PKCS #11 Profiles Version 3.2

Committee Specification 01

14 November 2025

This stage:
https://docs.oasis-open.org/pkcs11/pkcs11-profiles/v3.2/cs01/pkcs11-profiles-v3.2-cs01.docx
(Authoritative)
https://docs.oasis-open.org/pkcs11/pkcs11-profiles/v3.2/cs01/pkcs11-profiles-v3.2-cs01.html
https://docs.oasis-open.org/pkcs11/pkcs11-profiles/v3.2/cs01/pkcs11-profiles-v3.2-cs01.pdf

Previous stage:
https://docs.oasis-open.org/pkcs11/pkcs11-profiles/v3.2/csd01/pkcs11-profiles-v3.2-csd01.docx
https://docs.oasis-open.org/pkcs11/pkcs11-profiles/v3.2/csd01/pkcs11-profiles-v3.2-csd01.html
https://docs.oasis-open.org/pkcs11/pkcs11-profiles/v3.2/csd01/pkcs11-profiles-v3.2-csd01.pdf

Latest stage:
https://docs.oasis-open.org/pkcs11/pkcs11-profiles/v3.2/pkcs11-profiles-v3.2.docx (Authoritative)
https://docs.oasis-open.org/pkcs11/pkcs11-profiles/v3.2/pkcs11-profiles-v3.2.html
https://docs.oasis-open.org/pkcs11/pkcs11-profiles/v3.2/pkcs11-profiles-v3.2.pdf

Technical Committee:
OASIS PKCS 11 TC

Chairs:
Robert Relyea (rrelyea@redhat.com), Red Hat
Greg Scott (greg.scott@cryptsoft.com), Cryptsoft Pty Ltd

Editor:
Tim Hudson (tjh@cryptsoft.com), Cryptsoft Pty Ltd

Additional artifacts:

This prose specification is one component of a Work Product that also includes:

- PKCS #11 test cases: https://docs.oasis-open.org/pkcs11/pkcs11-profiles/v3.2/cs01/test-cases/

Related work:

This specification replaces or supersedes:

- *PKCS #11 Profiles Version 3.1*. Edited by Tim Hudson. OASIS Standard. Latest stage: https://docs.oasis-open.org/pkcs11/pkcs11-profiles/v3.1/pkcs11-profiles-v3.1.html.

This specification is related to:

- *PKCS #11 Specification Version 3.2.* Edited by Dieter Bong and Greg Scott. Latest stage: https://docs.oasis-open.org/pkcs11/pkcs11-spec/v3.2/pkcs11-spec-v3.2.html.

Abstract:

This document defines data types, functions and other basic components of the PKCS #11 Cryptoki interface.

Status:

This document was last revised or approved by the OASIS PKCS 11 TC on the above date. The level of approval is also listed above. Check the "Latest stage" location noted above for possible later revisions of this document. Any other numbered Versions and other technical work produced by the Technical Committee (TC) are listed at https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=pkcs11#technical.

TC members should send comments on this document to the TC's email list. Others should send comments to the TC's public comment list, after subscribing to it by following the instructions at the "Send A Comment" button on the TC's web page at https://www.oasis-open.org/committees/pkcs11/.

This specification is provided under the RF on RAND Terms Mode of the OASIS IPR Policy, the mode chosen when the Technical Committee was established. For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights section of the TC's web page (https://www.oasis-open.org/committees/pkcs11/ipr.php).

Note that any machine-readable content (Computer Language Definitions) declared Normative for this Work Product is provided in separate plain text files. In the event of a discrepancy between any such plain text file and display content in the Work Product's prose narrative document(s), the content in the separate plain text file prevails.

Key words:

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] and [RFC8174] when, and only when, they appear in all capitals, as shown here.

Citation format:

When referencing this document, the following citation format should be used:

[PKCS11-Profiles-v3.2]

PKCS #11 Profiles Version 3.2. Edited by Tim Hudson. 15 April 2025. OASIS Committee Specification 01. https://docs.oasis-open.org/pkcs11/pkcs11-profiles/v3.2/cs01/pkcs11-profiles-v3.2-cs01.html. Latest stage: https://docs.oasis-open.org/pkcs11/pkcs11-profiles/v3.2/pkcs11-profiles-v3.2.html.

Notices:

Table of Contents

1 Introduction

This document intends to meet this OASIS requirement on conformance clauses for providers and consumers of cryptographic services via PKCS#11 ([PKCS11_Spec] Section 7 - PKCS#11 Implementation Conformance) through profiles that define the use of PKCS#11 data types, objects, functions and mechanisms within specific contexts of provider and consumer interaction. These profiles define a set of normative constraints for employing PKCS#11 within a particular environment or context of use. They may, optionally, require the use of specific PKCS#11 functionality or in other respects define the processing rules to be followed by profile actors.

For normative definition of the elements of PKCS#11 specified in these profiles, see the PKCS#11 Specification [PKCS11_Spec].

2 Profiles

This document defines a selected set of conformance clauses which form PKCS #11 Profiles. A profile may be standalone or may be specified in terms of changes relative to another profile.

The PKCS 11 TC also welcomes proposals for new profiles. PKCS 11 TC members are encouraged to submit these proposals to the PKCS 11 TC for consideration for inclusion in a future version of this TC-approved document.

2.1 Profile Requirements

The following items SHALL be addressed by each profile:

1. Specify the versions of the PKCS#11 specification that SHALL be supported if versions other than [PKCS11_Spec] are supported
2. Specify the list of additional data types that SHALL be supported
3. Specify the list of additional attributes that SHALL be supported
4. Specify the list of additional objects that SHALL be supported
5. Specify the list of additional functions that SHALL be supported
6. Specify the list of additional mechanisms that SHALL be supported
7. Specify any other requirements that SHALL be supported
8. Specify any mandatory test cases that SHALL be supported by conforming implementations
9. Specify optional test cases that MAY be supported by conforming implementations

Note: items may be specified either directly in a profile or by reference to other profiles. Where another profile is referenced as required, the combination of the requirements of all referenced required profiles (directly or indirectly) SHALL apply.

2.2 Guidelines for other Profiles

Any vendor or organization, such as other standards bodies, MAY create a PKCS#11 Profile and publish it.

1. The profile SHALL be publicly available.
2. The PKCS11 Technical Committee SHALL be formally advised of the availability of the profile and the location of the published profile.
3. The profile SHALL meet all the requirements of section 2.1
4. The PKCS11 Technical Committee SHOULD review the profile prior to final publication.

2.3 Defined Profile Identifiers

Profile objects (object class *CKO_PROFILE*) describe which PKCS #11 profiles a provider implements.

The *CKA_PROFILE_ID* attribute identifies a profile that the provider implements.

Attributes	Data Types	Meaning
CKA_PROFILE_ID	CK_PROFILE_ID	ID of the supported profile

The following table defines the CK_PROFILE_ID values:

Constant	Meaning
CKP_INVALID_ID	Invalid Profile
CKP_BASELINE_PROVIDER	Baseline Provider
CKP_EXTENDED_PROVIDER	Extended Provider

CKP_AUTHENTICATION_TOKEN	Authentication Token Provider or Consumer
CKP_PUBLIC_CERTIFICATES_TOKEN	Public Certificates Token Provider or Consumer
CKP_COMPLETE_PROVIDER	Complete Provider
CKP_HKDF_TLS_TOKEN	HKDF TLS Token
CKP_VENDOR_DEFINED	Vendor defined

3 Conformance Test Cases

The test cases define a sequence of PKCS#11 function calls with specified input and output parameters.

Each test case is provided in the XML format specified in PKCS#11 XML Representation (4) intended to be both human-readable and usable by automated tools.

Each test case has a unique label (the section name) which includes indication of mandatory (-M-) or optional (-O-) status and the specification version major and minor numbers as part of the identifier.

The test cases may depend on a specific configuration of a PKCS#11 provider and consumer and being configured in a manner consistent with the test case assumptions.

Where possible the flow of identifiers between tests, date values, and other dynamic items are indicated using symbolic identifiers – in actual request and response messages these dynamic values will be filled in with valid values.

Symbolic identifiers SHALL be of the form ${ParameterName}. Wherever a symbolic identifier occurs in a test case the implementation must replace it with a reasonable appearing datum of the expected type.

The symbolic identifier may reference return parameters or array or list items by index number. Array index numbers SHALL be of the form ${ParmeterName[ArrayIndex]} and the first element SHALL be indicated by index zero.

The symbolic identifier may reference elements nested within other elements. Nested references SHALL be of the form ${ParameterName.SubElement} and MAY also include an array index.

Note: the values for the returned items are illustrative. Actual values from a real consumer or provider MAY vary as specified in section 3.1.

3.1 Permitted Test Case Variations

Whilst the test cases provided in a Profile define the allowed call and return content, some inherent variations MAY occur and are permitted within a successfully completed test case.

Each test case MAY include allowed variations in the description of the test case in addition to the variations noted in this section.

Other variations not explicitly noted in this section SHALL be deemed non-conformant.

3.1.1 Variable Items

An implementation conformant to a Profile MAY vary the following values (expressed using the XML name for the items):

Provider specific information within the Info, SlotInfo and TokenInfo elements:

1. LibraryDescription
2. LibraryVersion
3. ManufacturerID
4. SlotDescription
5. HardwareVersion
6. FirmwareVersion
7. serialNumber
8. label
9. model
10. utcTime

Session specific information:

1. SlotID
2. Object

3. Session

Object specific information:

1. Object

Operation specific information:

1. Data
2. EncryptedData
3. RandomData

Attribute specific information:

1. VALUE
2. PUBLIC_EXPONENT
3. PRIVATE_EXPONENT
4. PRIME_1
5. PRIME_2
6. EXPONENT_1
7. EXPONENT_2
8. COEFFICIENT
9. PRIME
10. SUBPRIME
11. BASE
12. EC_POINT
13. UNIQUE_ID

3.1.2 Variable behavior

An implementation conformant to a Profile SHALL allow variation of the following behavior:

1. A test may omit the clean-up functions at the end of the test provided there is a separate mechanism to remove the created objects during testing.
2. A test may omit the test identifiers in various attributes if the consumer is unable to include them in calls.
3. The number of entries and order of entries in the list returned in the *C_GetSlotList*, *C_GetMechanismList*, and *C_GetInterfaceList* functions make vary, provided that at least one entry within the list matches the logical context of the test case.

4 PKCS#11 XML Representation

4.1 Normalizing Names

PKCS#11 parameter and structure field names SHALL be normalized to create a ‘CamelCase’ format that would be suitable to be used as a variable name in C/Java or an XML element name.

Hungarian notation type indicators are entirely omitted from names (i.e. *h, ph, ul, pul,* and *p* are omitted).

PKCS#11 function names are represented as-is (unchanged) as XML elements of the same name.

4.2 Omitted Items

PKCS#11 pointers for callback functions and reserved items are entirely omitted (i.e. *pApplication*, *pReserved*, *Notify* are not present).

Hungarian notation type indicators are entirely omitted from names (i.e. *h, ph, ul, pul,* and *p* are omitted).

4.3 Value Representation

The value for PKCS#11 binary (*CK_BYTE)* information SHALL be encoded as hexadecimal strings.

The value for PKCS#11 textual information (*CK_CHAR, CK_UTF8CHAR)* SHALL be encoded as hex strings.

The value for PKCS#11 numeric information SHALL be encoded as integers or as hexadecimal strings.

4.3.1 Enumerated Type Representation

Each PKCS#11 type value SHALL be represented in string/text form using the uppercase C macro name with the type prefix omitted. E.g. *CKR_OK* has a representation of “*OK*”.

4.3.2 Boolean Representation

Each PKCS#11 boolean value (*CK_BBOOL*) SHALL be represented in string/text form either as “true” (non-zero) or “false” (zero). No other representation SHALL be used.

4.3.3 Flag Type Representation

Each PKCS#11 flag value SHALL be represented using the uppercase C macro names with the type prefix omitted for each bit. If multiple bit flags are set then each SHALL be present separated by either a space (‘ ‘) or a pipe (‘|’) character.

4.3.4 Special Value Representation

For PKCS#11 CK_ULONG values which have special interpretation as CK_UNAVAILABLE_INFORMATION or CK_EFFECTIVELY_INFINITE the string values “UnavailableInformation” and “EffectivelyInfinite” SHOULD be used instead of the numeric values to improve readability. This approach is used in the CK_TOKEN_INFO structure for various count and length and size values.

4.3.5 Function Call and Return Representation

PKCS#11 function calls are represented as an XML element of the same name containing the input parameters each represented as XML elements and an XML element of the same name as the PKCS#11 function name with an XML element attribute named *rv* containing the return value. The XML element for the input parameters is always immediately followed by the XML element for the output results.

PKCS#11 parameters and structure members that are not arrays or lists are represented as XML elements with the value of the parameter or structure member contained within the XML element attribute *value*.

4.3.6 Array and List Representation

PKCS#11 parameters and structure members that are arrays or lists are represented as XML elements with the length of the array or list contained in XML element attribute *length* and the members of the array or list represented as nested XML elements unless an XML element attribute-based representation has been separately defined (e.g for *CK_ATTRIBUTE*).

PKCS#11 parameters and structure member elements that represent the count of arrays are omitted as input parameters as the lengths can be determined by a count of the number of XML elements within the call or return XML element within the element representing the PKCS#11 function call.

4.3.7 Determining Array or List Length

The PKCS#11 approach of passing in a NULL pointer value and using an input/output parameter to determine the required pointer buffer length for a subsequent call SHALL be encoded as request where the XML element for pointer has no specified value or length for the function call and the returned length is contained in the XML element attribute *length*.

4.3.8 Hexadecimal String Encoding

Hexadecimal strings SHALL NOT include any white space.

Hexadecimal strings SHALL use either uppercase ‘A’-‘F’ or lowercase ‘a’-‘f’ along with ‘0’ to ‘9’.

Numeric values represented as hexadecimal strings SHALL begin with ‘0x’.

Binary values represented as hexadecimal strings SHOULD omit the ‘0x’.

4.4 XML Root Element

XML documents representing a sequence of PKCS#11 function calls and returns SHALL have an XML root element of *PKCS11*.

4.5 XML Namespaces

If namespaces are necessary within a specific context, then each XML element SHALL use the following namespace:

urn:oasis:tc:pkcs11:xmlns

4.6 XML Element Encoding

For XML, each function call is represented as a sequence of two XML element with optional attributes.

The parameters to each call are represented as nested XML elements, and any structures used within those parameters are represented as nested XML elements within the nested XML elements.

The types of each parameter or structure element are fixed within the PKCS#11 specification and are not separately represented within the XML encoding. i.e. the types are inherently known by implementations and are fixed, matching the underlying C static type declaration.

4.6.1 Boolean

XML value uses [XML-SCHEMA] type xsd:Boolean. The value SHALL be FALSE, false, TRUE or true.

```
<TokenPresent value="false"/>
```

4.6.2 Text String

XML value uses [XML-SCHEMA] type xsd:string

```
<Pin value="12345678"/>
```

4.6.3 Byte String

XML value uses [XML-SCHEMA] type xsd:hexBinary

```
<EncryptedData value="8dce78ad"/>
```

4.6.4 Enumerated Type

XML value uses [XML-SCHEMA] type xsd:string and is either a hexadecimal string or the *Enumerated Type Representation* name. If an XSD with xsd:enumeration restriction is used to define valid values parsers should also accept any hexadecimal string in addition to the defined enumeration values to allow for user extensions and non-textual encoding parsers.

```
<Type value="AES_CBC"/>
<Type value="0x00001082"/>
<Type value="4426"/>
```

4.6.5 Function Call and Return

PKCS#11 function call and return SHALL be encoded as an XML element for the function call with any required parameters as nested XML elements, followed by an XML element for the function return with an XML element attribute of *rv* containing the return code from the function call encoded as an Enumerated Type and any output parameters as nested XML elements.

```
<C_Initialize/>
<C_Initialize rv="OK"/>
<C_GetSlotList>
  <TokenPresent value="false"/>
  <SlotList/>
</C_GetSlotList>
<C_GetSlotList rv="OK">
  <SlotList length="1"/>
</C_GetSlotList>
```

4.6.6 Attribute

PKCS#11 attributes (*CK_ATTRIBUTE*) SHALL be encoded as an XML element with an XML element attribute *type* containing the name of the PKCS#11 attribute and an XML element attribute *value* containing the value of the attribute. Where the PKCS#11 attribute has a specified type, the *value* SHALL be encoding using the encoding rules for that type of PKCS#11 value.

```
<Attribute type="CLASS" value="SECRET_KEY"/>
<Attribute type="KEY_TYPE" value="AES"/>
<Attribute type="LABEL" value="timing-key"/>
<Attribute type="TOKEN" value="TRUE"/>
<Attribute type="PRIVATE" value="TRUE"/>
<Attribute type="EXTRACTABLE" value="TRUE"/>
<Attribute type="SENSITIVE" value="TRUE"/>
<Attribute type="ENCRYPT" value="TRUE"/>
<Attribute type="DECRYPT" value="TRUE"/>
<Attribute type="VALUE_LEN" value="16"/>
```

4.6.7 XML Element Attributes

XML element attributes other than “type”, “value”, “length” and “rv” as defined in this specification SHALL not be used. All other PKCS#11 concepts are represented as XML elements and not XML element attributes.

5 Base Profiles

The following subsections describe currently-defined profiles related to the use of PKCS #11. The profiles define classes of PKCS #11 functionality to which an implementation can declare conformance.

5.1 Baseline Provider

A PKCS #11 provider makes cryptographic functionality available to a consuming application in terms of the PKCS #11 API.

This profile specifies the most basic functionality that would be expected of a conformant PKCS #11 provider – the ability to provide information about the capabilities of the cryptographic services provided.

An implementation conforms to this specification as a Baseline Provider if it meets the following conditions:

1. Supports the conditions required by the *PKCS#11 Provider Implementation Conformance* clauses [PKCS11_Spec]
2. Supports the following data types [PKCS11_Spec]:
   a. *CK_VERSION*
   b. *CK_INFO*
   c. *CK_SLOT_ID*
   d. *CK_SLOT_INFO*
   e. *CK_TOKEN_INFO*
   f. *CK_SESSION_HANDLE*
   g. *CK_USER_TYPE*
   h. *CK_SESSION_INFO*
   i. *CK_OBJECT_HANDLE*
   j. *CK_OBJECT_CLASS*
   k. *CK_ATTRIBUTE_TYPE*
   l. *CK_ATTRIBUTE*
   m. *CK_PROFILE_ID*
   n. *CK_RV*
   o. *CK_FUNCTION_LIST*
   p. *CK_INTERFACE*
   q. *CK_C_INITIALIZE_ARGS*
3. Supports the following attributes [PKCS11_Spec]:
   a. *CKA_CLASS*
   b. *CKA_TOKEN*
   c. *CKA_VALUE*
   d. *CKA_ID*
   e. *CKA_PRIVATE*
   f. *CKA_MODIFIABLE*
   g. *CKA_LABEL*
   h. *CKA_UNIQUE_IDENTIFIER*
   i. *CKA_PROFILE_ID*
4. Supports the following objects [PKCS11_Spec]:
   a. *CKO_PROFILE* with value *CKP_BASELINE_PROVIDER*
5. Supports the following functions [PKCS11_Spec]:
   a. *C_GetFunctionList*
   b. *C_GetInterfaceList*
   c. *C_GetInterface*
   d. *C_Initialize*
   e. *C_Finalize*
   f. *C_GetInfo*
   g. *C_GetSlotList*
   h. *C_GetSlotInfo*
   i. *C_GetTokenInfo*

    j. *C_OpenSession*
    k. *C_CloseSession*
    l. *C_GetSessionInfo*
    m. *C_FindObjectsInit*
    n. *C_FindObjects*
    o. *C_FindObjectsFinal*
    p. *C_GetAttributeValue*

6. Supports the following mechanisms:
    a. None specified
7. Supports *Error Handling* [PKCS11_Spec] for any supported object, function or mechanism
8. Optionally supports any clause within [PKCS11_Spec] that is not listed above
9. Optionally supports extensions outside the scope of this standard (e.g., vendor defined extensions, conformance clauses) that do not contradict any PKCS #11 requirements

5.1.1 Baseline Provider Mandatory Test Cases

5.1.1.1 BL-M-1-32

See test-cases/pkcs11-v3.2/mandatory/BL-M-1-32.xml

5.2 Complete Provider

A PKCS #11 provider makes cryptographic functionality available to a consuming application in terms of the PKCS #11 API.

This profile specifies the functionality that would be expected of a conformant PKCS #11 provider that implements the entire specification.

An implementation conforms to this specification as a Complete Provider if it meets the following conditions:

1. Supports the conditions required by the *PKCS#11 Provider Implementation Conformance* clauses [PKCS11_Spec]
2. Supports all data types [PKCS11_Spec]
3. Supports all attributes [PKCS11_Spec]
4. Supports all objects [PKCS11_Spec]
5. Supports all functions [PKCS11_Spec]
6. Supports all mechanisms [PKCS11_Spec] Section 6
7. Supports *Error Handling* [PKCS11_Spec]
8. Optionally supports extensions outside the scope of this standard (e.g., vendor defined extensions, conformance clauses) that do not contradict any PKCS #11 requirements

5.3 Extended Provider

This profile builds on the PKCS#11 Baseline Provider to add support for mechanism-based usage.

An implementation conforms to this specification as an Extended Provider if it meets the following conditions:

1. Supports the conditions required by the PKCS #11 conformance clauses ([PKCS11_Spec] Section 7 (PKCS#11 Implementation Conformance)
2. Supports the conditions required by the PKCS #11 Baseline Provider clauses section5.1.
3. Supports the following data types [PKCS11_Spec]:
    a. *CK_MECHANISM_TYPE*
    b. *CK_MECHANISM*
4. Supports the following attributes [PKCS11_Spec]:
    a. None specified
5. Supports the following objects [PKCS11_Spec]:
    a. *CKO_PROFILE* with value *CKP_EXTENDED_PROVIDER*
6. Supports the following functions [PKCS11_Spec]:

a. *C_GetMechanismList*
b. *C_GetMechanismInfo*
c. *C_Login*
d. *C_LoginUser*
e. *C_Logout*

7. Supports the following mechanisms:
   a. None specified
8. Supports *Error Handling* [PKCS11_Spec] for any supported object, function or mechanism
9. Optionally supports any clause within [PKCS11_Spec] that is not listed above
10. Optionally supports extensions outside the scope of this standard (e.g., vendor defined extensions, conformance clauses) that do not contradict any PKCS #11 requirements

5.3.1 Extended Provider Mandatory Test Cases

5.3.1.1 EXT-M-1-32

See test-cases/pkcs11-v3.2/mandatory/EXT-M-1-32.xml

5.4 Authentication Token

This profile builds on the PKCS #11 Baseline Provider and/or Baseline Consumer profiles to provide for use in the context of an authentication token.

An implementation conforms to this specification as an Authentication Token if it meets the following conditions:

1. If the implementation is a consumer then it SHALL support the conditions required by the PKCS #11 Baseline Consumer Clause (Section 5.7)
2. If the implementation is a provider then it SHALL support the conditions required by the PKCS #11 Baseline Provider Clause (Section 5.1)
3. Supports the following data types [PKCS11_Spec]:
   a. None specified
4. Supports the following attributes [PKCS11_Spec]:
   a. None specified
5. Supports the following objects [PKCS11_Spec]:
   a. *CKO_PRIVATE_KEY*
   b. *CKO_PUBLIC_KEY*
   c. *CKO_PROFILE* with value *CKP_AUTHENTICATION_TOKEN*
6. Supports the following functions [PKCS11_Spec]:
   a. *C_Login*
   b. *C_LoginUser*
   c. *C_Logout*
   d. *C_SignInit*
   e. *C_Sign* and/or *C_SignUpdate* and *C_SignFinal*
7. Supports the following mechanisms:
   a. None specified
8. Supports *Error Handling* [PKCS11_Spec] for any supported object, function or mechanism
9. Optionally supports any clause within [PKCS11_Spec] that is not listed above
10. Optionally supports extensions outside the scope of this standard (e.g., vendor defined extensions, conformance clauses) that do not contradict any PKCS #11 requirements.

5.4.1 Authentication Token Provider Mandatory Test Cases

5.4.1.1 AUTH-M-1-32

See test-cases/pkcs11-v3.2/mandatory/AUTH-M-1-32.xml

5.5 Public Certificates Token

This profile builds on the PKCS #11 Baseline Provider and/or Baseline Consumer profiles to provide for use in the context of a public certificates token.

An implementation conforms to this specification as a Public Certificates Token if it meets the following conditions:

1. If the implementation is a consumer then it SHALL support the conditions required by the PKCS #11 Baseline Consumer Clause (Section 5.7)
2. If the implementation is a provider then it SHALL support the conditions required by the PKCS #11 Baseline Provider Clause (Section 5.1)
3. Supports the following data types [PKCS11_Spec]:
    a. None specified
4. Supports the following attributes [PKCS11_Spec]:
    a. None specified
5. Supports the following objects [PKCS11_Spec]:
    a. CKO_CERTIFICATE
    b. *CKO_PROFILE* with value *CKP_PUBLIC_CERTIFICATES_TOKEN*
6. Supports the following functions [PKCS11_Spec]:
    a. None specified
7. Supports the following mechanisms [PKCS11_Spec]:
    a. None specified
8. Supports the following object location requirements:
    a. All certificates are publicly readable, able to be found on the token without a login having been performed
    b. All certificates for which a matching private key also exists on the token must have a matching *CKA_ID* attribute for the certificate and private key
    c. One or more of the following conditions must be met:
        i. The matching private key for a certificate can be found via *C_FindObjects* using the matching *CKA_ID* value without a login having been performed;
        ii. The matching public key for a certificate can be found via *C_FindObjects* using the matching *CKA_ID* value without a login having been performed
9. Supports Error Handling [PKCS11_Spec] for any supported object, function or mechanism
10. Optionally supports any clause within [PKCS11_Spec] that is not listed above
11. Optionally supports extensions outside the scope of this standard (e.g., vendor defined extensions, conformance clauses) that do not contradict any PKCS #11 requirements.

5.5.1 Public Certificates Token Provider Mandatory Test Cases

5.5.1.1 CERT-M-1-32

See test-cases/pkcs11-v3.2/mandatory/CERT-M-1-32.xml

5.6 HKDF TLS Token

This profile builds on the PKCS #11 Baseline Provider and/or Baseline Consumer profiles to provide for use in the context of TLS 1.3 connections using the CKM_HKDF_DERIVE_DATA mechanism.

An implementation conforms to this specification as an HKDF TLS Token if it meets the following conditions:

1. If the implementation is a consumer then it SHALL support the conditions required by the PKCS #11 Baseline Consumer Clause (Section 5.7)
2. If the implementation is a provider then it SHALL support the conditions required by the PKCS #11 Baseline Provider Clause (Section 5.1)
3. Supports the following data types [PKCS11_Spec]:
    b. CK_HKDF_PARAMS
4. Supports the following attributes [PKCS11_Spec]:

   a. None specified
5. Supports the following objects [PKCS11_Spec]:
   a. *CKO_DATA*
   b. *CKO_SECRET_KEY*
   c. *CKO_PROFILE* with value *CKP_HKDF_TLS_TOKEN*
6. Supports the following functions [PKCS11_Spec]:
   a. *C_DeriveKey*
7. Supports the following mechanisms:
   a. CKM_HKDF_DATA

      A conformant provider SHALL not reject derive requests based on the pInfo value if the following pInfo values are given:

      i. The string L1,L2,"tls iv",0 (L1, L2, 0x74, 0x6c, 0x73, 0x20, 0x69, 0x76, 0x00) where L1 is the most significant byte of CKA_VALUE_LEN and L2 is the least significant byte of CKA_VALUE_LEN.
      ii. The string L1,L2,"tls quic iv",0 (L1, L2, 0x74, 0x6c, 0x73, 0x20, 0x71, 0x75, 0x69, 0x63, 0x20, 0x69, 0x76, 0x00) where L1 is the most significant byte of CKA_VALUE_LEN and L2 is the least significant byte of CKA_VALUE_LEN.

      A conformant provider MAY accept other values for pInfo.

8. Supports *Error Handling* [PKCS11_Spec] for any supported object, function or mechanism
9. Optionally supports any clause within [PKCS11_Spec] that is not listed above
10. Optionally supports extensions outside the scope of this standard (e.g., vendor defined extensions, conformance clauses) that do not contradict any PKCS #11 requirements.

5.7 Baseline Consumer

A PKCS #11 consumer calls a PKCS #11 provider implementation of the PKCS #11 API in order to use the cryptographic functionality from that provider.

This profile specifies the most basic functionality that would be expected of a conformant PKCS #11 consumer – the ability to consume information via the cryptographic services offered by a provider.

An implementation conforms to this specification as a Baseline Consumer if it meets the following conditions:

1. Supports the conditions required by the *PKCS#11 Consumer Implementation Conformance* clauses [PKCS11_Spec]
2. Supports the following data types [PKCS11_Spec]:
   a. *CK_VERSION*
   b. *CK_INFO*
   c. *CK_SLOT_ID*
   d. *CK_SLOT_INFO*
   e. *CK_TOKEN_INFO*
   f. *CK_SESSION_HANDLE*
   g. *CK_USER_TYPE*
   h. *CK_SESSION_INFO*
   i. *CK_OBJECT_HANDLE*
   j. *CK_OBJECT_CLASS*
   k. *CK_ATTRIBUTE_TYPE*
   l. *CK_ATTRIBUTE*
   m. *CK_RV*
   n. *CK_FUNCTION_LIST*
   o. *CK_C_INITIALIZE_ARGS*
   p. *CK_INTERFACE (if C_GetInterfaceList and C_GetInterface is supported)*
3. Supports the following attributes [PKCS11_Spec]:
   a. *CKA_CLASS*

    b. *CKA_VALUE*

4. Supports the following objects:
    a. None specified
5. Supports the following functions [PKCS11_Spec]:
    a. *C_GetFunctionList* or *C_GetInterfaceList* and *C_GetInterface*
    b. *C_Initialize*
    c. *C_Finalize*
    d. *C_GetInfo*
    e. *C_GetSlotList*
    f. *C_GetSlotInfo*
    g. *C_GetTokenInfo*
    h. *C_OpenSession*
    i. *C_CloseSession*
6. Supports the following mechanisms:
    a. None specified
7. Supports *Error Handling* [PKCS11_Spec] for any supported object, function or mechanism
8. Optionally supports any clause within [PKCS11_Spec] that is not listed above
9. Optionally supports extensions outside the scope of this standard (e.g., vendor defined extensions, conformance clauses) that do not contradict any PKCS #11 requirements

5.8 Extended Consumer

This profile builds on the PKCS#11 Baseline Consumer profile to add support for mechanism-based usage.

An implementation conforms to this specification as an Extended Consumer if it meets the following conditions:

1. Supports the conditions required by the PKCS11 conformance clauses ([PKCS11_Spec] Section 7 (PKCS#11 Implementation Conformance)
2. Supports the conditions required by the PKCS11 Baseline Consumer clauses section 5.7
3. Supports the following data types [PKCS11_Spec]:
    a. *CK_MECHANISM_TYPE*
    b. *CK_MECHANISM*
4. Supports the following attributes [PKCS11_Spec]:
    a. None specified
5. Supports the following objects [PKCS11_Spec]:
    a. None specified
6. Supports the following functions [PKCS11_Spec]:
    a. *C_GetMechanismList*
    b. *C_GetMechanismInfo*
7. Supports the following mechanisms:
    a. None specified
8. Supports *Error Handling* [PKCS11_Spec] for any supported object, function or mechanism
9. Optionally supports any clause within [PKCS11_Spec] that is not listed above
10. Optionally supports extensions outside the scope of this standard (e.g., vendor defined extensions, conformance clauses) that do not contradict any PKCS #11 requirements

6 Conformance

The baseline provider and consumer profiles provide the most basic functionality that is expected of a conformant PKCS#11 consumer or provider. The complete provider profile defines a PKCS#11 provider that implements the entire specification. A PKCS#11 implementation conformant to this specification (the PKCS#11 Profiles) SHALL meet all the conditions documented in one or more of the following sections.

6.1 Baseline Provider Profile Conformance

PKCS#11 provider implementations conformant to this profile:

1. SHALL support [PKCS11_Spec];
2. SHALL support the Baseline Provider conditions (5.1) and;
3. SHALL support one or more of the Baseline Provider Mandatory Test Cases (5.1.1).

6.2 Complete Provider Profile Conformance

PKCS#11 provider implementations conformant to this profile:

1. SHALL support [PKCS11_Spec];
2. SHALL support the Complete Provider conditions (5.2) and;
3. SHALL support all of the provider conformance clauses contained within Conformance (6).

6.3 Extended Provider Profile Conformance

PKCS#11 provider implementations conformant to this profile:

1. SHALL support [PKCS11_Spec];
2. SHALL support the Extended Provider conditions (5.3) and;
3. SHALL support one or more of the Extended Provider Mandatory Test Cases (5.3.1).

6.4 Authentication Token Provider Profile Conformance

PKCS#11 provider implementations conformant to this profile:

1. SHALL support [PKCS11_Spec];
2. SHALL support the Authentication Token conditions (5.4) and;
3. SHALL support all of the Authentication Token Provider Mandatory Test Cases (5.4.1).

6.5 Public Certificates Token Provider Profile Conformance

PKCS#11 provider implementations conformant to this profile:

1. SHALL support [PKCS11_Spec];
2. SHALL support the Public Certificates Token conditions (5.5) and;
3. SHALL support all of the Public Certificates Token Provider Mandatory Test Cases (5.5.1).

6.6 HKDF TLS Token Provider Profile Conformance

PKCS#11 provider implementations conformant to this profile:

1. SHALL support [PKCS11_Spec];
2. SHALL support the HKDF TLS Token conditions (5.6).

6.7 Baseline Consumer Profile Conformance

PKCS#11 consumer implementations conformant to this profile:

1. SHALL support [PKCS11_Spec]; and
2. SHALL support the Baseline Consumer conditions (5.7).

6.8 Authentication Token Consumer Profile Conformance

PKCS#11 provider implementations conformant to this profile:

1. SHALL support [PKCS11_Spec]; and
2. SHALL support the Authentication Token conditions (5.4)

6.9 Public Certificates Token Consumer Profile Conformance

PKCS#11 provider implementations conformant to this profile:

1. SHALL support [PKCS11_Spec]; and
2. SHALL support the Public Certificates Token conditions (5.5)

7 PKCS #11 Implementation Conformance

7.1 PKCS#11 Consumer Implementation Conformance

An implementation is a conforming PKCS#11 Consumer if the implementation meets the conditions specified in one or more consumer profiles specified in **[PKCS11-Prof]**.

A PKCS#11 consumer implementation SHALL be a conforming PKCS#11 Consumer.

If a PKCS#11 consumer implementation claims support for a particular consumer profile, then the implementation SHALL conform to all normative statements within the clauses specified for that profile and for any subclauses to each of those clauses.

7.2 PKCS#11 Provider Implementation Conformance

An implementation is a conforming PKCS#11 Provider if the implementation meets the conditions specified in one or more provider profiles specified in **[PKCS11-Prof]**.

A PKCS#11 provider implementation SHALL be a conforming PKCS#11 Provider.

Appendix A. References

This appendix contains the normative and informative references that are used in this document.

While any hyperlinks included in this appendix were valid at the time of publication, OASIS cannot guarantee their long-term validity.

A.1 Normative References

The following documents are referenced in such a way that some or all of their content constitutes requirements of this document.

[PKCS11_Spec]

PKCS #11 Specification Version 3.2. Edited by Dieter Bong and Greg Scott. Latest stage: https://docs.oasis-open.org/pkcs11/pkcs11-spec/v3.2/pkcs11-spec-v3.2.html.

[RFC2119]

Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC8174]

Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[XML]

Bray, Tim, et.al. eds, Extensible Markup Language (XML) 1.0 (Fifth Edition), W3C Recommendation 26 November 2008,
<http://www.w3.org/TR/2008/REC-xml-20081126>.

A.2 Informative References

The following referenced documents are not required for the application of this document but may assist the reader with regard to a particular subject area.

[RFC3552]

Rescorla, E. and B. Korver, "Guidelines for Writing RFC Text on Security Considerations", BCP 72, RFC 3552, DOI 10.17487/RFC3552, July 2003, <https://www.rfc-editor.org/info/rfc3552>.

[XML-SCHEMA]

Paul V. Biron, Ashok Malhotra, XML Schema Part 2: Datatypes Second Edition, W3C Recommendation 26 November 2008, <https://www.w3.org/TR/2004/REC-xmlschema-2-20041028>.

Appendix B. Acknowledgments

B.1 Special Thanks

Substantial contributions to this document from the following individuals are gratefully acknowledged:

Ms. Dina Kurktchi-Nimeh, Oracle

B.2 Participants

The following individuals were members of this Technical Committee during the creation of this document and their contributions are gratefully acknowledged:

Dr. Warren Armstrong, QuintessenceLabs Pty Ltd.

Anthony Berglas, Cryptsoft Pty Ltd.

Mr. Dieter Bong, Utimaco IS GmbH

Hamish Cameron, nCipher

Kenli Chong, QuintessenceLabs Pty Ltd.

Mr. Justin Corlett, Cryptsoft Pty Ltd.

Mr. Tony Cox, TC Logic/Tony Cox

Ms. Valerie Bubb Fenwick, Apple

Ms. Susan Gleeson, Oracle

Tim Hudson, Cryptsoft Pty Ltd. | OpenSSL Software Services Inc.

Mr. Gershon Janssen, Reideate

Mr. Jakup Jelen, Red Hat

Mr. Darren Johnson, THALES

Sun-Ho Lee, MDS Intelligence Inc

John Leiseboer, QuintessenceLabs Pty Ltd.

Mr. John Leser, Oracle

Scott Leubner, THALES

Scott Marshall, Cryptsoft Pty Ltd.

Dr. Michael Markowitz, Information Security Corporation

Mr. Darren Moffat, Oracle

Mr. Tim Ober, THALES

Dr. Florian Poppa, QuintessenceLabs Pty Ltd.

Mr. Robert Relyea, Red Hat

Mr. Jonathan Schulze-Hewett, Information Security Corporation

Mr. Greg Scott, Cryptsoft Pty Ltd.

Mr. Martin Shannon, QuintessenceLabs Pty Ltd.

Mr. Oscar So, Individual

Simo Sorce, Red Hat

Mr. Manish Upasani, Utimaco IS GmbH

Ms. Magda Zdunkiewicz, Cryptsoft Pty Ltd.

Appendix C. Revision History

Revision	Date	Editor	Changes Made
WD01	23 Sep-2023	Tim Hudson	Initial Draft
WD02	16 Jul 2024	Tim Hudson	Updated XML encoding approach to avoid use of XML element attributes for structure fields to improve readability of the XML.
WD03	31 Jul 2024	Tim Hudson	Additional typos and cross reference errors and formatting corrected based on review feedback.
WD04	15 Apr 2025	Dieter Bong	Updated link to [PKCS11_Spec]; updated Appendix B Acknowledgements

Appendix D. Notices

As stated in the OASIS IPR Policy, the following three paragraphs in brackets apply to OASIS Standards Final Deliverable documents (Committee Specifications, OASIS Standards, or Approved Errata).

[OASIS requests that any OASIS Party or any other party that believes it has patent claims that would necessarily be infringed by implementations of this OASIS Standards Final Deliverable, to notify OASIS TC Administrator and provide an indication of its willingness to grant patent licenses to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this deliverable.]

[OASIS invites any party to contact the OASIS TC Administrator if it is aware of a claim of ownership of any patent claims that would necessarily be infringed by implementations of this OASIS Standards Final Deliverable by a patent holder that is not willing to provide a license to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this OASIS Standards Final Deliverable. OASIS may include such claims on its website, but disclaims any obligation to do so.]

[OASIS takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this OASIS Standards Final Deliverable or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on OASIS' procedures with respect to rights in any document or deliverable produced by an OASIS Technical Committee can be found on the OASIS website. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this OASIS Standards Final Deliverable, can be obtained from the OASIS TC Administrator. OASIS makes no representation that any information or list of intellectual property rights will at any time be complete, or that any claims in such list are, in fact, Essential Claims.]

The name "OASIS" is a trademark of OASIS, the owner and developer of this document, and should be used only to refer to the organization and its official outputs. OASIS welcomes reference to, and implementation and use of, documents, while reserving the right to enforce its marks against misleading uses. Please see https://www.oasis-open.org/policies-guidelines/trademark/ for above guidance.