A Uniform Resource Name (URN) Namespace for Certificates
draft-seantek-certspec-01

Abstract

Digital certificates are used in many systems and protocols to identify and authenticate parties. This document describes a Uniform Resource Name (URN) namespace that identifies certificates. These URNs can be used when certificates need to be identified by value or reference.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

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

This Internet-Draft will expire on April 25, 2014.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.
1. Introduction

Digital certificates are used in many systems and protocols to identify and authenticate parties. Security considerations frequently require that the certificate must be identified with certainty, because selecting the wrong certificate will lead to validation errors (resulting in denial of service), or in improper credential selection (resulting in unwanted disclosure or substitution attacks). The goal of this namespace is to provide a uniform syntax for identifying certificates with precision in Uniform Resource Identifiers (URIs), specifically Uniform Resource Names (URNs).

Using this syntax, any protocol or system that refers to a certificate in a textual format can unambiguously identify that certificate by value or reference. Implementers that parse these URNs can resolve them into actual certificates.

Examples:
urn:cert:SHA-1:3ea3f070773971539b9dbf1b98c54be3a4f0f3c8
urn:cert:issuersn:cn=AcmeIssuingCompany, st=California, c=US;0134F1
urn:cert:base64:MIICAS...

1.1. Requirements Language

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 RFC 2119 [RFC2119].

2. Motivation and Purpose

Although certificates have diverse applications, there has been traditionally no way to refer to a certificate uniformly and unambiguously by reference in text. Certificates that identify long public keys (e.g., 2048-bit RSA keys) and that contain required and recommended PKIX extensions can easily exceed many kilobytes in length, which is impractical for certain applications.

The purpose of this specification is to provide a uniform textual format for identifying individual certificates. When a resolver resolves certspec, the resolver’s output is either a single certificate or nothing. This specification is not designed or intended to provide a search tool or query language to match multiple certificates.

2.1. Static Identification
Identifying a specific certificate by reference or value allows diverse applications to have a common syntax. For example, applications can store certspecs as local or shared preferences, so that users can edit them without resorting to application-specific storage formats or relying on the availability of particular protocols represented by URLs (such as http:, ldap:, file:, or ni: schemes). When conveyed in protocol, a certspec can identify a specific certificate to a client or server using text-based formats such as YAML, XML, JSON, and others. The format described in this document is intended to be readily reproducible by users using common certificate processing tools, so that users can easily create, recognize, compare, and reproduce them at a glance. For example, the hash-based identifications use hexadecimal encoding so that a user can easily compose or compare an URN with a simple copy-and-paste operation.

2.2. Resolution to Context-Appropriate Schemes

When the certificate represented by a certspec needs to be resolved, an application can resort to any number of schemes. For example, when the certificate is identified by hash, the application can resolve the cert: URN to a Named Information (ni:) URI [RFC6920] for further processing. When the certificate is identified by issuer and serial number, the application can resolve the cert: URN to an LDAP service (for example, ldap:///cn=ExampleCA,o=ExampleCo,st=California,c=US ).

3. certspec Syntax

The Namespace Specific String (NSS) portion of a certspec has the following ABNF specification:

NSS = spec-type "::" spec-value ( '?' certattrs)
  spec-type = scheme
  certattrs = <URN chars>
  hexOctet = hexDigit hexDigit
  hexDigit =
    "0" / "1" / "2" / "3" / "4" / "5" / "6" / "7" / "8" / "9" / "a" / "b" / "c" / "d" / "e" / "f" / "A" / "B" / "C" / "D" / "E" / "F"
3.1. spec-type and spec-value

The spec-type identifies the certificate specification type. The acceptable characters for spec-type are the same as an URI scheme name [RFC3986 sec. 3.1]. spec-types are compared case-insensitively. The spec-value identifies the certificate specification value. The acceptable characters for spec-value depend on the spec-type.

3.2. certattrs

A certspec can include attributes that are associated with the identified certificate. These attributes do NOT affect certificate identification; the syntax is intended primarily to convey certificate metadata such as attributes found in PKCS #9, PKCS #11, PKCS #12, and particular implementations of cryptographic libraries. This Internet-Draft does not further define certattrs; the characters of certattrs can be any valid URN character from [RFC2141] (effectively, any URI character from [RFC3986] except [ and ]).

4. Certificate Specifications

A certificate specification (certspec) unambiguously identifies a single certificate. However, multiple certsps may exist for a particular certificate. This Internet-Draft provides five cryptographic hash-based specs, two value-based specs, and two data-based specs.

4.1. Cryptographic Hash-Based Specifications

A cryptographic hash of a certificate uniquely identifies that certificate. Such a hash may also be called a "certificate fingerprint". In all certspecs in this specification *or* derived from this specification, the hash is computed over the octets of the DER encoding of the certificate, namely, the Certificate type of [RFC5280 sec. 4.1]. The DER encoding includes tag and length octets, so it always starts with 30h (the tag for SEQUENCE).

In certspecs in this specification, the spec-value is the hexadecimal encoding of the hash value octets. For example, a 256-bit SHA-256 hash is represented by exactly 32 hex octets, or 64 hex characters. The following ABNF defines proper spec-values:

```
spec-value-sha-1   = 20hexOctet
spec-value-sha-256 = 32hexOctet
spec-value-SHA-384 = 48hexOctet
spec-value-SHA-512 = 64hexOctet
```
Lexical equivalence of two certspecs that have the same spec-type SHALL be determined by a case-insensitive comparison of spec-values, or by converting the hexadecimal spec-values to octets and comparing exact equivalence of the octets. A conforming implementation MUST reject values that contain non-hex digits, such as spaces, tabs, hyphens, or anything else.

Conforming implementations to this Internet-Draft MUST process these hash-based certspecs, unless security considerations dictate otherwise. Acceptable reasons for refusing to process a certspec include a) the local policy prohibits use of the hash, or b) the hash has known cryptographic weaknesses, such as a preimage attacks, which weaken the cryptographic uniqueness guarantees of the hash.

4.1.1. SHA-1

The spec-type is "SHA-1". The hash is computed using SHA-1 [SHS].

4.1.2. SHA-256

The spec-type is "SHA-256". The hash is computed using SHA-256 [SHS].

4.1.3. SHA-384

The spec-type is "SHA-384". The hash is computed using SHA-384 [SHS].

4.1.4. SHA-512

The spec-type is "SHA-512". The hash is computed using SHA-512 [SHS].

4.2. Value-Based Specifications

A certificate may be identified reflexively by its constituent octets. For small-to-medium certificates, identifying the certificate by embedding it in the certspec will be computationally efficient and resistant to denial-of-service attacks (by being always available). A conforming implementation MUST implement base64 and hex specs.

The octets of a certificate are the octets of the DER encoding of the certificate, namely, the Certificate type of [RFC 5280 sec. 4.1].
Lexical equivalence of two certspecs that are value-based SHALL be determined by converting the spec-value to certificate octets, and comparing the octets for strict equivalence. Accordingly, it is possible that base64 and hex certspecs are lexically equivalent.

4.2.1. base64

The spec-type is "base64". The spec-value is the base64 encoding [RFC4648 sec. 4] of the certificate octets. Like the data: URL [RFC2397], the characters '+' and '/' refer to values 62 and 63, respectively. Additionally, if the length of certificate octets is not a multiple of 3, it is expected that one or two trailing equal signs '=' will be present.

'+' and '/' have no reserved meaning in this spec-type. While the URN syntax rules [RFC2141] state that '/' should not be used in unencoded form, in this specification, '/' MAY be present in unencoded form in the base64 spec-type. In any case, a conforming implementation MUST be able to process "%"-encoded characters.

While a strict implementation would reject non-base64 characters, a lenient implementation MAY ignore non-base64 characters, such as CR, LF, whitespace, or the absence of trailing '='. As a result, two certspecs that have the same base64-encoded data but different stray non-base64 characters MAY be judged lexically equivalent. Similarly, [RFC2141] requires that non-reserved characters (in this case, alphanumerics) must not be "%"-encoded, but a lenient implementation MAY decode these "%"-encoded characters anyway. This specification neither recommends nor discourages such leniency, but implementors should weigh the benefits and risks as discussed further in the Security Considerations section [TODO: link].

4.2.2. hex

The spec-type is "hex". The spec-value is the hexadecimal encoding [RFC4648 sec. 8] of the certificate octets. Whether an implementation should process "%"-encoded characters is subject to the same considerations as the equivalent characters in the base64 spec.

4.2.3. data (Reserved)

The spec-type is "data". This type SHOULD NOT be used for interchange. This document reserves this spec-type for future use.

4.3. Data-Based Specifications
A certificate may be identified by data contained within it. The following specs reflect the traditional reliance of PKIX [RFC5280] and CMS [RFC5652] on a certificate’s issuer distinguished name and serial number, or a certificate’s subject key identifier. These specs provide textual representations for these identifiers.

4.3.1. issuersn: Issuer Name and Serial Number

The spec-type is "issuersn". The spec-value is given by the following ABNF:

spec-value-issuersn = distinguishedName SEMI serialNumber
serialNumber = 1*hexOctet

<distinguishedName> is defined in [RFC4514], and <SEMI> is defined in RFC4512. Note that RFC 4514 no longer separates relative distinguished names (RDNs) by semicolons, as required by its predecessor, [RFC2253]. Accordingly, ‘;’ is used to separate the issuer’s DN from the subject’s serial number.

Care should be taken in escaping and "%-encoding the relevant characters. In particular: "?" is permitted in a distinguishedName, but is RESERVED by this specification and [RFC2141]. Any question marks in distinguished names MUST be "%"-encoded when placed in the spec-value. "#" is used as a token at the beginning of the hexstring production for attributeValue data, but is RESERVED by [RFC2141]. The updated draft draft-ietf-urnbis-rfc2141bis-urn clarifies that "#" denotes the beginning of the fragment identifier component, and therefore can be used after (not within) the Namespace Specific String. Any "#" characters in distinguished names MUST be "%"-encoded when placed in the spec-value.

<serialNumber> is the hexadecimal encoding of the certificate’s serial number, with the exact same (DER encoded) contents octets of a CertificateSerialNumber ::= INTEGER as specified in [RFC5280] sec. 4.1. If the serial number hex octets are malformed, the certspec is invalid.

A conforming implementation SHOULD implement this issuersn spec. If the implementation implements it, the implementation MUST process serial numbers up to the same length as required by [RFC5280] sec. 4.1.2.2 (20 octets), and MUST process distinguished name strings as required by [RFC4514], including the table of minimum AttributeType name strings that MUST be recognized.

Lexical equivalence of two issuersn certspecs SHALL be determined by comparing the integer values of the serialNumbers for exact
equivalence, and comparing the distinguished names for a match. Distinguished names match if they satisfy the name matching requirements of \[\text{RFC5280}\].

4.3.2. ski: Subject Key Identifier

The spec-type is "ski". The spec-value is given by the following ABNF:

\[
\text{spec-value-ski} = \text{keyIdentifier}
\]

\[
\text{keyIdentifier} = 1^{*}\text{hexOctet}
\]

\(<\text{keyIdentifier}>\) is the hexadecimal encoding of the certificate’s subject key identifier, which is recorded in the certificate’s Subject Key Identifier extension \[\text{RFC5280}\] sec. 4.2.1.2. A certificate that lacks a subject key identifier cannot be identified using this spec.

Lexical equivalence of two ski certspecs SHALL be determined by converting the hexadecimal spec-values to octets and comparing the exact equivalence of the octets.

A conforming implementation MAY implement this ski spec.

4.3.3. dbkey (Reserved)

The spec-type is "dbkey". This type SHOULD NOT be used for interchange. This document reserves this spec-type for future use.

5. Registration Template

Namespace ID:

cert

Registration Information:

Version: 1
Date: 2013-10-21

Declared registrant of the namespace:

IETF

Declaration of syntactic structures:

The structure of the Namespace Specific String is provided above.

Relevant ancillary documentation:

Certificates are defined by \[\text{RFC5280}\] and \[X.509]\.
Identifier uniqueness considerations:
The spec type is assigned by IANA through the IETF consensus process, so this process guarantees uniqueness of these identifiers. The uniqueness of the spec value depends on the spec type. For specs that identify cryptographic hashes, the cryptographic hash algorithm itself guarantees uniqueness. For specs that identify certificates by value, the inclusion of the certificate in the URN itself guarantees uniqueness. For specs that identify certificates by certificate data, the resolver’s database of certificates and implementation of certification path validation [RFC5280 sec. 6] ensure uniqueness.

Identifier persistence considerations:
A certificate is a permanent digital artifact, irrespective of its origin. As the assignment process records mathematical or existential facts about the certificate, such as one of its cryptographic hashes, the binding between the URN and the certificate resource is permanent. Changing even one bit of the certificate will alter its URN, will make the certificate unresolvable, or both.

Process of identifiers assignment:
Generating a certspec (cert URN) does not require that a registration authority be contacted.

Process for identifier resolution:
This Internet Draft does not specify a resolution service for certspecs. However, resolving certificate references to actual certificates is a common practice with a wide number of offline and online implementations.

Rules for Lexical Equivalence:
Certspecs (cert URNs) are lexically equivalent if they both have the same spec type (compared case-insensitively) and the same space value, and therefore impliedly point to the same certificate. Comparison of spec values depends on the rules of the spec. Although extensions may be appended to a certspec, these extensions are guaranteed not to affect lexical equivalence.

Certspecs are semantically equivalent if they both resolve to the same certificate.

Conformance with URN Syntax:
The character ‘?’ is reserved for future extensions to this
specification. The URN of this namespace conforms to URN Syntax [RFC2141] and Uniform Resource Identifier (URI): Generic Syntax [RFC3986].

Validation mechanism:
Each spec defines the validation mechanism for its respective value. It may be appreciated that validation of the URN is a completely different process from the Certification Path Validation Algorithm [RFC5280 sec. 6], which determines whether the *certificate* is valid.

Scope:
Global.

6. Use of certspec outside URN

certspec is useful wherever a system may need to refer to a certificate by value or by reference. Some implementations may wish to refer to a certificate without enabling all of the expressive power (and security considerations) of URIs. Accordingly, this section provides a uniform method for using a certspec outside of a URN. Examples:

Examples:
SHA-1:aaaaaaaaaaa
WHIRLPOOL:aaaaaaaaaaaaaa
base64:MIICAS...

To use certspec outside of a URI (URN) context, simply omit the prefix "urn:cert:". All other lexical rules remain in effect, including "%"-encoding. Care should be taken to process '?' in particular, since '?' separates the certspec from appended attributes. A conforming implementation of raw certspecs MUST permit the prefix "urn:cert:" in addition to the raw certspec, which starts with the spec-type. This specification guarantees that the the cert-type "urn" is RESERVED and will never be used. However, implementors must take note that a raw certspec is not a valid URI, because cert-types are not registered URI schemes and do not have the same semantics as a URI.

7. IANA Considerations

This document requests the assignment of formal URN namespace ID "cert".
This document requests the creation of a registry to record specs. New specs shall be ratified by the IETF consensus process.

8. Security Considerations

Digital certificates are important building blocks for authentication, integrity, authorization, and (occasionally) confidentiality services. Accordingly, identifying digital certificates incorrectly can have significant security ramifications.

When using specs that are cryptographic hashes (fingerprints), the cryptographic hash algorithm MUST be implemented properly and SHOULD have no known attack vectors. For this reason, algorithms that are considered "broken" as of the date of this Internet-Draft, such as MD5 [MD5], are precluded from being valid certspecs. The registration of a particular algorithm spec in this namespace does NOT mean that it is acceptable or safe for every usage, even though this Internet-Draft requires that a conforming implementation MUST implement certain specs.

When using by-value specs, the implementation MUST be prepared to process URNs of arbitrary length. As of this writing, useful certificates rarely exceed 10KB, and most implementations are concerned with keeping certificate sizes down rather than up [CITE: Google SSL overclocking, etc.]. However, a pathological or malicious certificate could easily exceed these metrics. If an URN resolver cannot process a URN’s full length, it MUST reject the certspec.

When using specs that depend on certificate data, the implementation MUST be prepared to deal with multiple found certificates that contain the same certificate data, but are not the same certificate. In such a case, the implementation MUST segregate these certificates so that it only resolves the URN to certificates that it considers valid or trustworthy (as discussed further below). If, despite this segregation, multiple valid or trustworthy certificates match the certspec, the certspec MUST be rejected, because a certspec is meant to identify exactly one certificate (not a family of certificates).

Apart from the mechanics of certspecs (cert URNs), certificates should not be used unless they have passed the Certification Path Validation Algorithm [RFC5280 sec. 6], or another algorithm that provides some guarantee of validity. For example, if a certificate database contains a set of certificates that it considers inherently trustworthy, then the inclusion of a certificate in that set makes it trustworthy, regardless of the results of the Certification Path Validation Algorithm. Such a database is frequently used for "Root CA" lists.
9. References

9.1. Normative References


9.2. Informative References


Author’s Address

Sean Leonard
Penango, Inc.
11400 West Olympic Boulevard
Suite 1500
Los Angeles, CA  90064
USA

Email: dev+ietf@seantek.com
URI:  http://www.penango.com/