_notafter:
   The value of this attribute specifies the date/time that would be
   set in the Validity notAfter field in any new certificate issued
   for this particular client within the scope of this resource
   class, should the client request a new certificate. The time
   format used for the value of this attribute is specified as
   defined in ISO 8601 [ISO.8601:2004], and MUST use UTC time
   represented as YYYY-MM-DDThh:mm:ssZ (e.g., 2007-11-29T04:40:00Z).
   If the client's certificate has a validity notAfter time that is
   different from this time, then the client SHOULD request a new
   certificate to be issued for this resource class.

suggested_sia_head: (OPTIONAL)
   If this field is present, then its value is a directory URI that
   indicates a repository publication point that the server has made
   available to the client to use for the client's collection of
   published products. This specification does not encompass the
   protocols that the client may use with the operator of the
   repository publication point in order to publish objects at this
   publication point.

[issuer's certificate]
    value is the Base64 encoding of the DER-encoded issuer's CA
    certificate (the CA-enabled certificate where the issuer is the
    certificate's subject).

    Each certificate element describes the most recently issued
    current certificate where the certificate's subject refers to the
    client for each active client key pair. A "current" certificate
    is a non-expired, non-revoked certificate. If no current
    certificate has been issued, then no certificate element is to be
    included in the response.

cert_url:
    in the context of a certificate element, this is a pointer to the
    location where the certificate issuer has published this
    certificate. This field is the issuer's suggestion for the
    Authority Information Access (AIA) field for the subject to use in
    subordinate certificates that are issued by the subject.
    According to the Resource Certificate Profile [RFC6487], the AIA
    field is a non-empty (contains a minimum of 1 element) list of
    URI's, one of which MUST be an rsync URI [RFC5781]. The order of
    URI's in the AIA field may be interpreted as the publisher's
    relative preference for access methods for this certificate. The
    cert_url conforms to this AIA specification. Its value is a
    comma-separated list of URIs, one of which MUST be an rsync URI.
    Any comma values within a URI MUST be escaped ("%2C").

req_resource_set_as:
    the set of AS numbers that were specified in the corresponding
    original certificate request that defined the maximal requested
    span of the certified AS number set, following the syntax
    described above. If this attribute was present in the certificate
    request, then the attribute MUST be present in this response;
    otherwise, it MUST NOT be present.

req_resource_set_ipv4:
    the set of IPv4 addresses that were specified in the corresponding
    original certificate request that defined the maximal requested
    span of the certified IPv4 address set, following the syntax
    described above. If this attribute was present in the certificate
    request, then the attribute MUST be present in this response;
    otherwise, it MUST NOT be present.

req_resource_set_ipv6:
    the set of IPv6 addresses that were specified in the corresponding
    original certificate request that defined the maximal requested
    span of the certified IPv6 address set, following the syntax
    described above. If this attribute was present in the certificate

      request, then the attribute MUST be present in this response;
      otherwise, it MUST NOT be present.

   [certificate]
      value is the Base64 encoding of the DER-encoded certificate.

3.4. CA - Certificate Issuance

   This query is used by the client to request the server's CA to issue
   a resource certificate for the resources that have been allocated or
   assigned to the client. If the request can be successfully
   processed, then the server's response includes the issued
   certificate.

3.4.1. Certificate Issuance Request

   The value of the message "type" element for this request is:

      type="issue"

   Payload:

   <request
           class_name="class name"
           req_resource_set_as="as resource set"
           req_resource_set_ipv4="ipv4 resource set"
           req_resource_set_ipv6="ipv6 resource set">
           [Certificate request]
            </request>

   The client MUST use different key pairs for each distinct resource
   class.

   The req_resource_set attributes are optional in the request.

   If none of the req_resource_set attributes are specified, then the
   request signifies that the complete set of all resources that match
   the client's current resource allocation is to be included in the
   issued certificate.

   If any of the req_resource_set attributes are specified in the
   request, then any missing req_resource_set attributes are to be
   interpreted as specifying the complete set of the corresponding

resource type that match the client's current resource allocation are
to be included in the issued certificate.

If the value of any included req_resource_set attributes is the null
value (""), then this indicates that no resources of that resource
type are to be included in the issued certificate.

The requested resource set values are held as a local record by the
issuer against the resource class and the client's public key. Any
subsequent Certificate Issuance Requests that specify the same
resource class and the same client's public key will (re)set the
issuer's local record of the requested resource sets to the most
recently specified values.

class_name:
   value is the server's identifier of a resource class.

req_resource_set_as: (OPTIONAL)
   the set of AS numbers that define the maximal requested span of
   the certified AS number set, formatted as per the resource_set_as
   attribute of the resource class list response.

req_resource_set_ipv4: (OPTIONAL)
   the set of IPv4 addresses that define the maximal requested span
   of the certified IPv4 address set, formatted as per the
   resource_set_ipv4 attribute of the resource class list response.

req_resource_set_ipv6: (OPTIONAL)
   the set of IPv6 addresses that define the maximal requested span
   of the certified IPv6 address set, formatted as per the
   resource_set_ipv6 attribute of the resource class list response.

[Certificate request]
   value is the certificate request. This is a Base64 encoded DER
   version of a request formatted using PKCS#10 [RFC2986]. The
   certificate request is signed using the private key part of the
   key pair whose public part is the subject key value in the
   certification request. The signing algorithm is specified in
   [RFC6485]. (This signature component is intended to demonstrate
   proof of possession of the private key.)

The response to this request is a Certificate Issuance Response if
the request can be processed online. If the request cannot be
undertaken immediately, then the server MUST respond with a "Request-
Not-Performed" message, using the appropriate error code:

o If the resource class is not defined by the server, then the
   server MUST return error code 1201.

o If the client holds no resources in a defined resource class, then
   the server MUST return error code 1202 and not proceed with the
   request.

o If the certificate request payload is badly formed, then the
   server MUST return error code 1203.

o If the public key used in the certificate request implies that the
   client is attempting to use identical key pairs for multiple
   resource classes, then the server MUST respond with a 1204 error
   code.

o If the certificate issuer uses an off-line process to undertake
   certificate issuance, and the server cannot directly respond to
   the certificate issuance request with an issued certificate, then
   the certificate issuer MUST respond to the first instance of this
   request with an error code 1104 to indicate that the request is
   being processed asynchronously. Subsequent repetitions of this
   request while the off-line actions are being undertaken SHOULD
   cause a response with error code 1101. In this context, where
   off-line processes are invoked for certificate issuance, if the
   certificate issuer determines in processing the request that the
   issued certificate would be identical in all respects to the most
   recently issued certificate for this client, other than the
   certificate's serial number, were the certificate to be issued,
   the issuer may choose to respond with the most recently issued
   certificate and not initiate an off-line certificate issuance
   request.

Note that a client, when receiving a 1104 response to a certificate
issuance request, MAY periodically resubmit the request, in which
case the client MUST receive an error code 1101 response while the
request is being processed, and a Certificate Issuance Response when
the certificate issuance process has completed. In such
circumstances, a client SHOULD limit the frequency of such repeated
requests to no more than 1 request in each 24-hour interval.

3.4.2. Certificate Issuance Response

   The value of the message "type" element for this response is:

      type="issue_response"

      Payload:

       <class class_name="class name"
              cert_url="url"
              resource_set_as="as resource set"
              resource_set_ipv4="ipv4 resource set"
              resource_set_ipv6="ipv6 resource set" >
               <certificate cert_url="url"
                     req_resource_set_as="as resource set"
                     req_resource_set_ipv4="ipv4 resource set"
                     req_resource_set_ipv6="ipv6 resource set" >
               [certificate]
               </certificate>
               <issuer>[issuer's certificate]</issuer>
             </class>

   If the certificate issuer determines that the issued certificate
   would be identical in all respects to the most recently issued
   certificate for this client, other than the certificate's serial
   number, were the certificate to be issued, the issuer may choose to
   respond with the most recently issued certificate and not issue a new
   certificate for this request.

   The definition of the attributes and syntax of the values is the same
   as the resource class list response, but the response only references
   the (single) named resource class, and the (single) certificate
   issued against the client's public key as provided in the
   corresponding certificate request.

3.5. Certificate Revocation

   This request 'retires' a client's key pair by requesting that the
   server's CA revoke all certificates for this client (i.e., where this
   client is the subject) that contain the matching public key, within
   the scope of a named resource class. Individual certificates cannot
   be revoked within the scope of this protocol.

3.5.1. Certificate Revocation Request

   The value of the message "type" element for this request is:

      type="revoke"

      Payload:

      <key class_name="class name"
           ski="[encoded hash of the subject public key]" />

   This command directs the server's CA to immediately mark all issued
   valid certificates issued by this issuer within the named resource
   class with this client's subject name and the provided SKI value to
   be marked as revoked, causing the issued certificates to be withdrawn
   from the publication repository and to be listed in the server's
   subsequent CRLs within this resource class. The issuer MUST ensure
   that all certificates to be revoked were issued with the requesting
   client as the certificate's subject.

   class_name:
      value is the issuer-assigned name of the issuer's resource class.

      ski:
      value is the encoded hash of the client's public key that is to be
      revoked. The algorithm for the encoding is to generate the
      160-bit SHA-1 hash of the client's public key, as defined in
      method (1) of Section 4.2.1.2 of [RFC5280], and encode this value
      using the Base 64 encoding with URL and Filename Safe Alphabet, as
      defined in Section 5 of [RFC4648].

3.5.2. Certificate Revocation Response

   The value of the message "type" element for this response is:

      type="revoke_response"

      Payload:

      <key class_name="class name"
        ski="[encoded hash of the subject public key]" />

      ----

   class_name:
      value is the issuer-assigned name of the server's resource class.

   ski:
      value is the encoded hash of the client's public key that is to be
      revoked. The algorithm for the encoding is to generate the
      160-bit SHA-1 hash of the client's public key, as defined in
      method (1) of Section 4.2.1.2 of [RFC5280], and encode this value
      using the Base 64 encoding with URL and Filename Safe Alphabet, as
      defined in Section 5 of [RFC4648].

3.6. Request-Not-Performed Response

   The value of the message "type" element for this response is:

      type="error_response"

      Payload:

      <status>[Code]</status>
      <description xml:lang="en-US">[Readable text]</description>

   All states where an error response if to be generated, either due to
   detected errors or inconsistencies in the content of the request or
   server-side states that prevent the request being performed, generate
   a Request-Not-Performed response.

description:
    value is a text field. This element MAY be present. It's value
    has no defined meaning within the scope of this protocol, and
    implementations may assume that some form of human-readable text
    may be used here. If the HTTP request that triggered this error
    response includes an Accept-Language header as defined in Section
    14.4 of the HTTP/1.1 specification [RFC2616], then the server MAY
    include a second description element using the highest ranked
    preferred language of the client. The en-US description MUST
    always be included if the element is present.

    The error code set is:

        Code Value   Description
        1101   already processing request
        1102   version number error
        1103   unrecognized request type
        1104   request scheduled for processing
        1201   request - no such resource class
        1202   request - no resources allocated in resource class
        1203   request - badly formed certificate request
        1204   request - already used key in request
        1301   revoke - no such resource class
        1302   revoke - no such key
        2001   Internal Server Error - Request not performed

3.7. XML Schema

    The following is a RELAX NG compact form schema describing version 1
    of this protocol.

        Note: As discussed in [XML], "the namespace name, to serve its
        intended purpose, SHOULD have the characteristics of uniqueness
        and persistence. It is not a goal that it be directly usable for
        retrieval of a schema (if any exists)".

    default namespace = "http://www.apnic.net/specs/rescerts/up-down/"

    grammar {
        resource_set_as = xsd:string { maxLength="512000"
                                        pattern="[\-,0-9]*" }
        resource_set_ip4 = xsd:string { maxLength="512000"
                                        pattern="[\-,/.0-9]*" }
        resource_set_ip6 = xsd:string { maxLength="512000"
                                        pattern="[\-,/:0-9a-fA-F]*" }

        class_name = xsd:token { minLength="1" maxLength="1024" }
        ski = xsd:token { minLength="27" maxLength="1024" }

```
      label = xsd:token { minLength="1" maxLength="1024" }
      cert_url = xsd:string { minLength="10" maxLength="4096" }
      base64_binary = xsd:base64Binary { minLength="4"
                                         maxLength="512000" }

      start = element message {
        attribute version { xsd:positiveInteger {
                                            maxInclusive="1" } },
        attribute sender { label },
        attribute recipient { label },
        payload
      }

      payload |= attribute type { "list" }, list_request
      payload |= attribute type { "list_response"}, list_response
      payload |= attribute type { "issue" }, issue_request
      payload |= attribute type { "issue_response"}, issue_response
      payload |= attribute type { "revoke" }, revoke_request
      payload |= attribute type { "revoke_response"}, revoke_response
      payload |= attribute type { "error_response"}, error_response

      list_request = empty
      list_response = class*

      class = element class {
        attribute class_name { class_name },
        attribute cert_url { cert_url },
        attribute resource_set_as { resource_set_as },
        attribute resource_set_ipv4 { resource_set_ip4 },
        attribute resource_set_ipv6 { resource_set_ip6 },
        attribute resource_set_notafter { xsd:dateTime },
        attribute suggested_sia_head { xsd:anyURI { maxLength="1024"
                                           pattern="rsync://.+"} }?,
        element certificate {
          attribute cert_url { cert_url },
          attribute req_resource_set_as { resource_set_as }?,
          attribute req_resource_set_ipv4 { resource_set_ip4 }?,
          attribute req_resource_set_ipv6 { resource_set_ip6 }?,
          base64_binary
        }*,
        element issuer { base64_binary }
      }

      issue_request = element request {
        attribute class_name { class_name },
        attribute req_resource_set_as { resource_set_as }?,
        attribute req_resource_set_ipv4 { resource_set_ip4 }?,
        attribute req_resource_set_ipv6 { resource_set_ip6 }?,
```

```
      base64_binary
    }
    issue_response = class

    revoke_request = revocation
    revoke_response = revocation

    revocation = element key {
      attribute class_name { class_name },
      attribute ski { ski }
    }

    error_response =
      element status { xsd:positiveInteger { maxInclusive="9999" } },
      element description { attribute xml:lang { xsd:language },
                                   xsd:string { maxLength="1024" } }*
    }
```

4. Security Considerations

   This protocol supports the maintenance of resource certificates that
   the issuer issues for a subject in certifying resources that have
   been allocated or assigned by the issuer to the subject [RFC6480].
   This protocol assumes that the issuer and subject are known to each
   other and have exchanged credentials so as to support the mutual
   recognition of the digital signatures used to sign the CMS messages.
   The mechanisms used to perform the associated credential exchange are
   not described in this specification.

   The protocol is a minimal query/response protocol that imposes strict
   serialization on each query/response transaction, reducing the
   potential for the subject and the issuer to lose synchronization over
   the issued certificate state.

   Validation of protocol objects (Section 3.1.2) requires that the CMS
   signing-time value be greater than or equal to the time value passed
   in the previously valid protocol objects that were passed from the
   same originator to the same recipient. If a party inadvertently
   sends a valid message (protocol object) with a signing time in the
   future, then subsequent messages from the party in the same
   client/server context can use signing-time value consistent with this
   validation constraint, such that the signing times contained in
   subsequent messages are greater than or equal to the signing-time
   value of the previous valid message. (Note that it is not a
   normative requirement that the signing time be precisely aligned to a
   time of day clock, thus permitting arbitrarily large clock skew
   values in the context of this protocol message exchange.) If the
   client and server wish to reset the signing time to a mutually agreed

value, then, (as noted in Section 2) the interactions between the
client and the server to achieve this outcome are not encompassed in
this protocol.

5. IANA Considerations

   IANA has registered the following media type:

   application/rpki-updown

5.1. application/rpki-updown

   Type name: application
   Subtype name: rpki-updown
   Required parameters: None
   Optional parameters: None
   Encoding considerations: binary
   Security considerations: Carries an RPKI Provisioning Protocol
      Message, as defined in this document.
   Interoperability considerations: None
   Published specification: This document
   Applications that use this media type: HTTP [RFC5652]
   Additional information:
      Magic number(s): None
      File extension(s):
      Macintosh File Type Code(s):
   Person & email address to contact for further information:
      Geoff Huston <gih@apnic.net>
   Intended usage: COMMON
   Restrictions on usage: Only to be used as an RPKI Provisioning
      Protocol message object type, as defined in this document.
   Author: Geoff Huston <gih@apnic.net>
   Change controller: Geoff Huston <gih@apnic.net>

6. Acknowledgements

   The authors would like to acknowledge the valued contributions from
   Russ Housley, Steve Kent, Randy Bush, George Michaelson, Robert
   Kisteleki, Tim Bruijnzeels, and Carsten Bormann in the preparation of
   the protocol described in this document.

7. References

7.1. Normative References

   [ISO.8601:2004]
              ISO, "ISO 8601:2004 Representation of dates and Times",
              2004.

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

   [RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H.,
              Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext
              Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

   [RFC2986] Nystrom, M. and B. Kaliski, "PKCS #10: Certification
              Request Syntax Specification Version 1.7", RFC 2986,
              November 2000.

   [RFC3779] Lynn, C., Kent, S., and K. Seo, "X.509 Extensions for IP
              Addresses and AS Identifiers", RFC 3779, June 2004.

   [RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data
              Encodings", RFC 4648, October 2006.

   [RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S.,
              Housley, R., and W. Polk, "Internet X.509 Public Key
              Infrastructure Certificate and Certificate Revocation List
              (CRL) Profile", RFC 5280, May 2008.

   [RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70,
              RFC 5652, September 2009.

   [RFC5781] Weiler, S., Ward, D., and R. Housley, "The rsync URI
              Scheme", RFC 5781, February 2010.

   [RFC5952] Kawamura, S. and M. Kawashima, "A Recommendation for IPv6
              Address Text Representation", RFC 5952, August 2010.

   [RFC6019] Housley, R., "BinaryTime: An Alternate Format for
              Representing Date and Time in ASN.1", RFC 6019, September
              2010.

   [RFC6485] Huston, G., "The Profile for Algorithms and Key Sizes for
              Use in the Resource Public Key Infrastructure (RPKI)", RFC
              6485, February 2012.

      [X.509-88] CCITT, "Recommendation X.509: The Directory-
                 Authentication Framework", 1988.

7.2. Informative References

      [RFC6480] Lepinski, M. and S. Kent, "An Infrastructure to Support
                 Secure Internet Routing", RFC 6480, February 2012.

      [RFC6487] Huston, G., Michaelson, G., and R. Loomans, "A Profile for
                 X.509 PKIX Resource Certificates", RFC 6487, February
                 2012.

      [XML]   Bray, T., Hollander, D., Layman, A., Tobin, R., and H.
                 Thompson, "Namespaces in XML 1.0 (Third Edition)", World
                 Wide Web Consortium Recommendation REC-xml-names-20091208,
                 December 2009, <http://www.w3.org/TR/2009/REC-xml-
                 names-20091208/>.

Authors' Addresses

   Geoff Huston
   APNIC

   EMail: gih@apnic.net
   URI: http://www.apnic.net

   Robert Loomans
   APNIC

   EMail: robertl@apnic.net
   URI: http://www.apnic.net

   Byron Ellacott
   APNIC

   EMail: bje@apnic.net
   URI: http://www.apnic.net

   Rob Austein
   Internet Systems Consortium

   EMail: sra@hactrn.net