Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 16, 2015

J. Pechanec
D. Moffat
Oracle Corporation

October 13, 2014

The PKCS#11 URI Scheme
draft-pechanec-pkcs11uri-16

Abstract

This memo specifies a PKCS#11 Uniform Resource Identifier (URI) Scheme for identifying PKCS#11 objects stored in PKCS#11 tokens, for identifying PKCS#11 tokens themselves, or for identifying PKCS#11 libraries. The URI is based on how PKCS#11 objects, tokens, and libraries are identified in the PKCS#11 Cryptographic Token Interface Standard.

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 16, 2015.

Copyright Notice

Table of Contents

1. Introduction

The PKCS #11: Cryptographic Token Interface Standard [pkcs11_spec] specifies an API, called Cryptoki, for devices which hold cryptographic information and perform cryptographic functions. Cryptoki, pronounced crypto-key and short for cryptographic token interface, follows a simple object-based approach, addressing the goals of technology independence (any kind of device may be used) and resource sharing (multiple applications may access multiple devices), presenting applications with a common, logical view of the device - a cryptographic token.

It is desirable for applications or libraries that work with PKCS#11 tokens to accept a common identifier that consumers could use to identify an existing PKCS#11 storage object in a PKCS#11 token, an existing token itself, or an existing Cryptoki library (also called a producer, module, or provider). The set of storage object types that can be stored in a PKCS#11 token includes a certificate, a public, private or secret key, and a data object. These objects can be uniquely identifiable via the PKCS#11 URI scheme defined in this document. The set of attributes describing a storage object can contain an object label, its type, and its ID. The set of attributes that identifies a PKCS#11 token can contain a token label, a manufacturer name, a serial number, and a token model. Attributes that can identify a Cryptoki library are a library manufacturer, a library description, and a library version. Library attributes may

be necessary to use if more than one Cryptoki library provides a token and/or PKCS#11 objects of the same name. A set of query attributes is provided as well.

The PKCS#11 URI cannot identify other objects defined in the specification [pkcs11_spec] aside from storage objects. For example, objects not identifiable by a PKCS#11 URI include a hardware feature and mechanism. Note that a Cryptoki library does not have to provide for storage objects at all. The URI can still be used to identify a specific PKCS#11 token or an API producer in such a case.

A subset of existing PKCS#11 structure members and object attributes was chosen believed to be sufficient in uniquely identifying a PKCS#11 token, storage object, or library in a configuration file, on a command line, or in a configuration property of something else. Should there be a need for a more complex information exchange on PKCS#11 entities a different means of data marshalling should be chosen accordingly.

A PKCS#11 URI is not intended to be used to create new PKCS#11 objects in tokens, or to create PKCS#11 tokens. It is solely to be used to identify and work with existing storage objects and tokens through the PKCS#11 API, or identify Cryptoki libraries themselves.

The URI scheme defined in this document is designed specifically with a mapping to the PKCS#11 API in mind. The URI uses the scheme, path and query components defined in the Uniform Resource Identifier (URI): Generic Syntax [RFC3986] document. The URI does not use the hierarchical element for a naming authority in the path since the authority part could not be mapped to PKCS#11 API elements. The URI does not use the fragment component.

If an application has no access to a producer or producers of the PKCS#11 API the query component module attributes can be used. However, the PKCS#11 URI consumer can always decide to provide its own adequate user interface to locate and load PKCS#11 API producers.

2. Contributors

Stef Walter, Nikos Mavrogiannopoulos, Nico Williams, Dan Winship, and Jaroslav Imrich contributed to the development of this document.

3. PKCS#11 URI Scheme Definition

In accordance with [RFC4395], this section provides the information required to register the PKCS#11 URI scheme.

3.1. PKCS#11 URI Scheme Name

pkcs11

3.2. PKCS#11 URI Scheme Status

Permanent.

3.3. PKCS#11 URI Scheme Syntax

The PKCS#11 URI is a sequence of attribute value pairs separated by a semicolon that form a one level path component, optionally followed by a query. In accordance with Section 2.5 of [RFC3986], the data should first be encoded as octets according to the UTF-8 character encoding [RFC3629]; then only those octets that do not correspond to characters in the unreserved set or to permitted characters from the reserved set should be percent-encoded. This specification suggests one allowable exception to that rule for the "id" attribute, as stated later in this section. Grammar rules "unreserved" and "pct-encoded" in the PKCS#11 URI specification below are imported from [RFC3986]. As a special case, note that according to Appendix A of [RFC3986], a space must be percent-encoded.

PKCS#11 specification imposes various limitations on the value of attributes, be it a more restrictive character set for the "serial" attribute or fixed sized buffers for almost all the others, including "token", "manufacturer", and "model" attributes. However, the PKCS#11 URI notation does not impose such limitations aside from removing generic and PKCS#11 URI delimiters from a permitted character set. We believe that being too restrictive on the attribute values could limit the PKCS#11 URI usefulness. What is more, possible future changes to the PKCS#11 specification should not affect existing attributes.

A PKCS#11 URI takes the form (for explanation of Augmented BNF, see [RFC5234]):

```
pk11-URI             = "pkcs11" ":" pk11-path *1("?" pk11-query)
; Path component and its attributes.  Path may be empty.
pk11-path            = *1(pk11-pattr *(";" pk11-pattr))
pk11-pattr           = pk11-token / pk11-manuf / pk11-serial /
                       pk11-model / pk11-lib-manuf /
                       pk11-lib-ver / pk11-lib-desc /
                       pk11-object / pk11-type / pk11-id /
                       pk11-x-pattr
; Query component and its attributes.  Query may be empty.
pk11-qattr           = pk11-pin-source / pk11-pin-value /
                       pk11-module-name / pk11-module-path /
                       pk11-x-qattr
pk11-query           = *1(pk11-qattr *("&" pk11-qattr))
; RFC 3986 section 2.2 mandates all potentially reserved characters
; that do not conflict with actual delimiters of the URI do not have
; to be percent-encoded.
pk11-res-avail       = ":" / "[" / "]" / "@" / "!" / "$" /
                       "'" / "(" / ")" / "*" / "+" / "," / "="
pk11-path-res-avail  = pk11-res-avail / "&"
; We allow "/" and "?" in the query to be unencoded but "&" must
; be encoded since it may be used as a delimiter in the component.
pk11-query-res-avail = pk11-res-avail / "/" / "?" / "|"
pk11-pchar           = unreserved / pk11-path-res-avail / pct-encoded
pk11-qchar           = unreserved / pk11-query-res-avail / pct-encoded
pk11-token           = "token" "=" *pk11-pchar
pk11-manuf           = "manufacturer" "=" *pk11-pchar
pk11-serial          = "serial" "=" *pk11-pchar
pk11-model           = "model" "=" *pk11-pchar
pk11-lib-manuf       = "library-manufacturer" "=" *pk11-pchar
pk11-lib-desc        = "library-description" "=" *pk11-pchar
pk11-lib-ver         = "library-version" "=" 1*DIGIT *1("." 1*DIGIT)
pk11-object          = "object" "=" *pk11-pchar
pk11-type            = "type" "=" *1("public" / "private" / "cert" /
                       "secret-key" / "data")
pk11-id              = "id" "=" *pk11-pchar
pk11-pin-source      = "pin-source" "=" *pk11-qchar
pk11-pin-value       = "pin-value" "=" *pk11-qchar
pk11-module-name     = "module-name" = *pk11-qchar
pk11-module-path     = "module-path" = *pk11-qchar
pk11-x-attr-nm-char  = ALPHA / DIGIT / "-" / "_"
; Permitted value of a vendor specific attribute is based on
; whether the attribute is used in the path or in the query.
pk11-x-pattr         = "x-" 1*pk11-x-attr-nm-char "=" *pk11-pchar
pk11-x-qattr         = "x-" 1*pk11-x-attr-nm-char "=" *pk11-qchar
```

The URI path component contains attributes that identify a resource in a one level hierarchy provided by Cryptoki producers. The query component can contain a few attributes that may be needed to retrieve the resource identified by the URI path. Both path and query components may contain vendor specific attributes. Such attribute names must start with an "x-" prefix. Attributes in the path component are delimited by ';' character, attributes in the query component use '&' as a delimiter.

The general '/' delimiter was removed from available characters that do not have to be percent-encoded in the path component so that generic URI parsers never split the path component into multiple segments. The '/' delimiter can be used unencoded in the query component. Delimiter '?' was removed since the PKCS#11 URI uses a query component. Delimiter '#' was removed so that generic URI parsers are not confused by unencoded hash characters. All other generic delimiters are allowed to be used unencoded (':', '[', ']', and '@') in the PKCS#11 URI.

The following table presents mapping between the PKCS#11 URI path component attributes and the PKCS#11 API structure members and object attributes. Given that PKCS#11 URI users may be quite ignorant about the PKCS#11 specification the mapping is a product of a necessary compromise between how precisely are the URI attribute names mapped to the names in the specification and the ease of use and understanding of the URI scheme.

URI component path attribute name	Attribute represents	Attribute corresponds in the PKCS#11 specification to
id	key identifier for object	"CKA_ID" object attribute
library-description	character-string description of the library	"libraryDescription" member of CK_INFO structure
library-manufacturer	ID of the Cryptoki library manufacturer	"manufacturerID" member of the CK_INFO structure
library-version	Cryptoki library version number	"libraryVersion" member of CK_INFO

		structure
manufacturer	ID of the token manufacturer	"manufacturerID" member of CK_TOKEN_INFO structure
model	token model	"model" member of CK_TOKEN_INFO structure
object	description (name) of the object	"CKA_LABEL" object attribute
serial	character-string serial number of the token	"serialNumber" member of CK_TOKEN_INFO structure
token	application-defined label, assigned during token initialization	"label" member of the CK_TOKEN_INFO structure
type	object class (type)	"CKA_CLASS" object attribute

Table 1: Mapping between URI path component attributes and PKCS#11 specification names

The query component attribute "pin-source" specifies where the application or library should find the normal user's token PIN, the "pin-value" attribute provides the normal user's PIN value directly, if needed, and the "module-name" and "module-path" attributes modify default settings for accessing PKCS#11 providers. For the definition of a "normal user", see [pkcs11_spec].

The ABNF rules above is a best effort definition and this paragraph specifies additional constraints. The PKCS#11 URI must not contain duplicate attributes of the same name in the URI path component. It means that each attribute may be present at most once in the PKCS#11 URI path. Aside from the query attributes defined in this document, duplicate attributes may be present in the URI query component and it is up to the URI consumer to decide on how to deal with such duplicates.

It is recommended to percent-encode the whole value of the "id" attribute which is supposed to be handled as arbitrary binary data.

The "library-version" attribute represents the major and minor version number of the library and its format is "M.N". Both numbers are one byte in size, see the "libraryVersion" member of the CK_INFO structure in [pkcs11_spec] for more information. Value "M" for the attribute must be interpreted as "M" for the major and "0" for the minor version of the library. If the attribute is present the major version number is mandatory. Both "M" and "N" must be decimal numbers.

An empty PKCS#11 URI path attribute that does allow for an empty value matches a corresponding structure member or an object attribute with an empty value. Note that according to the PKCS#11 specification [pkcs11_spec], empty character values in a PKCS#11 API producer must be padded with spaces and should not be NULL terminated.

3.4. PKCS#11 URI Scheme Query Attribute Semantics

An application may always ask for a PIN by any means it decides to. What is more, in order not to limit PKCS#11 URI portability the "pin-source" attribute value format and interpretation is left to be implementation specific. However, we recommend the certain rules to be followed in descending order for the value of the "pin-source" attribute:

- if the value represents a local absolute path the implementation should use it as a PIN file containing the PIN value
- if the value contains "|<absolute-command-path>" the implementation should read the PIN from the output of an application specified with absolute path "<absolute-command-path>". Note that character "|" representing a pipe does not have to be percent encoded in the query component of the PKCS#11 URI.
- if the value represents a URI treat it as an object containing the PIN. Such a URI may be "file:", "https:", another PKCS#11 URI, or something else.
- interpret the value as needed in an implementation dependent way

If a URI contains both "pin-source" and "pin-value" query attributes the URI should be refused as invalid.

Use of the "pin-value" attribute may have security related consequences. Section 6 should be consulted before this attribute is

ever used. Standard percent encoding rules should be followed for the attribute value.

A consumer of PKCS#11 URIs may modify default settings for accessing a PKCS#11 provider or providers by accepting query component attributes "module-name" and "module-path"."

Processing the URI query module attributes should follow these rules:

- attribute "module-name" is expected to contain a case-insensitive PKCS#11 module name (not path nor filename) without system specific affixes. Such affix could be an ".so" or ".DLL" suffix, or a "lib" prefix, for example. Not using system specific affixes is expected to increase portability of PKCS#11 URIs among different systems. A URI consumer searching for PKCS#11 modules is expected to use a system or application specific locations to find modules based on the name provided in the attribute.

- attribute "module-path" is expected to contain a system specific absolute path to the PKCS#11 module, or a system specific absolute path to the directory of where PKCS#11 modules are located. For security reasons, a URI with a relative path in this attribute should be always rejected.

- the URI consumer may refuse to accept either of the attributes, or both. If use of an attribute present in the URI string is not accepted a warning message should be presented to the provider of the URI.

- if either of the module attributes is present, only those modules found matching these query attributes should be used to search for an object represented by the URI.

- use of the module attributes does not suppress matching of any other URI path component attributes present in a URI.

- semantics of using both attributes in the same URI string is implementation specific but such use should be avoided. Attribute "module-name" is preferred to "module-path" due to its system independent nature but the latter may be more suitable for development and debugging.

- a URI may not contain multiple module attributes of the same name.

Use of the module attributes may have security related consequences. Section 6 should be consulted before these attributes are ever used.

A word "module" was chosen over word "library" in these query attribute names to avoid confusion with semantically different library attributes used in the URI path component.

3.5. PKCS#11 URI Matching Guidelines

The PKCS#11 URI can identify PKCS#11 storage objects, tokens, or Cryptoki libraries. Note that since a URI may identify three different types of entities the context within which the URI is used may be needed to determine the type. For example, a URI with only library attributes may either represent all objects in all tokens in all Cryptoki libraries identified by the URI, all tokens in those libraries, or just the libraries.

The following guidelines should help a PKCS#11 URI consumer (eg. an application accepting PKCS#11 URIs) to match the URI with the desired resource.

- the consumer must know whether the URI is to identify PKCS#11 storage object(s), token(s), or Cryptoki producer(s).
- if the consumer is willing to accept query component module attributes only those PKCS#11 providers matching these attributes should be worked with. See Section 3.4 for more information.
- an unrecognized attribute in the URI path component, including a vendor specific attribute, should result in an empty set of matched resources. The consumer should consider whether an error message presented to the user is appropriate in such a case.
- an unrecognized attribute in the URI query should be ignored. The consumer should consider whether a warning message presented to the user is appropriate in such a case.
- an attribute not present in the URI path but known to a consumer matches everything. Each additional attribute present in the URI path further restricts the selection.
- a logical extension of the above is that an empty URI path matches everything. For example, if used to identify storage objects it matches all accessible objects in all tokens provided by all PKCS#11 API producers found in the system.
- use of PIN attributes may change the set of storage objects visible to the consumer.
- in addition to query component attributes defined in this document, vendor specific query attributes may contain further

information about how to perform the selection or other related information.

3.6. PKCS#11 URI Comparison

Comparison of two URIs is a way of determining whether the URIs are equivalent without comparing the actual resource the URIs point to. The comparison of URIs aims to minimize false negatives while strictly avoiding false positives.

Two PKCS#11 URIs are said to be equal if URIs as character strings are identical as specified in Section 6.2.1 of [RFC3986], or if both following rules are fulfilled:

- set of attributes present in the URI is equal. Note that the ordering of attributes in the URI string is not significant for the mechanism of comparison.

- values of respective attributes are equal based on rules specified below

The rules for comparing values of respective attributes are:

- values of path component attributes "library-description", "library-manufacturer", "manufacturer", "model", "object", "serial", "token", "type", and query component attribute "module-name" must be compared using a simple string comparison as specified in Section 6.2.1 of [RFC3986] after the case and the percent-encoding normalization are both applied as specified in Section 6.2.2 of [RFC3986].

- value of attribute "id" must be compared using the simple string comparison after all bytes are percent-encoded using uppercase letters for digits A-F.

- value of attribute "library-version" must be processed as a specific scheme-based normalization permitted by Section 6.2.3 of [RFC3986]. The value must be split into a major and minor version with character '.' (dot) serving as a delimiter. Library version "M" must be treated as "M" for the major version and "0" for the minor version. Resulting minor and major version numbers must be then separately compared numerically.

- value of "pin-source", if deemed containing the filename with the PIN value, must be compared using the simple string comparison after the full syntax based normalization as specified in Section 6.2.2 of [RFC3986] is applied. If value of the "pin-source" attribute is believed to be overloaded it is recommended

to perform case and percent-encoding normalization before the values are compared but the exact mechanism of comparison is left to the application.

- value of attribute "module-path" must be compared using the simple string comparison after the full syntax based normalization as specified in Section 6.2.2 of [RFC3986] is applied.

- when comparing vendor specific attributes it is recommended to perform case and percent-encoding normalization before the values are compared but the exact mechanism of such a comparison is left to the application.

4. Examples of PKCS#11 URIs

This section contains some examples of how PKCS#11 token objects, PKCS#11 tokens, and PKCS#11 libraries can be identified using the PKCS#11 URI scheme. Note that in some of the following examples, newlines and spaces were inserted for better readability. As specified in Appendix C of [RFC3986], whitespace should be ignored when extracting the URI. Also note that all spaces as part of the URI are percent-encoded, as specified in Appendix A of [RFC3986].

An empty PKCS#11 URI might be useful to PKCS#11 consumers. See Section 3.5 for more information on semantics of such a URI.

```
pkcs11:
```

One of the simplest and most useful forms might be a PKCS#11 URI that specifies only an object label and its type. The default token is used so the URI does not specify it. Note that when specifying public objects, a token PIN might not be required.

```
pkcs11:object=my-pubkey;type=public
```

When a private key is specified either the "pin-source" attribute, "pin-value, or an application specific method would be usually used. Note that '/' is not percent-encoded in the "pin-source" attribute value since this attribute is part of the query component, not the path, and thus is separated by '?' from the rest of the URI.

```
pkcs11:object=my-key;type=private?pin-source=/etc/token
```

The following example identifies a certificate in the software token. Note an empty value for the attribute "serial" which matches only empty "serialNumber" member of the "CK_TOKEN_INFO" structure. Also note that the "id" attribute value is entirely percent-encoded, as recommended. While ',' is in the reserved set it does not have to be percent-encoded since it does not conflict with any sub-delimiters used. The '#' character as in "The Software PKCS#11 Softtoken" must be percent-encoded.

```
pkcs11:token=The%20Software%20PKCS%2311%20Softtoken;
       manufacturer=Snake%20Oil,%20Inc.;
       model=1.0;
       object=my-certificate;
       type=cert;
       id=%69%95%3E%5C%F4%BD%EC%91;
       serial=
       ?pin-source=/etc/token_pin
```

The next example covers how to use the "module-name" query attribute. Considering that the module is located in /usr/lib/libmypkcs11.so.1 file, the attribute value is "mypkcs11", meaning only the module name without the full path, and without the platform specific "lib" prefix and ".so.1" suffix.

```
pkcs11:object=my-sign-key;
       type=private
       ?module-name=mypkcs11
```

The following example covers how to use the "module-path" query attribute. The attribute may be useful if a user needs to provide the key via a PKCS#11 module stored on a removable media, for example. Getting the PIN to access the private key here is left to be application specific.

```
pkcs11:object=my-sign-key;
       type=private
       ?module-path=/mnt/libmypkcs11.so.1
```

In the context where a token is expected the token can be identified without specifying any PKCS#11 objects. A PIN might still be needed in the context of listing all objects in the token, for example. Section 6 should be consulted before the "pin-value" attribute is ever used.

```
pkcs11:token=Software%20PKCS%2311%20softtoken;
       manufacturer=Snake%20Oil,%20Inc.
       ?pin-value=the-pin
```

The Cryptoki library alone can be also identified without specifying a PKCS#11 token or object.

```
pkcs11:library-manufacturer=Snake%20Oil,%20Inc.;
       library-description=Soft%20Token%20Library;
       library-version=1.23
```

The following example shows that the attribute value can contain a semicolon. In such case, it is percent-encoded. The token attribute value must be read as "My token; created by Joe". Lower case letters can also be used in percent-encoding as shown below in the "id" attribute value but note that Sections 2.1 and 6.2.2.1 of [RFC3986] read that all percent-encoded characters should use the uppercase hexadecimal digits. More specifically, if the URI string was to be compared the algorithm defined in Section 3.6 explicitly requires percent-encoding to use the uppercase digits A-F in the "id" attribute values. And as explained in Section 3.3, library version "3" should be interpreted as "3" for the major and "0" for the minor version of the library.

```
pkcs11:token=My%20token%25%20created%20by%20Joe;
       library-version=3;
       id=%01%02%03%Ba%dd%Ca%fe%04%05%06
```

If there is any need to include literal "%;" substring, for example, both characters must be escaped. The token value must be read as "A name with a substring %;".

```
pkcs11:token=A%20name%20with%20a%20substring%20%25%3B;
       object=my-certificate;
       type=cert
```

The next example includes a small A with acute in the token name. It must be encoded in octets according to the UTF-8 character encoding and then percent-encoded. Given that a small A with acute is U+225 unicode code point, the UTF-8 encoding is 195 161 in decimal, and that is "%C3%A1" in percent-encoding.

```
pkcs11:token=Name%20with%20a%20small%20A%20with%20acute:%20%C3%A1;
       object=my-certificate;
       type=cert
```

Both the path and query components may contain vendor specific attributes. Attributes in the query component must be delimited by '&'.

```
pkcs11:token=my-token;
       object=my-certificate;
       type=cert;
       x-vend-aaa=value-a
       ?pin-source=/etc/token_pin
       &x-vend-bbb=value-b
```

5. IANA Considerations

This document moves the "pkcs11" URI scheme from the provisional to permanent URI scheme registry. The registration template for the URI scheme is accessible on http://www.iana.org/assignments/uri-schemes.

6. Security Considerations

There are general security considerations for URI schemes discussed in Section 7 of [RFC3986].

From those security considerations, Section 7.1 of [RFC3986] applies since there is no guarantee that the same PKCS#11 URI will always identify the same object, token, or a library in the future.

Section 7.2 of [RFC3986] applies since by accepting query component attributes "module-name" or "module-path" the consumer potentially allows loading of arbitrary code into a process.

Section 7.5 of [RFC3986] applies since the PKCS#11 URI may be used in world readable command line arguments to run applications, stored in public configuration files, or otherwise used in clear text. For that reason the "pin-value" attribute should only be used if the URI string itself is protected with the same level of security as the token PIN itself otherwise is.

7. References

7.1. Normative References

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", RFC 3629, STD 63, November 2003.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", RFC 3986, STD 66, January 2005.

[RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 5234, STD 68, January 2008.

7.2. Informative References

[RFC4395] Hansen, T., Hardie, T., and L. Masinter, "Guidelines and Registration Procedures for New URI Schemes", RFC 4395, February 2006.

[pkcs11_spec] RSA Laboratories, "PKCS #11: Cryptographic Token Interface Standard v2.20", June 2004.

Authors' Addresses

Jan Pechanec
Oracle Corporation
4180 Network Circle
Santa Clara CA 95054
USA

Email: Jan.Pechanec@Oracle.COM
URI: http://www.oracle.com

Darren J. Moffat
Oracle Corporation
Oracle Parkway
Thames Valley Park
Reading RG6 1RA
UK

Email: Darren.Moffat@Oracle.COM
URI: http://www.oracle.com