

Network Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: April 03, 2014

J. Pechanec  
D. Moffat  
Oracle Corporation  
September 30, 2013

The PKCS#11 URI Scheme  
draft-pechanec-pkcs11uri-13

## 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 03, 2014.

## Copyright Notice

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

Internet-Draft

The PKCS#11 URI Scheme

September 2013

This document is subject to [BCP 78](http://trustee.ietf.org/license-info) 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.

## Table of Contents

<a href="#">1.</a>	Introduction . . . . .	<a href="#">2</a>
<a href="#">2.</a>	Contributors . . . . .	<a href="#">3</a>
<a href="#">3.</a>	PKCS#11 URI Scheme Definition . . . . .	<a href="#">3</a>
<a href="#">3.1.</a>	PKCS#11 URI Scheme Name . . . . .	<a href="#">4</a>
<a href="#">3.2.</a>	PKCS#11 URI Scheme Status . . . . .	<a href="#">4</a>
<a href="#">3.3.</a>	PKCS#11 URI Scheme Syntax . . . . .	<a href="#">4</a>
<a href="#">3.4.</a>	PKCS#11 URI Matching Guidelines . . . . .	<a href="#">7</a>
<a href="#">3.5.</a>	PKCS#11 URI Comparison . . . . .	<a href="#">8</a>
<a href="#">4.</a>	Examples of PKCS#11 URIs . . . . .	<a href="#">9</a>
<a href="#">5.</a>	IANA Considerations . . . . .	<a href="#">11</a>
<a href="#">6.</a>	Security Considerations . . . . .	<a href="#">11</a>
<a href="#">7.</a>	References . . . . .	<a href="#">12</a>
<a href="#">7.1.</a>	Normative References . . . . .	<a href="#">12</a>
<a href="#">7.2.</a>	Informative References . . . . .	<a href="#">12</a>
	Authors' Addresses . . . . .	<a href="#">12</a>

## [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(s).

The PKCS#11 URI cannot identify other objects aside from storage objects, for example a hardware feature or 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 it is left to its implementation to provide adequate user interface to locate and load such producer(s).

## [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

Pechanec & Moffat

Expires April 03, 2014

[Page 3]

---

Internet-Draft

The PKCS#11 URI Scheme

September 2013

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's 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-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-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 may contain a PIN source attribute 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 attribute "token" represents a token label and corresponds to the "label" member of the CK\_TOKEN\_INFO structure, the attribute "manufacturer" corresponds to the "manufacturerID" member of CK\_TOKEN\_INFO, the attribute "serial" corresponds to the "serialNumber" member of CK\_TOKEN\_INFO, the attribute "model" corresponds to the "model" member of CK\_TOKEN\_INFO, the attribute "library-manufacturer" represents the Cryptoki library manufacturer and corresponds to the "manufacturerID" member of the CK\_INFO structure, the attribute "library-description" corresponds to the "libraryDescription" member of CK\_INFO, the attribute "library-version" corresponds to the "libraryVersion" member of CK\_INFO, the attribute "object" represents a PKCS#11 object label and corresponds to the "CKA\_LABEL" object attribute, the attribute "type" represents the type of the object and corresponds to the "CKA\_CLASS" object attribute, the attribute "id" represents the object ID and corresponds to the "CKA\_ID" object attribute, and the attribute "pin-source" specifies where the application or library should find the token PIN, if needed.

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 "pin-source" attribute, 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.

The "pin-source" attribute may represent a filename that contains a token PIN but an application may overload this attribute. For example, "pin-source=%7Cprog-name" could mean to read a PIN from an

external application (%7C denotes a pipe '|' character). Note that an application may always ask for a PIN and/or interpret the "pin-source" attribute by any means it decides to. However, as discussed in [Section 6](#), the attribute should never contain the PIN itself.

It is recommended to percent-encode the whole value of the "id" attribute which is supposed to be handled as arbitrary binary data. Value "M" of the "library-version" attribute should be interpreted as

"M" for the major and "0" for the minor version of the library. Note that if the "library-version" attribute is present, the major version number is mandatory.

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 Matching Guidelines

The PKCS#11 URI can identify PKCS#11 storage objects, tokens, or Cryptoki 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.

- o the consumer must know whether the URI is to identify PKCS#11 storage object(s), token(s), or Cryptoki producer(s).
- o 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.
- o 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.
- o 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.
- o 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.

- o use of the PIN attribute may change the set of storage objects



visible to the consumer.

- o in addition to the PIN attribute, query string attributes may contain further information about how to perform the selection or other related information.

### [3.5.](#) 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:

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

The rules for comparing values of respective attributes are:

- o values of attributes "library-description", "library-manufacturer", "manufacturer", "model", "object", "serial", "token", and "type" 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 \[RFC3986\]](#)
- o 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
- o value for attribute "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.

- o 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.
- o 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 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:

```
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 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". 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 token alone can be identified without specifying any PKCS#11 objects. A PIN may still be needed to list all objects, for example.

```
pkcs11:token=Software%20PKCS%2311%20softtoken;
    manufacturer=Snake%20Oil,%20Inc.
    ?pin-source=/etc/token_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 3.5](#) 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
    ?pin-source=/etc/token_pin
```

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 may be delimited by either ';' or '&'. We use '&' in the example that follows.

```
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 the 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 7.5 of \[RFC3986\]](#) applies since the PKCS#11 URI may be used in command line arguments to run applications, and those arguments can be world readable on some systems. For that reasons, the URI intentionally does not allow for specifying the PKCS#11 token PIN as a URI attribute.

## 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>

Pechanec & Moffat

Expires April 03, 2014

[Page 12]

---

Internet-Draft

The PKCS#11 URI Scheme

September 2013

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

Email: Darren.Moffat@Oracle.COM

URI: <http://www.oracle.com>

