Internet Engineering Task Force (IETF)
Request for Comments: 7512
Category: Standards Track
ISSN: 2070-1721

J. Pechanec
D. Moffat
Oracle Corporation

April 2015

The PKCS #11 URI Scheme

Abstract

This memo specifies a PKCS #11 Uniform Resource Identifier (URI) Scheme for identifying PKCS #11 objects stored in PKCS #11 tokens and also for identifying PKCS #11 tokens, slots, or libraries. The URI scheme is based on how PKCS #11 objects, tokens, slots, and libraries are identified in "PKCS #11 v2.20: Cryptographic Token Interface Standard".

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc7512.

Table of Contents

1. Introduction

"PKCS #11 v2.20: Cryptographic Token Interface Standard" [PKCS11] specifies an API, called Cryptoki, for devices that 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, a slot, 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 data object; and a public, private, or secret key. 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, manufacturer name, serial number, and token model. Attributes that can identify a slot are a slot ID, description, and manufacturer. Attributes that can identify a Cryptoki library are a library

manufacturer, description, and 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.

A PKCS #11 URI cannot identify other objects defined in the specification [PKCS11] 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, slot, or an API producer in such a case.

A subset of existing PKCS #11 structure members and object attributes was chosen to uniquely identify a PKCS #11 storage object, token, slot, 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, tokens, and slots through the PKCS #11 API, or to 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 scheme definition uses the scheme, path, and query components defined in the "Uniform Resource Identifier (URI): Generic Syntax" [RFC3986] document. 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 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.

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. PKCS #11 URI Scheme Definition

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

2.1. PKCS #11 URI Scheme Name

pkcs11

2.2. PKCS #11 URI Scheme Status

Permanent

2.3. PKCS #11 URI Scheme Syntax

A 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. Except for the value of the "id" attribute defined later in this section, these attribute value pairs and query components are composed entirely of textual data and therefore SHOULD all first be encoded as octets according to the UTF-8 character encoding [RFC3629], in accordance with Section 2.5 of [RFC3986]; 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. Note that the value of the "id" attribute SHOULD NOT be encoded as UTF-8 because it can contain non-textual data, instead it SHOULD be entirely percent-encoded. See important caveats in Sections 2.5 and 5 regarding working with UTF-8 strings containing characters outside the US-ASCII character set.

Grammar rules "unreserved" and "pct-encoded" in the PKCS #11 URI scheme definition below are imported from [RFC3986]. As a special case, note that according to Appendix A of [RFC3986], a space must be percent-encoded.

The PKCS #11 specification imposes various limitations on the value of attributes, be it a more restrictive character set for the "serial" attribute or fixed-size buffers for almost all the others, including "token", "manufacturer", and "model" attributes. The syntax of the PKCS #11 URI scheme does not impose such limitations. However, if the consumer of a PKCS #11 URI encounters values that would not be accepted by the PKCS #11 specification, it MUST refuse the URI as invalid.

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

```
pk11-URI             = "pkcs11:" pk11-path [ "?" pk11-query ]
; Path component and its attributes.  Path may be empty.
pk11-path            = [ 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-slot-desc / pk11-slot-manuf /
                       pk11-slot-id / pk11-v-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-v-qattr
pk11-query           = [ pk11-qattr *("&" pk11-qattr) ]
; Section 2.2 of [RFC3986] 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 / "&"
; "/" and "?" in the query component MAY be unencoded but "&" MUST
; be encoded since it functions as a delimiter within 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*DIGIT ]
pk11-object          = "object" "=" *pk11-pchar
pk11-type            = "type" "=" ( "public" / "private" / "cert" /
                       "secret-key" / "data" )
pk11-id              = "id" "=" *pk11-pchar
pk11-slot-manuf      = "slot-manufacturer" "=" *pk11-pchar
pk11-slot-desc       = "slot-description" "=" *pk11-pchar
pk11-slot-id         = "slot-id" "=" 1*DIGIT
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-v-attr-nm-char  = ALPHA / DIGIT / "-" / "_"
; The permitted value of a vendor-specific attribute is based on
; whether the attribute is used in the path or in the query.
pk11-v-pattr         = 1*pk11-v-attr-nm-char "=" *pk11-pchar
pk11-v-qattr         = 1*pk11-v-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 component. Attributes in the path component are delimited by the ';' character, attributes in the query component use '&' as a delimiter.

Both path and query components MAY contain vendor-specific attributes. Such attribute names MUST NOT clash with existing attribute names. Note that in accordance with [BCP178], the previously used convention of starting vendor attributes with an "x-" prefix is now deprecated.

The general '/' delimiter MUST be percent-encoded in the path component so that generic URI parsers never split the path component into multiple segments. It MAY be unencoded in the query component. The delimiter '?' MUST be percent-encoded in the path component since the PKCS #11 URI scheme uses a query component. The delimiter '#' MUST be always percent-encoded so that generic URI parsers do not treat a hash as a beginning of a fragment identifier component. All other generic delimiters MAY be used unencoded (':', '[', ']', and '@') in a 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 the URI attribute names are 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	PKCS #11 specification counterpart
id	key identifier for object	"CKA_ID" object attribute
library-description	character-string description of the library	"libraryDescription" member of CK_INFO structure. It is a UTF-8 string.

library-manufacturer	ID of the Cryptoki library manufacturer	"manufacturerID" member of the CK_INFO structure. It is a UTF-8 string.
library-version	Cryptoki library version number	"libraryVersion" member of the CK_INFO structure.
manufacturer	ID of the token manufacturer	"manufacturerID" member of CK_TOKEN_INFO structure. It is a UTF-8 string.
model	token model	"model" member of CK_TOKEN_INFO structure. It is a UTF-8 string.
object	description (name) of the object	"CKA_LABEL" object attribute. It is a UTF-8 string.
serial	character-string serial number of the token	"serialNumber" member of CK_TOKEN_INFO structure.
slot-description	slot description	"slotDescription" member of CK_SLOT_INFO structure. It is a UTF-8 string.
slot-id	Cryptoki-assigned value that identifies a slot	decimal number of "CK_SLOT_ID" type.
slot-manufacturer	ID of the slot manufacturer	"manufacturerID" member of CK_SLOT_INFO structure. It is a UTF-8 string.

token	application-defined label, assigned during token initialization	"label" member of the CK_TOKEN_INFO structure. It is a UTF-8 string.
type	object class (type)	"CKA_CLASS" object attribute.

Table 1: Mapping between URI Path Component Attributes and PKCS #11 Specification Names

The following table presents mapping between the "type" attribute values and corresponding PKCS #11 object classes.

Attribute value	PKCS #11 object class
cert	CKO_CERTIFICATE
data	CKO_DATA
private	CKO_PRIVATE_KEY
public	CKO_PUBLIC_KEY
secret-key	CKO_SECRET_KEY

Table 2: Mapping between the "type" Attribute and PKCS #11 Object Classes

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].

The ABNF rules above are a best-effort definition, and this paragraph specifies additional constraints. A 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 component. Aside from the query attributes defined in this document, duplicate (vendor) 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.

As stated earlier in this section, the value of the "id" attribute can contain non-textual data. This is because the corresponding PKCS #11 "CKA_ID" object attribute can contain arbitrary binary data.

Therefore, the whole value of the "id" attribute SHOULD be percent-encoded.

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] 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 REQUIRED. Both "M" and "N" MUST be decimal numbers.

Slot ID is a Cryptoki-assigned number that is not guaranteed to be stable across PKCS #11 module initializations. However, there are certain libraries and modules that provide stable slot identifiers. For these cases, when the slot description and manufacturer ID is not sufficient to uniquely identify a specific reader, the slot ID MAY be used to increase the precision of the token identification. In other scenarios, using the slot IDs is likely to cause usability issues.

An empty PKCS #11 URI path component 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], empty character values in a PKCS #11 API producer must be padded with spaces and should not be NULL terminated.

2.4. PKCS #11 URI Scheme Query Attribute Semantics

An application can 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, the following rules SHOULD be followed in descending order for the value of the "pin-source" attribute:

- If the value represents a URI, it SHOULD be treated as an object containing the PIN. Such a URI may be "file:", "https:", another PKCS #11 URI, or something else.

- 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 a PKCS #11 URI.

- 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 accept query component attributes "module-name" and "module-path" in order to modify default settings for accessing a PKCS #11 provider or providers.

Processing the URI query module attributes SHOULD follow these rules:

- The attribute "module-name" SHOULD contain a case-insensitive PKCS #11 module name (not path nor filename) without system-specific affixes; said affix could be a ".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 SHOULD use a system or application-specific locations to find modules based on the name provided in the attribute.

- The attribute "module-path" SHOULD 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 rejected.

- The URI consumer MAY refuse to accept either of the attributes, or both. If use of the attribute present in the URI string is not accepted, a warning message SHOULD be presented to the provider of the URI and system-specific module locations SHOULD be used.

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

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

- The 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 MUST 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 a word "library" in these query attribute names to avoid confusion with semantically different library attributes used in the URI path component.

2.5. PKCS #11 URI Matching Guidelines

A PKCS #11 URI can identify PKCS #11 storage objects, tokens, slots, or Cryptoki libraries. Note that since a URI may identify four 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 can help a PKCS #11 URI consumer (e.g., an application accepting PKCS #11 URIs) to match the URI with the desired resource.

- The consumer needs to know whether the URI is to identify PKCS #11 storage object(s), token(s), slot(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 2.4 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 can 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 can consider whether a warning message presented to the user is appropriate in such a case.

- An attribute not present in the URI path component but known to a consumer matches everything. Each additional attribute present in the URI path component further restricts the selection.

- A logical extension of the above is that a URI with an empty path component matches everything. For example, if used to identify storage objects, it matches all accessible objects in all tokens provided by all relevant PKCS #11 API producers.

- Note that 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.

As noted in Section 5, the PKCS #11 specification is not clear about how to normalize UTF-8-encoded Unicode characters [RFC3629]. For that reason, it is RECOMMENDED not to use characters outside the US-ASCII character set for labels and names. However, those who discover a need to use such characters should be cautious, conservative, and expend extra effort to be sure they know what they are doing and that failure to do so may create both operational and security risks. It means that when matching UTF-8 string-based attributes (see Table 1) with characters outside the US-ASCII repertoire, normalizing all UTF-8 strings before string comparison may be the only safe approach. For example, for objects (keys), it means that PKCS #11 attribute search template would only contain attributes that are not UTF-8 strings and another pass through returned objects is then needed for UTF-8 string comparison after the normalization is applied.

2.6. PKCS #11 URI Comparison

Comparison of two URIs is a way of determining whether the URIs are equivalent without comparing the actual resource to which the URIs point. The comparison of URIs aims to minimize false negatives while strictly avoiding false positives. When working with UTF-8 strings with characters outside the US-ASCII character sets, see important caveats in Sections 2.5 and 5.

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 of the following rules are fulfilled:

- The 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.

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

The rules for comparing values of respective attributes are:

- The values of path component attributes "library-description", "library-manufacturer", "manufacturer", "model", "object", "serial", "slot-description", "slot-manufacturer", "token", "type", and the 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 were both applied as specified in Section 6.2.2 of [RFC3986].

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

- The value of the 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. A library-version "M" MUST be treated as "M" for the major version and "0" for the minor version. Then, resulting minor and major version numbers MUST be separately compared numerically.

- The value of the attribute "slot-id" MUST be processed as a specific scheme-based normalization permitted by Section 6.2.3 of [RFC3986] and compared numerically.

- The value of "pin-source", if containing a "file:" URI or "|<absolute-command-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. If the value of the "pin-source" attribute is believed to be overloaded, the case and percent-encoding normalization SHOULD be applied before the values are compared, but the exact mechanism of comparison is left to the application.

- The value of the 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, the case and percent-encoding normalization, as specified in Section 6.2.2 of [RFC3986], SHOULD be applied before the values are compared, but the exact mechanism of such a comparison is left to the application.

2.7. Generating PKCS #11 URIs

When generating URIs for PKCS #11 resources, the exact set of attributes used in a URI is inherently context specific. A PKCS #11 URI template [RFC6570] support MAY be provided by a URI-generating application to list URIs to access the same resource(s) again if the template captured the necessary context.

3. Examples of PKCS #11 URIs

This section contains some examples of how PKCS #11 token objects, tokens, slots, and libraries can be identified using the PKCS #11 URI scheme. Note that in some of the following examples, line breaks 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 that are part of the URIs are percent-encoded, as specified in Appendix A of [RFC3986].

An empty PKCS #11 URI might be useful to PKCS #11 consumers. See Section 2.5 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 may 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 component, and thus is separated by '?' from the rest of the URI.

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

The following example identifies a certificate in the software token. Note the use of 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=file:/etc/token_pin
```

The next example covers how to use the "module-name" query attribute. Considering that the module is located in the /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 of 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
```

In the context where a slot is expected, the slot can be identified without specifying any PKCS #11 objects in any token that may be inserted in the slot.

```
pkcs11:slot-description=Sun%20Metaslot
```

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 an attribute value with a semicolon. In such a case, it MUST be percent-encoded. The token attribute value MUST be read as "My token; created by Joe". Lowercase letters MAY 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] state that all percent-encoded characters SHOULD use the uppercase hexadecimal digits. More specifically, if the URI string were to be compared, the algorithm defined in Section 2.6 explicitly requires percent-encoding to use the uppercase digits A-F in the "id" attribute values. And as explained in Section 2.3, library version "3" MUST 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 a 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. See also Section 5 on the use of characters outside the US-ASCII character set for labels.

```
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;
       vendor-aaa=value-a
       ?pin-source=file:/etc/token_pin
       &vendor-bbb=value-b
```

4. IANA Considerations

4.1. URI Scheme Registration

This document moves the "pkcs11" URI scheme from the "Provisional URI Schemes" registry to the "Permanent URI Schemes" registry. The registration request complies with [RFC4395].

URI scheme name: pkcs11

URI scheme status: permanent

URI scheme syntax: Defined in Section 2.3 of [RFC7512].

URI scheme semantics: Defined in Section 1 of [RFC7512].

Encoding considerations: See Sections 2.3 and 5 of [RFC7512].

Applications/protocols that use this URI scheme name: For general information, see Section 1 of [RFC7512]. A list of known consumers of the PKCS #11 URI include GnuTLS, Gnome, p11-kit, Oracle Solaris 11 and higher, OpenSC, OpenConnect, and FreeIPA.

Interoperability considerations: See Section 5 of [RFC7512].

Security considerations: See Section 6 of [RFC7512].

Contact: Jan Pechanec <Jan.Pechanec@Oracle.com>, Darren Moffat <Darren.Moffat@Oracle.com>

Author/Change Controller: IESG <iesg@ietf.org>

References: [RFC7512]

5. Internationalization Considerations

The PKCS #11 specification does not specify a canonical form for strings of characters of the CK_UTF8CHAR type. This presents the usual false negative and false positive (aliasing) concerns that arise when dealing with unnormalized strings. Because all PKCS #11 items are local and local security is assumed, these concerns are mainly about usability and interoperability.

In order to improve the user experience, it is RECOMMENDED that applications that create PKCS #11 objects or label tokens not use characters outside the US-ASCII character set for the labels. If that is not possible, labels SHOULD be normalized to Normalization Form C (NFC) [UAX15]. For the same reason, PKCS #11 libraries, slots (token readers), and tokens SHOULD use US-ASCII characters only for their names, and if that is not possible, they SHOULD normalize their names to NFC. When listing PKCS #11 libraries, slots, tokens, and/or objects, an application SHOULD normalize their names to NFC if characters outside of the US-ASCII character set are expected. When matching PKCS #11 URIs to libraries, slots, tokens, and/or objects, applications MAY convert names to a chosen normalization form before the string comparison for matching, as those might predate these recommendations. See also Section 2.5.

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, slot, 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 a 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.

The PKCS #11 specification does not provide means to authenticate devices to users; it only authenticates users to tokens. Instead, local and physical security are demanded: the user must be in possession of their tokens, and the system into whose slots the

users' tokens are inserted must be secure. As a result, the usual security considerations regarding normalization do not arise. For the same reason, confusable script issues also do not arise. Nonetheless, if use of characters outside the US-ASCII character set is required, it is best to normalize to NFC all strings appearing in PKCS #11 API elements. See also Section 5.

7. References

7.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, November 2003, <http://www.rfc-editor.org/info/rfc3629>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005, <http://www.rfc-editor.org/info/rfc3986>.

[RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008, <http://www.rfc-editor.org/info/rfc5234>.

7.2. Informative References

[BCP178] Saint-Andre, P., Crocker, D., and M. Nottingham, "Deprecating the "X-" Prefix and Similar Constructs in Application Protocols", RFC 6648, BCP 178, June 2012, <http://www.rfc-editor.org/info/bcp178>.

[PKCS11] RSA Laboratories, "PKCS #11 v2.20: Cryptographic Token Interface Standard", Public Key Cryptography Standards PKCS#11-v2.20, June 2004.

[RFC4395] Hansen, T., Hardie, T., and L. Masinter, "Guidelines and Registration Procedures for New URI Schemes", BCP 35, RFC 4395, February 2006, <http://www.rfc-editor.org/info/rfc4395>.

[RFC6570] Gregorio, J., Fielding, R., Hadley, M., Nottingham, M., and D. Orchard, "URI Template", RFC 6570, March 2012, <http://www.rfc-editor.org/info/rfc6570>.

[UAX15] Davis, M., Ed. and K. Whistler, Ed., "Unicode Standard Annex #15: Unicode Normalization Forms", Version Unicode 7.0.0, June 2014, <http://unicode.org/reports/tr15/>.

Contributors

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

Authors' Addresses

Jan Pechanec
Oracle Corporation
4180 Network Circle
Santa Clara, CA 95054
United States

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

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

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