Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 1, 2012

J. Pechanec
D. Moffat
Oracle Corporation

February 29, 2012

# The PKCS#11 URI Scheme
# draft-pechanec-pkcs11uri-06

## 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 September 1, 2012.

## Copyright Notice

described in the Simplified BSD License.

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 object in a PKCS#11 token, or an existing token itself, or an existing Cryptoki library. The set of object types that can be stored in a PKCS#11 token includes a public key, a private key, a certificate, a 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 an 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.

Note that the 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 existing objects, tokens, or Cryptoki libraries.

The URI scheme defined in this document is designed specifically with a mapping to the PKCS#11 API in mind. The URI uses only the scheme and the path components which are required by the Uniform Resource Identifier generic syntax [RFC3986]. The URI scheme 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 scheme does not use the optional query and fragment elements.

## 2. Contributors

Stef Walter, Nikos Mavrogiannopoulos, and Nico Williams 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 scheme is a sequence of attribute value pairs separated by a semicolon. In accordance with [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. Rules "unreserved" and "pct-encoded" in the PKCS#11 URI specification below were imported from [RFC3986]. As a special case, note that according to [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 will not affect existing attributes.





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

```
pk11-URI            = "pkcs11" ":" pk11-identifier
pk11-identifier     = *1(pk11-attr *(";" pk11-attr))
pk11-attr           = pk11-token / pk11-manuf / pk11-serial /
                      pk11-model / pk11-lib-manuf /
                      pk11-lib-ver / pk11-lib-desc /
                      pk11-object / pk11-object-type / pk11-id /
                      pk11-pin-source
; Section 2.2 of RFC 3986 specifies that all potentially reserved
; characters that do not conflict with actual delimiters of the URI
; do not have to be percent-encoded. So, ";" was removed as a
; sub-delimiter of the PKCS#11 URI's path and "/", "?", and "#" as
; delimiters in a generic URI syntax.
pk11-reserved-avail = ":" / "[" / "]" / "@" / "!" / "$" /
                      "&" / "'" / "(" / ")" / "*" / "+" /
                      "," / "="
pk11-char           = unreserved / pk11-reserved-avail /
                      pct-encoded
pk11-token          = "token" "=" *pk11-char
pk11-manuf          = "manufacturer" "=" *pk11-char
```

```
pk11-serial         = "serial" "=" *pk11-char
pk11-model          = "model" "=" *pk11-char
pk11-lib-manuf      = "library-manufacturer" "=" *pk11-char
pk11-lib-desc       = "library-description" "=" *pk11-char
pk11-lib-ver        = "library-version" "=" *DIGIT *1("." 1*DIGIT)
pk11-object         = "object" "=" *pk11-char
pk11-object-type    = "object-type" "=" *1("public" / "private" /
                      "cert" / "secret-key" / "data")
pk11-id             = "id" "=" *pk11-char
pk11-pin-source     = "pin-source" "=" *pk11-char
```

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 "object-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 "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.

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

## 4. Examples of PKCS#11 URI Schemes

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 which is allowed by [RFC3986]. Also note that all spaces as part of the URI are percent-encoded, as required by [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;object-type=public
```

When a private key is specified either the "pin-source" attribute or an application specific method would be usually used. Also note that "/" must be percent-encoded in the "pin-source" attribute value since it must be prevented to be mistaken for a path segment delimiter.

```
pkcs11:object=my-key;object-type=private;
       pin-source=%2Fetc%2Ftoken_pin
```





The following example identifies a certificate in the software token. Note that all attributes may have an empty value. In our case, "serial" is empty. It is up to the consumer of the URI to perform necessary checks if that is not allowed. 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" is a general

delimiter as "/" so it must be percent-encoded, too.

```
  pkcs11:token=The%20Software%20PKCS%2311%20Softtoken;
         manufacturer=Snake%20Oil,%20Inc.;
         serial=;
         model=1.0;
         object=my-certificate;
         object-type=cert;
         id=%69%95%3E%5C%f4%BD%EC%91;
         pin-source=%2Fetc%2Ftoken_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=%2Fetc%2Ftoken_pin
```

The Cryptoki library alone can be also identified without specifying any PKCS#11 objects.

```
  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 value must be read as "My token; created by Joe". Lower characters can also be used in percent-encoding as shown below in the "id" attribute value but note that [RFC3986] recommends to use upercase hexadecimal digits for all percent-encoded characters. Library version ".9" should be interpreted as "0" for the major and "9" for the minor version of the library. Similarly, library version "9" would be interpreted as "9" for the major and "0" for the minor version of the library.

```
  pkcs11:token=My%20token%25%20created%20by%20Joe;
         library-version=.9
         id=%69%95%3e%5c%f4%bd%ec%91;
```





And 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 strange substring '\;'".

```
pkcs11:token=A%20name%20with%20a%20strange%20substring%20'%25%3B';
       object=my-certificate;
       object-type=cert;
       pin-source=%2Fetc%2Ftoken_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;
       object-type=cert
```

## 5. IANA Considerations

This document registers a URI scheme. The registration template can be found in Section 3 of this document.

## 6. Security Considerations

There are security considerations for URI schemes discussed in [RFC3986].

Given that the PKCS#11 URI is also supposed to be used in command line arguments to running programs, and those arguments can be world readable on some systems, the URI intentionaly does not allow for specifying the PKCS#11 token PIN as a URI attribute.

## 7. 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.

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

February 2006.

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

[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
US

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