TECHNICAL SPECIFICATION

ISO/IEC TS 23078-2

First edition
2020-09

Information technology — Specification of DRM technology for digital publications —

Part 2: User key-based protection

Reference number
ISO/IEC TS 23078-2:2020(E)

Contents

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents) or the IEC list of patent declarations received (see http://patents.iec.ch).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT), see www.iso.org/iso/foreword.html.

This document was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 34, *Document description and processing languages*.

A list of all parts in the ISO/IEC TS 23078 series can be found on the ISO website.

Any feedback or questions on this document should be directed to the user’s national standards body. A complete listing of these bodies can be found at www.iso.org/members.html.

Introduction

Ever since ebooks have grown in popularity, copyright protection has been an important issue for authors and publishers.

While the distribution of ebooks around the world is mostly based on the open EPUB standard, most ebook retailers are using proprietary technologies to enforce usage constraints on digital publications in order to impede oversharing of copyrighted content. The high level of interoperability and accessibility gained by the use of a standard publishing format is therefore cancelled by the use of proprietary and closed technologies: ebooks are only readable on specific devices of software applications (a retailer "lock-in" syndrome), cannot be accessed anymore if the ebook distributor which protected the publication goes out of business or if the DRM technology evolves drastically. As a result, users are deprived of any control over their ebooks.

Requirements related to security levels differ depending on which part of the digital publishing market is addressed. In many situations, publishers require a solution which technically enforces the digital rights they provide to their users; most publishers are happy to adopt a DRM solution which guarantees an easy transfer of publications between devices, a certain level of fair-use and provides permanent access to the publications acquired by their customers.

This is where this document comes into play.

Information technology — Specification of DRM technology for digital publications —

Part 2: User key-based protection

1 Scope

This document defines a technical solution for encrypting resources in digital publications (especially EPUB) and for securely delivering decryption keys to reading systems, included in licenses tailored to specific users. It also defines a simple passphrase-based authentication method for reading systems to verify the license and access the encrypted resources of such digital publications.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

EPUB Open Container Format (OCF) 3.2, W3C, available at https://www.w3.org/publishing/epub32/epub-ocf

ISO 8601-1, *Date and time — Representations for information interchange — Part 1: Basic rules*

ISO/IEC 8824-1, *Information technology — Abstract Syntax Notation One (ASN.1): Specification of basic notation — Part 1:*

RFC 4627, The application/json Media Type for JavaScript Object Notation (JSON), The Internet Society, available at https://www.ietf.org/rfc/rfc4627

RFC 5280, Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, Network Working Group, available at https://tools.ietf.org/html/rfc5280

RFC 7807, Problem Details for HTTP APIs, The Internet Engineering Task Force, available at https://tools.ietf.org/html/rfc7807

3 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

— ISO Online browsing platform: available at https://www.iso.org/obp

— IEC Electropedia: available at http://www.electropedia.org/

3.1
codec content type
content type that has intrinsic binary format qualities

EXAMPLE Such as video and audio media type.

Note 1 to entry: It is already designed for optimum compression or provides optimized streaming capabilities.

3.2
content key
symmetric key used to encrypt and decrypt *publication resources* (3.15)

3.3
encryption profile
set of encryption algorithms used in a specific *protected publication* (3.10) and associated *license document* (3.6)

3.4
container
EPUB container
zip-based packaging and distribution format for *EPUB publications* (3.13)

[SOURCE: EPUB OCF 3.2, clause 4]

3.5
license authority
entity which delivers *provider certificates* (3.12) to *content providers* (3.11)

3.6
license document
document that contains references to the various keys, links to related external resources, rights and restrictions that are applied to *protected publication* (3.10), and *user* (3.18) information

3.7
licensed content protection
LCP
Readium LCP
DRM technology published by the Readium Foundation

3.8
non-codec content type
content type that benefits from compression due to the nature of its internal data structure

EXAMPLE Such as a file format based on character strings (for example HTML, CSS, etc.)

3.9
package document
publication resource (3.15) carrying meta information about an *EPUB publication* (3.13)

3.10
protected publication
LCP-protected publication
publication (3.13) in which *resources* (3.15) have been encrypted according to this document

3.11
provider
content provider
entity that delivers LCP licenses for *protected publications* (3.10) to *users* (3.18)

3.12
provider certificate
certificate that is included in the *license document* (3.6) to identify the *content provider* (3.11) and validate the signature of the license document

3.13
publication
EPUB publication
logical document entity consisting of a set of interrelated *resources* (3.15) and packaged in an *EPUB container* (3.4)

[SOURCE: EPUB Content Documents 3.2]

3.14
reading system
system that processes *EPUB publications* (3.13) and presents them to *users* (3.18)

3.15
resource
publication resource
content or instructions that contribute to the logic and rendering of an *EPUB publication* (3.13)

3.16
root certificate
certificate possessed by the *license authority* (3.5) and embedded in each EPUB *reading system* (3.14) in order to confirm that the *provider certificate* (3.12) is valid

3.17
status document
license status document
document that contains the current status and possible interactions with a *license document* (3.6), along with historical information

3.18
user
individual that consumes an *EPUB publication* (3.13) using an EPUB *reading system* (3.14)

3.19
user key
hash value of the *user passphrase* (3.20), used to decrypt the *content key* (3.2) and any encrypted *user* (3.18) information embedded in a *license document* (3.6)

3.20
user passphrase
string of text entered by the *user* (3.18) for obtaining access to the *protected publication* (3.10)

4 Abbreviated terms

DRM	digital rights management
IANA	Internet Assigned Number Authority

5 Overview

5.1 General

In order to deliver a publication to users without risk of indiscriminate redistribution, most publication resources are encrypted and a license document is generated.

The license document can be transmitted outside an EPUB container or be embedded inside it. Following the EPUB OCF 3.2 specification, META-INF/encryption.xml identifies all encrypted publication resources and points to the content key needed to decrypt them. This content key is located inside the license document and is itself encrypted using the user key. The user key is generated by calculating a hash of a user passphrase. It is used to decrypt the content key, which in turn is used to decrypt the publication resources.

The license document may also contain information about which rights are conveyed to the user and which are not, and information identifying the user and links to external resources. Rights information may include things like the time for which the license is valid, whether the book may be printed or copied, etc. Finally, the license document always includes a digital signature to prevent modification of any of its components.

Figure 1 shows the relationships among the various components of LCP.

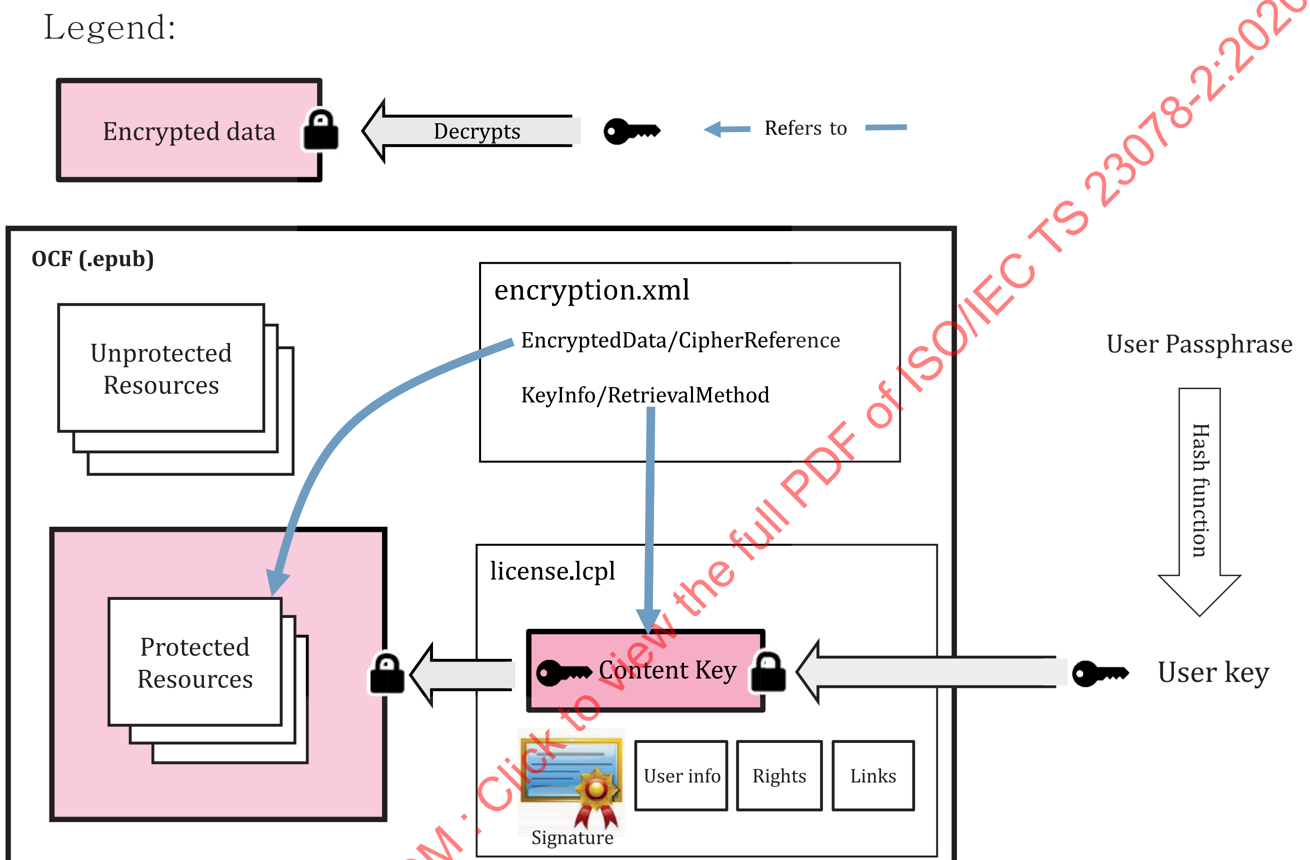

Figure 1 — Protected publication with a license document

5.2 Protecting the publication

To protect a publication, a content provider follows these steps.

a) Generate a unique content key for the publication.

b) Store this content key for future use in licensing the publication.

c) Encrypt each protected resource using that key, after compression if applicable.

d) Add these protected resources to the container, replacing unprotected versions.

e) Create a META-INF/encryption.xml document (as described in 9.3) which includes an EncryptedData element for each protected resource, that contains:

1) an EncryptionMethod element that lists the algorithm used;

2) a KeyInfo element with a RetrievalMethod child that points to the content key in the license document;

3) a CipherData element that identifies the protected resource.

f) Add META-INF/encryption.xml to the container.

The publication is now protected (i.e., has become a protected publication) and is ready for licensing to one or more users.

5.3 Licensing the publication

After a user requests a protected publication, the following steps are followed by the content provider to license the protected publication.

a) Generate the user key by hashing the user passphrase (as described in 6.4.2). It is assumed that the user and associated user passphrase are already known to the provider.

b) Encrypt the content key for the protected publication using the user key.

c) Create a license document (META-INF/license.lcpl) with the following contents:

1) a unique ID for this license;

2) the date the license was issued;

3) the URI that identifies the content provider;

4) the encrypted content key;

5) information relative to the user passphrase and user key;

6) links to additional information stored outside of the protected publication and license document (optional);

7) information on specific rights being granted to the user (optional);

8) information identifying the user (optional); some of the fields may be encrypted using the user key.

d) Generate a digital signature for the license document data and add it to the license document.

There are then two different methods to deliver the license document and protected publication to the user.

— **License document included inside protected publication:** The content provider adds the license document to the protected publication's container and delivers this to the user.

— **License document delivered separately:** The content provider includes a link from the license document to the protected publication, and then delivers just the license document to the user. The reading system processing the license document retrieves the protected publication and add the license document to the container of this protected publication.

Whichever method is used, the reading system is presented with an EPUB container that includes the protected publication and the license document.

5.4 Reading the publication

In order to decrypt and render a protected publication, the user's reading system follows these steps.

a) Verify the signature for the license document.

b) Get the user key (if already stored) or generate it by hashing the user passphrase.

c) Decrypt the content key using the user key.

d) Decrypt the protected resources using the content key.

6 License document

6.1 General

This clause defines the license document's syntax, its location in the container, its media type, file extension and processing model.

While META-INF/encryption.xml describes how the resources are encrypted and where the encrypted content key is located, every other relevant information for LCP is stored in the license document.

A.1 shows an example of a license document.

6.2 Content conformance

A license document shall meet all of the following criteria:

Document properties:

— It shall meet the conformance constraints for JSON documents as defined in RFC 4627.

— It shall be encoded using UTF-8.

File properties:

— Its filename shall use the file extension .lcpl.

— Its MIME media type shall be application/vnd.readium.lcp.license.v1.0+json.

— Its location in the container shall be META-INF/license.lcpl.

6.3 License information

6.3.1 General

The license document shall contain id, issued, provider, encryption, links and signature objects and may contain updated, rights and user objects as defined in Table 1.

Table 1 — Objects list of license document

Name	Value/Object	Format/data type	Required?
id	Unique identifier for the license	String	Yes
issued	Date and time of first issue of the license	ISO 8601-1 date and time	Yes
updated	Date and time of last update of the license	ISO 8601-1 date and time	No
provider	Identifier for the content provider	URI	Yes
encryption	encryption object	Object as defined in 6.3.2	Yes
links	links object	Object as defined in 6.3.3	Yes
rights	rights object	Object as defined in 6.3.4	No
user	user object	Object as defined in 6.3.5	No
signature	signature object	Object as defined in 6.3.6	Yes

The date and time format shall follow the rules defined in ISO 8601-1.

6.3.2 Encryption (transmitting keys)

6.3.2.1 General

To transmit keys, the encryption object shall contain profile, content_key and user_key objects.

6.3.2.2 Profile

The encryption/profile object shall contain the value defined in Table 2.

Table 2 — Profile information in encryption

Name	Value	Format/data type
profile	Identifier for the encryption profile used by this LCP-protected publication	URI

6.3.2.3 Content key

The encryption/content_key object contains the content key (encrypted using the user key) used to encrypt the publication resources. It shall contain the name/value pairs described in Table 3.

Table 3 — Content key information in *encryption*

Name	Value	Format/data type
encrypted_value	Encrypted content key	Base 64 encoded octet sequence
algorithm	Algorithm used to encrypt the content key, identified using the URIs defined in W3C XML Encryption. This shall match the content key encryption algorithm named in the encryption profile identified in encryption/profile.	URI

6.3.2.4 User key

The encryption/user_key object contains information regarding the user key used to encrypt the content key. It shall contain the name/value pairs defined in Table 4.

Table 4 — User key information in *encryption*

Name	Value	Format/data type
text_hint	Hint to be displayed to the user to help them remember the user passphrase	String
algorithm	Algorithm used to generate the user key from the user passphrase. This URI shall match the user passphrase hash algorithm specified in the encryption profile identified in encryption/profile.	URI
key_check	Value of the license document's id field encrypted using the user key and the same algorithm identified for content key encryption in encryption/content_key/algorithm. This is used to verify that the reading system handles the correct user key.	Base 64 encoded octet sequence

EXAMPLE Encryption information for a license document that uses the basic encryption profile for LCP 1.0.

```
{
  "id": "ef15e740-697f-11e3-949a-0800200c9a66",
  "issued": "2013-11-04T01:08:15+01:00",
  "updated": "2014-02-21T09:44:17+01:00",
  "provider": "https://www.imaginaryebookretailer.com",
  "encryption": {
       "profile": "http://readium.org/lcp/basic-profile",
       "content_key": {
```

```
       "encrypted_value": "/k8RpXqf4E2WEunCp76E8PjhS051NXwAXeTD1ioazYxCRGvHLAck/KQ3cCh5JxD
mCK0nRLyAxs1X0aA3z55boQ==",
           "algorithm": "http://www.w3.org/2001/04/xmlenc#aes256-cbc"
           },
          "user_key": {
               "text_hint": "Enter your email address",
               "algorithm": "http://www.w3.org/2001/04/xmlenc#sha256",
               "key_check": "jJEjUDipHK3OjGt6kFq7dcOLZuicQFUYwQ+TYkAIWKm6Xv6kpHFhF7LOkUK/
Owww"
          }
  },
  "links": "...",
  "rights": "...",
  "signature": "..."
}
```

6.3.3 Links (pointing to external resources)

6.3.3.1 General

To specify external resources, the license document shall contain a links object. This is used to associate the license document with resources that are not locally available.

Each link object nested in links contains the link URI, a link relationship and an optional set of other link properties.

6.3.3.2 Link object

Each link object supports the names/values defined in Table 5.

Table 5 — Object list in link

Name	Value	Format/data type	Required?
href	Location of the linked resource	URI or URI Template	Yes
rel	Link relationship to the document	List of well-known relation values, URIs for extensions as defined in 6.3.3.3	Yes
title	Title of the link	String	No
type	Expected MIME media type value for the external resources	MIME media type	No, but highly recommended
templated	Indicates that the href is a URI Template	Boolean	No, default value is "false"
profile	Expected profile used to identify the external resource	URI	No
length	Content length in octets	Integer	No
hash	SHA-256 hash of the resource	Base 64 encoded octet sequence	No

Templated URIs follow the RFC 6570 specification.

6.3.3.3 Link relationships

Table 6 introduces the link relationships for each link object which is used for value of rel.

Table 6 — Link relationships of link

Relation	Semantics	Required?
hint	Location where a reading system can redirect a user looking for additional information about the user passphrase	Yes
publication	Location where the publication associated with the license document can be downloaded	Yes
self	As defined in the IANA registry of link relations: "Conveys an identifier for the link's context"	No
support	Support resources for the user (either a website, an email or a telephone number)	No
status	Location of the status document associated with the license document	No

In addition to these link relationships, this document introduces the LCP Link Relations Registry, in which all link relations defined in this document are referenced.

Link relationships may also be extended for vendor-specific applications. Such links shall use a URI instead of a string to identify their link relationships.

The representation of the URI mentioned in the status link shall be a valid status document.

EXAMPLE A license document points to a publication, contains the location of a hint about its user passphrase, several support links and uses an extension to provide authentication.

```
{
  "id": "ef15e740-697f-11e3-949a-0800200c9a66",
  "issued": "2013-11-04T01:08:15+01:00",
  "updated": "2014-02-21T09:44:17+01:00",
  "provider": "https://www.imaginaryebookretailer.com",
  "encryption": "...",
  "links": [
    { "rel": "publication",
      "href": "https://www.example.com/file.epub",
      "type": "application/epub+zip",
      "length": "264929",
      "hash": "8b752f93e5e73a3efff1c706c1c2e267dffc6ec01c382cbe2a6ca9bd57cc8378"
    },
    { "rel": "hint",
      "href": "https://www.example.com/passphraseHint?user_id=1234",
      "type": "text/html"
    },
    { "rel": "support",
      "href": "mailto:support@example.org"
    },
    { "rel": "support",
      "href": "https://example.com/support",
      "type": "text/html"
    },
    { "rel": "www.imaginaryebookretailer.com/lcp/rel/authentication",
      "href": "https://www.example.com/authenticateMe",
      "title": "Authentication service",
      "type": "application/vnd.myextension.authentication+json"
    }
  ],
  "rights": "...",
  "signature": "..."
}
```

6.3.4 Rights (identifying rights and restrictions)

To identify rights and restrictions, the license document may also express a series of rights using the rights object. The rights object may include the fields defined in Table 7.

Table 7 — Object list in *rights*

Name	Value	Format/data type	Default
print	Maximum number of pages that can be printed over the lifetime of the license	Unsigned integer	Unlimited
copy	Maximum number of characters that can be copied to the clipboard over the lifetime of the license	Unsigned integer	Unlimited
start	Date and time when the license begins	ISO 8601-1 date and time	None (perpetual license)
end	Date and time when the license ends	ISO 8601-1 date and time	None (perpetual license)

The date and time format shall follow the rules defined in ISO 8601-1.

All name/value pairs using an integer as their data type (print and copy for this document) shall not use leading zeroes in values (e.g., “9”, not “09”).

For the print right, a page is defined as follows:

a) the page as defined in the publication, if it is pre-paginated (fixed layout), or

b) the page as defined by the page-list nav element of the EPUB navigation document, as defined in EPUB Packages 3.2, subclause 5.4.2.3, if this exists, or

c) 1024 Unicode characters for all other cases.

The copy right only covers the ability to copy to the clipboard and is limited to text (no images, etc.).

In addition to these rights, this document introduces the LCP rights registry, in which all rights defined in this document are referenced.

The rights object may be extended to include any number of implementor-specific rights. Each extension right shall be identified using a URI controlled by the implementor.

EXAMPLE A license document grants the following rights: print up to 10 pages and copy up to 2048 characters for the lifetime of the license; the license has a start and an expiration date. There is also a vendor extension granting the right to tweet parts of this book.

```
{
   "id": "ef15e740-697f-11e3-949a-0800200c9a66",
   "issued": "2013-11-04T01:08:15+01:00",
   "updated": "2014-02-21T09:44:17+01:00",
   "provider": "https://www.imaginaryebookretailer.com",
   "encryption": "...",
   "links": "...",
   "rights": {
       "print": 10,
       "copy": 2048,
       "start": "2013-11-04T01:08:15+01:00",
       "end": "2030-11-25T01:08:15+01:00",
       "https://www.imaginaryebookretailer.com/lcp/rights/tweet": true
  },
   "signature": "..."
}
```

6.3.5 User (identifying the user)

To identify a user, the license document may embed information about the user using the user object. The user object includes the fields defined in Table 8.

Table 8 — Object list in user

Name	Value	Format/data type	Required?
id	Unique identifier for the user at a specific Provider	String	No, but highly recommended
email	The user's e-mail address	String	No
name	The user's name	String	No
encrypted	A list of which user object values are encrypted in this license document	Array of one or more strings matching the above names or an extension	Yes if encryption is used for any field

In addition to this user information, this document introduces the LCP user fields registry, in which all user fields defined in this document are referenced.

As with rights, the user object may be extended to include any number of implementor-specific fields. Each extension field shall be identified by a URI controlled by the implementor.

To protect private user data, any of these fields may be encrypted, except the encrypted field which shall remain in plain text. If encrypted, the field values shall be encrypted using the user key and the same encryption algorithm identified in the encryption/content_key object. The names of all encrypted fields shall be listed in the encrypted array.

EXAMPLE A license document where an identifier, a provider and an email are provided. There is also an extension to indicate the user's preferred language. The email is encrypted.

```
{
  "id": "ef15e740-697f-11e3-949a-0800200c9a66",
  "issued": "2013-11-04T01:08:15+01:00",
  "updated": "2014-02-21T09:44:17+01:00",
  "provider": "https://www.imaginaryebookretailer.com",
  "encryption": "...",
  "links": "...",
  "rights": "...",
  "user": {
       "id": "d9f298a7-7f34-49e7-8aae-4378ecb1d597",
       "email": "EnCt2b8c6d2afd94ae4ed201b27049d8ce1afe31a90ceb8c6d2afd94ae4ed201b2704Rjka
XRveAAarHwdlID1KCIwEmS",
       "encrypted": ["email"],
       "https://www.imaginaryebookretailer.com/lcp/user/language": "fr"
       },
  "signature": "..."
}
```

6.3.6 Signature (signing the license)

To validate that it has not been altered, the license document shall include information about the signature using the signature object. The signature object shall include the fields defined in Table 9.

Table 9 — Object list in signature

Name	Value	Format/data type
algorithm	Algorithm used to calculate the signature, identified using the URIs given in W3C XML Signature. This shall match the signature algorithm named in the encryption profile identified in encryption/profile.	URI
certificate	The provider certificate: an X509 certificate used by the content provider	Base 64 encoded DER certificate
value	Value of the signature	Base 64 encoded octet sequence

For more information on how the signature and the certificate should be calculated, encoded and processed, see 6.5.

EXAMPLE Signature based on the basic LCP encryption profile.

```
{
  "id": "ef15e740-697f-11e3-949a-0800200c9a66",
  "issued": "2013-11-04T01:08:15+01:00",
  "updated": "2014-02-21T09:44:17+01:00",
  "provider": "https://www.imaginaryebookretailer.com",
  "encryption": "...",
  "links": "...",
  "rights": "...",
  "user": "...",
  "signature": {
    "algorithm": "http://www.w3.org/2001/04/xmldsig-more#rsa-sha256",
    "certificate": "MIIDEjCCAfoCCQDwMOjkYYOjPjANBgkqhkiG9w0BAQUFADBLMQswCQYDVQQGEwJVUzETMB
EGA1UECBMKQ2FsaWZvcm5pYTETMBEGA1UEBxMKRXZlcnl3aGVyZTESMBAGA1UEAxMJbG9jYWxob3N0MB4XDTE0MDEw
MjIxMjYxNloXDTE1MDEwMjIxMjYxNlowSzELMAkGA1UEBhMCVVMxEzARBgNVBAgTCkNhbGlmb3JuaWExEzARBgNVBA
cTCkV2ZXJ5d2hlcmUxEjAQBgNVBAMTCWxvY2FsaG9zdDCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAOpC
RECG7icpf0H37kuAM7s42oqggBoikoTpo5yapy+s5eFSp8HSqwhIYgZ4SghNLkj3e652SALav7chyZ2vWvitZycY+a
q50n5UTTxDvdwsC5ZNeTycuzVWZALKGhV7VUPEhtWZNm0gruntronNa8l2WS0aF7P5SbhJ65SDQGprFFaYOSyN6550
P3kqaAO7tDddcA1cmuIIDRf8tOIIeMkBFk1Qf+lh+3uRP2wztOTECSMRxX/hIkCe5DRFDK2MuDUyc/iY8IbY0hMFFG
w5J7MWOwZLBOaZHX+Lf5lOYByPbMH78O0dda6T+tLYAVzsmJdHJFtaRguCaJVtSXKQUAMCAwEAATANBgkqhkiG9w0B
AQUFAAOCAQEAi9HIM+FMfqXsRUY0rGxLlw403f3YtAG/ohzt5i8DKiKUG3YAnwRbL/VzXLZaHru7XBC40wmKefKqoA
0RHyNEddXgtY/aXzOlfTvp+xirop+D4DwJIbaj8/wHKWYGBucA/VgGY7JeSYYTUSuz2RoYtjPNRELIXN8A+D+nkJ3d
xdFQ6jFfVfahN3nCIgRqRIOt1KaNI39CShccCaWJ5DeSASLXLPcEjrTi/pyDzC4kLF0VjHYlKT7lq5RkMO6GeC+7YF
vJtAyssM2nqunA2lUgyQHb1q4Ih/dcYOACubtBwW0ITpHz8N7eO+r1dtH/BF4yxeWl6p5kGLvuPXNU21ThgA==",
    "value": "q/3IInic9c/EaJHyG1Kkqk5v1zlJNsiQBmxz4lykhyD3dA2jg2ZzrOenYU9GxP/xhe5H5Kt2WaJ/
hnt8+GWrEx1QOwnNEij5CmIpZ63yRNKnFS5rSRnDMYmQT/fkUYco7BUi7MPPU6OFf4+kaToNWl8m/ZlMxDcS3BZnVh
SEKzUNQn1f2y3sUcXjes7wHbImDc6dRthbL/E+assh5HEqakrDuA41M8XNfukEYQJnivqhqMLOGM33RnS5nZKrPPK/
c2F/vGjJffSrlX3W3Jlds0/MZ6wtVeKIugR06c56V6+qKsnMLAQJaeOxxBXmbFdAEyplP9irn4D9tQZKqbbMIw=="
  }
}
```

6.4 User key

6.4.1 General

Symmetric encryption relies on an agreed-upon key shared by the sender and receiver: in LCP, this is the user key. The content provider accesses the user key in order to secure the content key within the license document. The reading system also accesses the user key in order to decrypt the content key in the license document.

LCP uses a passphrase model for sharing the user key: in its most generic implementation, when the reading system receives a new license document, it prompts the user for a passphrase to access the content key. LCP defines the user key as a hash of this user passphrase. The user passphrase can be anything at all: a user-defined password, a content provider-defined password like an e-mail address, a library card number, etc.

6.4.2 Calculating the user key

The user passphrase shall be a UTF-8 encoded string. There are no restrictions on the length or content of the user passphrase. There are no requirements for how it is to be created.

The user key is the result of applying a hashing function on the user passphrase, using the algorithm provided in the encryption/user_key object in the license document. Processing of any kind, including whitespace escaping or normalization, shall not be done on the user passphrase before hashing.

6.4.3 Hints

In order to facilitate the entry of the passphrase by the user, the LCP license document supports two ways for the reading system to provide passphrase prompts or hints to the user.

a) The user_key/text_hint field is a simple prompt that can be displayed to the user when entering the user passphrase (e.g., “Enter your Imaginary Book Retailer password”).

b) The links object with rel=hint points to a location that provides passphrase hints or other assistance.

The content of user_key/text_hint and of the resource pointed to by the links object with rel=hint should be human-readable, directed to the user, and should help the user enter their passphrase.

6.4.4 Requirements for the user key and user passphrase

In order to simplify the process for accessing protected publications, the content provider should use the same user key for all licenses issued to the same user.

The security of LCP depends in large part on the security of the user key and user passphrase. Therefore, special care should be taken to secure these throughout the licensing workflow.

a) Passphrases should be sufficiently complex to prevent brute-force attacks.

b) Passphrases shall not be transmitted in plaintext.

c) If the content provider needs to share user key information among multiple systems, such information should be transmitted over secure channels.

d) User keys should be stored by Readium systems in a secure manner; reading systems shall not store user passphrases for privacy reason.

6.5 Signature and public key infrastructure

6.5.1 General

Given the importance of the precise expression of various objects in the license document, it is critical that the reading system be able to verify that the content of the license document is authentic and has not been altered. This is done via a digital signature that is verified via a public key infrastructure as defined in RFC 5280.

Calculating a signature is done on a byte stream, which is unique, while the license document is a JSON document where multiple representations might lead to the same structure. Thus, to ensure a stable signature between the content provider and the reading system, the license document is put in a canonical form before the signature is calculated, as described in 6.5.3. The signature and provider certificate are then added to the license document, as described in 6.5.4.

Reading systems contain the root certificate obtained from the license authority. They first validate the provider certificate, preventing a rogue entity from forging license documents. Then they validate the signature by putting the license document (minus the signature object itself) in its canonical form, calculating the hash of this canonical data, decrypting the signature value given in the signature object using the public key given in the provider certificate and confirm that it matches the calculated hash. This confirms that the contents of the license document have not been altered in transit.

Because the provider certificate is included in the license document and the root certificate is embedded in the reading system, the entire validation process can take place without any Internet connectivity.

To make sure that the provider certificate has not been revoked, the reading system also checks a certificate revocation list maintained by the license authority. In order to facilitate offline reading, the reading system does not need to check the revocation list every time it processes a license document. It is only necessary that it updates the list regularly when an Internet connection is available (e.g. every time it downloads a new license or book).

6.5.2 Certificates

6.5.2.1 Provider certificates

The content providers shall have a certificate in the X.509 v3 format (as defined in RFC 5280) issued and signed by the license authority using the root certificate: this is referred to here as the provider certificate. The subject of the provider certificate should represent the content provider.

Content providers shall distribute their provider certificate in any license document they issue in the signature/certificate field. They also shall use the private key paired with their provider certificate's public key to sign the license document. For a license document to be considered valid, the provider certificate shall have been valid at the time the license document was issued (as indicated by the issued field), and the provider certificate shall not have been revoked.

6.5.2.2 Root certificate

Reading systems shall obtain the root certificate in the X.509 v3 format (as defined in RFC 5280) from the license authority, and should keep it up to date. It shall be embedded in the reading system for offline use.

6.5.3 Canonical form of the license document

6.5.3.1 General

The canonical form of the license document is used when calculating and validating the signature. To create the canonical form of the license document, the following serialization rules shall be followed.

a) Since the signature object of the license document is the product of the calculation, it shall be removed from the license document.

b) All JSON object properties (i.e. key/value pairs) of the license document shall be lexicographically sorted by Unicode code point. Note that this rule is recursive, so that members are sorted at all levels of object nesting.

c) Within arrays, the order of elements shall not be altered.

d) Numbers shall not include insignificant leading or trailing zeroes. Numbers that include a fraction part (non-integers) shall be expressed as a number, fraction, and exponent (normalized scientific notation) using an upper-case "E".

e) Strings shall use escaping only for those characters for which it is required by RFC 4627: backslash (\), double-quotation mark ("), and control characters (U+0000 through U+001F). When escaping control characters, the hexadecimal digits shall be upper case.

f) Non-significant whitespace (as defined in RFC 4627) shall be removed. Whitespace found within strings shall be kept.

6.5.3.2 Example

For this example, a license document which contains a link (hint) and basic user information is used.

```
{
  "id": "ef15e740-697f-11e3-949a-0800200c9a66",
  "issued": "2013-11-04T01:08:15+01:00",
  "updated": "2014-02-21T09:44:17+01:00",
  "provider": "https://www.imaginaryebookretailer.com",
  "encryption": {
    "profile": "http://readium.org/lcp/basic-profile",
    "content_key": {
      "encrypted_value": "/k8RpXqf4E2WEunCp76E8PjhS051NXwAXeTD1ioazYxCRGvHLAck/KQ3cCh5JxD
mCK0nRLyAxs1X0aA3z55boQ==",
      "algorithm": "http://www.w3.org/2001/04/xmlenc#aes256-cbc"
    },
    "user_key": {
        "text_hint": "Enter your email address",
        "algorithm": "http://www.w3.org/2001/04/xmlenc#sha256",
        "key_check": "jJEjUDipHK3OjGt6kFq7dcOLZuicQFUYwQ+TYkAIWKm6Xv6kpHFhF7LOkUK/Owww"
    }
  },
  "links": [
    { "rel": "hint", "href": "https://www.imaginaryebookretailer.com/lcp/hint", "type":
"text/html"}
  ],
  "user": { "id": "d9f298a7-7f34-49e7-8aae-4378ecb1d597"}
}
```

First of all, sort this document. Every JSON object needs to be sorted:

```
{
  "encryption": {
     "content_key": {
    "algorithm": "http://www.w3.org/2001/04/xmlenc#aes256-cbc",
     "encrypted_value": "/k8RpXqf4E2WEunCp76E8PjhS051NXwAXeTD1ioazYxCRGvHLAck/KQ3cCh5JxDmC
K0nRLyAxs1X0aA3z55boQ=="
  },
  "profile": "http://readium.org/lcp/basic-profile",
  "user_key": {
      "algorithm": "http://www.w3.org/2001/04/xmlenc#sha256",
      "key_check": "jJEjUDipHK3OjGt6kFq7dcOLZuicQFUYwQ+TYkAIWKm6Xv6kpHFhF7LOkUK/Owww",
      "text_hint": "Enter your email address"
    }
  },
  "id": "ef15e740-697f-11e3-949a-0800200c9a66",
  "issued": "2013-11-04T01:08:15+01:00",
  "links": [
    {
      "rel": "hint",
      "href": "https://www.imaginaryebookretailer.com/lcp/hint",
      "type": "text/html"
    }
  ],
  "provider": "https://www.imaginaryebookretailer.com",
  "updated": "2014-02-21T09:44:17+01:00",
  "user": {"id": "d9f298a7-7f34-49e7-8aae-4378ecb1d597"}
}
```

Now that the document is sorted, strip all whitespaces and end of lines:

```
{"encryption":{"content_key":{"algorithm":"http://www.w3.org/2001/04/xmlenc#aes256-
cbc","encrypted_value":"/k8RpXqf4E2WEunCp76E8PjhS051NXwAXeTD1ioazYxCRGvHLAck/KQ3cCh5J
xDmCK0nRLyAxs1X0aA3z55boQ=="},"profile":"http://readium.org/lcp/basic-profile","user_
key":{"algorithm":"http://www.w3.org/2001/04/xmlenc#sha256","key_check":"jJEjUDipH
K3OjGt6kFq7dcOLZuicQFUYwQ+TYkAIWKm6Xv6kpHFhF7LOkUK/Owww","text_hint":"Enter your
email address"}},"id":"ef15e740-697f-11e3-949a-0800200c9a66","issued":"2013-11-04T01
:08:15+01:00","links":[{"rel":"hint","href":"https://www.imaginaryebookretailer.
com/lcp/hint","type":"text/html"}],"provider":"https://www.imaginaryebookretailer.
com","updated":"2014-02-21T09:44:17+01:00","user":{"id":"d9f298a7-7f34-49e7-8aae-
4378ecb1d597"}}
```

6.5.4 Generating the signature

6.5.4.1 General

In order to sign a license document, the content provider shall go through the following steps in order.

a) The canonical form of the license document shall be generated following the rules given in 6.5.3.

b) The signature shall be calculated using this canonical form of the license document, using the algorithm described in the algorithm field and the private key of the provider certificate.

c) The signature shall be embedded in the value field of the signature object after Base 64 encoding.

d) The provider certificate used to calculate the signature shall be inserted in the certificate field. It shall use the DER notation as defined in ISO/IEC 8824-1 and be encoded using Base 64.

6.5.4.2 Example

Given the following license document in its canonical form:

```
{"encryption":{"content_key":{"algorithm":"http://www.w3.org/2001/04/xmlenc#aes256-
cbc","encrypted_value":"/k8RpXqf4E2WEunCp76E8PjhS051NXwAXeTD1ioazYxCRGvHLAck/KQ3cCh5J
xDmCK0nRLyAxs1X0aA3z55boQ=="},"profile":"http://readium.org/lcp/basic-profile","user_
key":{"algorithm":"http://www.w3.org/2001/04/xmlenc#sha256","key_check":"jJEjUDipH
K3OjGt6kFq7dcOLZuicQFUYwQ+TYkAIWKm6Xv6kpHFhF7LOkUK/Owww","text_hint":"Enter your
email address"}},"id":"ef15e740-697f-11e3-949a-0800200c9a66","issued":"2013-11-04T01
:08:15+01:00","links":[{"rel":"hint","href":"https://www.imaginaryebookretailer.
com/lcp/hint","type":"text/html"}],"provider":"https://www.imaginaryebookretailer.
com","updated":"2014-02-21T09:44:17+01:00","user":{"id":"d9f298a7-7f34-49e7-8aae-
4378ecb1d597"}}
```

Consider that the signature algorithm required by the encryption profile is SHA-256; the license document is hashed using this algorithm, giving the following byte sequence, represented here in hexadecimal:

```
23c68442c7214ba294ddd1a2902756e9fe575116a88f36e55baf94590a90c2ad
```

This SHA-256 string is then signed using the content provider's private key, giving a Base 64 result of the form:

```
q/3IInic9c/EaJHyG1Kkqk5v1zlJNsiQBmxz4lykhyD3dA2jg2ZzrOenYU9GxP/xhe5H5Kt2WaJ/hnt8+GWrEx1
QOwnNEij5CmIpZ63yRNKnFS5rSRnDMYmQT/fkUYco7BUi7MPPU6OFf4+kaToNWl8m/ZlMxDcS3BZnVhSEKzUNQn
1f2y3sUcXjes7wHbImDc6dRthbL/E+assh5HEqakrDuA41M8XNfukEYQJnivqhqMLOGM33RnS5nZKrPPK/c2F/
vGjJffSrlX3W3Jlds0/MZ6wtVeKIugR06c56V6+qKsnMLAQJaeOxxBXmbFdAEyplP9irn4D9tQZKqbbMIw==
```

With this signature and the certificate, a valid license is of the form:

```
{
"issued":"2013-11-04T01:08:15+01:00",
"updated":"2014-02-21T09:44:17+01:00",
  "encryption": {
    "content_key": {
      "algorithm": "http://www.w3.org/2001/04/xmlenc#aes256-cbc",
      "encrypted_value": "/k8RpXqf4E2WEunCp76E8PjhS051NXwAXeTD1ioazYxCRGvHLAck/KQ3cCh5JxDm
CK0nRLyAxs1X0aA3z55boQ=="
    },
    "profile": "http://readium.org/lcp/basic-profile",
    "user_key": {
      "algorithm": "http://www.w3.org/2001/04/xmlenc#sha256",
"key_check": "jJEjUDipHK3OjGt6kFq7dcOLZuicQFUYwQ+TYkAIWKm6Xv6kpHFhF7LOkUK/Owww",
      "text_hint": "Enter your email address"
    }
  },
  "id": "ef15e740-697f-11e3-949a-0800200c9a66",
  "links": [
    { "rel": "hint",
      "href": "https://www.imaginaryebookretailer.com/lcp/hint",
      "type": "text/html"
    }
  ],
  "user": {"id": "d9f298a7-7f34-49e7-8aae-4378ecb1d597"},
  "signature": {
    "algorithm": "http://www.w3.org/2001/04/xmldsig-more#rsa-sha256",
    "certificate": "MIIDEjCCAfoCCQDwMOjkYYOjPjANBgkqhkiG9w0BAQUFADBLMQswCQYDVQQGEwJVUzETMB
EGA1UECBMKQ2FsaWZvcm5pYTETMBEGA1UEBxMKRXZlcnl3aGVyZTESMBAGA1UEAxMJbG9jYWxob3N0MB4XDTE0MDEw
MjIxMjYxNloXDTE1MDEwMjIxMjYxNlowSzELMAkGA1UEBhMCVVMxEzARBgNVBAgTCkNhbGlmb3JuaWExEzARBgNVBA
cTCkV2ZXJ5d2hlcmUxEjAQBgNVBAMTCWxvY2FsaG9zdDCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAOpC
RECG7icpf0H37kuAM7s42oqggBoikoTpo5yapy+s5eFSp8HSqwhIYgZ4SghNLkj3e652SALav7chyZ2vWvitZycY+a
q50n5UTTxDvdwsC5ZNeTycuzVWZALKGhV7VUPEhtWZNm0gruntronNa8l2WS0aF7P5SbhJ65SDQGprFFaYOSyN6550
P3kqaAO7tDddcA1cmuIIDRf8tOIIeMkBFk1Qf+lh+3uRP2wztOTECSMRxX/hIkCe5DRFDK2MuDUyc/iY8IbY0hMFFG
w5J7MWOwZLBOaZHX+Lf5lOYByPbMH78O0dda6T+tLYAVzsmJdHJFtaRguCaJVtSXKQUAMCAwEAATANBgkqhkiG9w0B
AQUFAAOCAQEAi9HIM+FMfqXsRUY0rGxLlw403f3YtAG/ohzt5i8DKiKUG3YAnwRbL/VzXLZaHru7XBC40wmKefKqoA
0RHyNEddXgtY/aXzOlfTvp+xirop+D4DwJIbaj8/wHKWYGBucA/VgGY7JeSYYTUSuz2RoYtjPNRELIXN8A+D+nkJ3d
xdFQ6jFfVfahN3nCIgRqRIOt1KaNI39CShccCaWJ5DeSASLXLPcEjrTi/pyDzC4kLF0VjHYlKT7lq5RkMO6GeC+7YF
vJtAyssM2nqunA2lUgyQHb1q4Ih/dcYOACubtBwW0ITpHz8N7eO+r1dtH/BF4yxeWl6p5kGLvuPXNU21ThgA==",
    "value": "q/3IInic9c/EaJHyG1Kkqk5v1zlJNsiQBmxz4lykhyD3dA2jg2ZzrOenYU9GxP/xhe5H5Kt2WaJ/
hnt8+GWrEx1QOwnNEij5CmIpZ63yRNKnFS5rSRnDMYmQT/fkUYco7BUi7MPPU6OFf4+kaToNWl8m/ZlMxDcS3BZnVh
SEKzUNQn1f2y3sUcXjes7wHbImDc6dRthbL/E+assh5HEqakrDuA4lM8XNfukEYQJnivqhqMLOGM33RnS5nZKrPPK/
c2F/vGjJffSrlX3W3Jlds0/MZ6wtVeKIugR06c56V6+qKsnMLAQJaeOxxBXmbFdAEyplP9irn4D9tQZKqbbMIw=="
  }
}
```

6.5.5 Validating the certificate and signature

6.5.5.1 Validating the certificate

a) The reading system shall check the signature of the provider certificate using the root certificate it embeds.

b) If a network connection is available, the reading system shall periodically update its certificate revocation list, as defined in RFC 5280.

3) The reading system shall check that the provider certificate was not revoked, as defined in RFC 5280.

4) The reading system shall check that the provider certificate was not expired when the license document was last updated.

6.5.5.2 Validating the signature

In order to validate the signature, the following steps shall be followed in order.

a) The reading system shall extract and remove the signature from the license document.

b) The reading system shall calculate the canonical form of the license document following the rules expressed in 6.5.3.

c) The reading system shall recalculate the signature as defined in 6.5.4.

d) The reading system shall verify that the calculated signature value is consistent with the one previously extracted from the license document.

7 License status document

7.1 General

This clause defines the status of a DRM license along with the interactions that might affect its status. It may also contain a history of the events associated with the license.

The interactions defined in this document aim at supporting lending in public libraries, where a user may have the ability to renew a time-limited loan or return one before it expires.

In case the oversharing of a protected publication has been detected by a content provider, the license status is the way a license revocation is notified to a reading system.

A.2 shows an example of a license status document. Annex B depicts use case scenarios for library lending model using license status document.

7.2 Content conformance

A license status document shall meet all of the following criteria.

— It shall meet the conformance constraints for JSON documents as defined in RFC 4627.

— It shall parse as a single JSON object.

— It shall be encoded using UTF-8.

— Its access shall not require any form of authentication.

— It should not be served over an insecure connection.

— Its MIME media type for a status document is application/vnd.readium.license.status.v1.0+json, and HTTP servers should set the “Content-Type” header appropriately.

7.3 License status information

7.3.1 General

A status document contains objects defined in Table 10.

Table 10 — Object list of license status document

Key	Semantics	Format/data type	Required?
id	Unique identifier for the license document associated to the status document	String	Yes
status	Current status of the license	Controlled vocabulary, as defined in 7.3.2	Yes
message	A message meant to be displayed to the user regarding the current status of the license	String	Yes
updated	updated object	Object as defined in 7.3.3	Yes
links	links object	Object as defined in 7.3.4	Yes
potential_rights	potential_rights object	Object as defined in 7.3.5	No
events	events object	Object as defined in 7.3.6	No

All messages provided in the message key should be localized based on the HTTP Accept-Language header included in the request for the status document.

7.3.2 Status

The status field succinctly describes the current status of the license. The values defined in Table 11 are allowed.

Table 11 — Allowed value list in status

Name	Semantics
ready	The license document is available, but the user hasn't accessed the license and/or status document yet.
active	The license is active, and a device has been successfully registered for this license. This is the default value if the license document does not contain a registration link, or a registration mechanism through the license itself.
revoked	The license is no longer active, it has been invalidated by the issuer.
returned	The license is no longer active, it has been invalidated by the user.

Table 11 *(continued)*

Name	Semantics
cancelled	The license is no longer active because it was cancelled prior to activation.
expired	The license is no longer active because it has expired.

7.3.3 Updated (timestamps)

The status document shall include an updated object where the keys and associated timestamps defined in Table 12 are provided.

Table 12 — Object list in updated

Key	Semantics	Format/data type
license	Time and date when the license document was last updated	ISO 8601-1 time and date
status	Time and date when the status document was last updated	ISO 8601-1 time and date

The time and date format shall follow the rules defined in ISO 8601-1.

7.3.4 Links

7.3.4.1 General

The status document shall contain a links object, which is an array of link objects where each link object contains the link URI, a link relationship and an optional set of other properties.

A status document shall include at least one link where the relation is set to license.

7.3.4.2 Link object

Each link object contained in links supports the keys defined in Table 13.

Table 13 — Object list in link

Name	Value	Format/data type	Required
href	Link location	URI or URI template	Yes
rel	Link relationship to the document	List of well-known relation values, URIs for extensions as defined in 7.3.4.3	Yes
title	Title of the link	String	No
type	Expected MIME media type value for the external resources	MIME media type	No, but highly recommended
templated	Indicates that the href is a URI template	Boolean	No, default value is "false"
profile	Expected profile used to identify the external resource	URI	No, default value is "http://readium.org/lcp/basic-profile"

7.3.4.3 Link relationships

This document introduces the link relationships defined in Table 14 for each link object which is used for value of rel.

Table 14 — Allowed value list in rel

Relation	Semantics	Templated	Required	HTTP verb
license	Location of the license document associated to the status document	No	Yes	GET
register	Interaction way associated to register new device	Yes	No	POST
return	Interaction way associated to return a license	Yes	No	PUT
renew	Interaction way associated to renew a license	Yes	No	PUT

7.3.5 Potential rights

A status document may provide a list of potential changes allowed for a license document using the potential_rights object.

This document defines the expression of potential rights in Table 15.

Table 15 — Object list in potential_rights

Key	Semantics	Format/data type
end	Time and date when the license ends	ISO 8601-1 time and date

The time and date format shall follow the rules defined in ISO 8601-1.

7.3.6 Events

A status document may provide an ordered list of events related to the change in status of a license document.

These events are documented in an events object where the keys defined in Table 16 are used to describe an event.

Table 16 — Object list in events

Key	Semantics	Format/data type
type	Identifies the type of event	See the enumeration below.
name	Name of the client, as provided by the client during an interaction	String
id	Identifies the client, as provided by the client during an interaction	String
timestamp	Time and date when the event occurred	ISO 8601-1 time and date

The time and date format shall follow the rules defined in ISO 8601-1.

The allowed type values are defined in Table 17.

Table 17 — Allowed value list of type

Name	Semantics
register	Signals a successful registration event by a device
renew	Signals a successful renew event
return	Signals a successful return event
revoke	Signals a revocation event
cancel	Signals a cancellation event

7.4 Interactions

7.4.1 General

This subclause defines the interactions that are exposed through the links object in the license status document, and the errors which can appear during such interactions.

7.4.2 Handling errors

To provide a consistent behavior for both clients and end users, all servers shall handle errors using the Problem Details JSON object as defined in RFC 7807 with the following additional requirements:

— The server shall return a title and a type in the Problem Details JSON object;

— The server should attempt to localize both title and detail based on the Accept-Language header sent by the client.

Each interaction in this document provides a number of failure types that are returned to the client if the interaction fails.

7.4.3 Checking the status of a license

A client should on a regular basis retrieve a status document in order to update a license document.

If the license timestamp in the updated object of the status document is more recent than the timestamp contained in the local copy of the license document, the client shall download the license document again and replace its previous copy with the new one.

If the links object contains more than one link with a relation set to license, the client shall select one of the link objects based on the value of their profile property. The selected link object shall correspond to the highest profile value the client can handle. The order of the links is of no importance.

If the client does not find a profile it can handle in the previous set, it shall consider that no updated license is available.

If the status value of a status document contradicts the corresponding up-to-date license document, the license document takes precedence.

If the status document is unavailable or if the client is unable to obtain an internet connection, it shall not block the user from accessing the publication tied to the license document.

7.4.4 Registering a device

Registration is meant to provide the server with a hint of how many clients have interacted with a license document.

When a client opens a license document for the first time and gets access to its associated status document:

— it shall attempt to register itself using the link exposed in the status document;

— it shall not block the user from accessing the publication associated to the license document if the registration fails;

— it shall attempt to register itself again if it couldn't do so the first time that the license document was opened.

During the registration, the client shall always try to send the same unique identifier for a specific device, no matter which status document it interacts with. Any further interaction with a provider should use the same identifier/name. The client should consider user privacy when generating a unique identifier, for example by generating a random string during software installation. To prevent user tracking across providers, the client may generate device unique ids for each provider.

If a provider uses registration to monitor license abuse, the provider should take care to prevent forged registrations. Table 18 defines a HTTP protocol for the register interaction.

Table 18 — HTTP protocol for register interaction

Relation	Semantics	Templated?	Required?	HTTP verb
register	Associate a new device with the License	Yes	No	POST
Parameter	**Format**	**Semantics**		**Required?**
id	String	A unique identifier for the device		Yes
name	String	Human readable name for the device		Yes

Table 19 defines expected behavior of server and client on the register interaction.

Table 19 — Expected behavior of register interaction

Server side behavior	HTTP status code	Client side behavior
The server registers the device identified by 'id' and returns an updated status document. The server shall update the timestamp of the status document contained in the status key of the updated object. If the status was previously set to ready, it shall be updated by the server to 'active' instead. The server may also add a new event in the events object of the status document.	200	The client shall not attempt to register the device again.

Table 20 defines expected failures of server response on the register interaction.

Table 20 — Expected failures of register interaction

Type	HTTP status code	Title
http://readium.org/license-status-document/error/registration	400	Your device could not be registered properly.
http://readium.org/license-status-document/error/server	5xx	An unexpected error has occurred.

EXAMPLE A simple registration link.

```
{
  "links": [
    {
      "rel": "register",
      "href": "https://example.org/license/aaa-bbbb-ccc/register?id,name",
      "type": "application/vnd.readium.license.status.v1.0+json",
      "templated": true
    }
  ]
}
```

7.4.5 Returning a publication

Returning a publication is meant primarily for library use cases, where a patron can return a publication early in order to get access to a new loan.

If returning is unsuccessful, the client should attempt to return the license again at a later time without necessarily asking the user again. Table 21 defines a HTTP protocol for the return interaction.

Table 21 — HTTP protocol for return interaction

Relation	Semantics	Templated?	Required?	HTTP verb
return	Request for the License to be immediately deactivated.	Yes	No	PUT
Parameter	**Format**	**Semantics**		**Required?**
id	String	Unique identifier for the device		No
name	String	Human readable name for the device		No

Table 22 defines expected behavior of server and client on the return interaction.

Table 22 — Expected behavior of return interaction

Server side behavior	HTTP status code	Client side behavior
The server shall return an updated status document. The status document shall contain a status with its value set to ‘returned’ if the status was previously set to ‘active’ or ‘cancelled’ if the status was previously set to ‘ready’. The server shall update the timestamp of both license and status document contained in the status and license keys of the updated object. The server may also add a new event in the events object of the status document.	200	The client shall download an updated status document. The client shall not allow the user to open the publication anymore. The client should not attempt to return the license anymore.

Table 23 defines expected failures of server response on the return interaction.

Table 23 — Expected failures of return interaction

Type	HTTP status code	Title
http://readium.org/license-status-document/error/return	400	Your publication could not be returned properly.
http://readium.org/license-status-document/error/return/already	403	Your publication has already been returned before.
http://readium.org/license-status-document/error/return/expired	403	Your publication has already expired.
http://readium.org/license-status-document/error/server	5xx	An unexpected error has occurred.

EXAMPLE A simple return link.

```
{
  "links": [
    {
      "rel": "return",
        "href": "https://example.org/license/aaa-bbbb-ccc/return?id,name",
        "type": "application/vnd.readium.license.status.v1.0+json",
        "templated": true
}
  ]
}
```

7.4.6 Renewing a license

Renewing a license is also meant primarily for library use cases, where a patron can renew a loan for an extended period of time. Table 24 defines a HTTP protocol for the renew interaction.

Table 24 — HTTP protocol for renew interaction

Relation	Semantics	Templated?	Required?	HTTP verb
renew	Request for extending the term of the license.	Yes	No	See below
Media type	**Semantics**		**Templated?**	**Verb**
text/html	A URL where human interactions will be required. Returns an HTML page. These interactions should ultimately result in the extension of the term of a license document.		No	GET
application/vnd.readium.license.status.v1.0+json	A URL where the License Document can be programmatically renewed. Returns a status document.		Yes	PUT

The parameters defined in Table 25 are strictly for the case where the license document is programmatically renewed.

Table 25 — Parameters for renew interaction protocol

Parameter	Format	Semantics	Required?
id	String	A unique identifier for the device	No
name	String	Human readable name for the device	No
end	ISO 8601-1	A new expiration date for the license	No

The time and date format shall follow the rules defined in ISO 8601-1.

Table 26 defines expected behavior of server and client on the renew interaction.

Table 26 — Expected behavior of renew interaction

Server side behavior	HTTP status code	Client side behavior
The server shall return an updated status document. The server shall update the timestamp of both license and status document contained in the status and license keys of the updated object. The server may also add a new event in the events object of the status document.	200	The client shall download an updated status document. The client may attempt to renew the license again later.

Table 27 defines expected failures of server response on the renew interaction

Table 27 — Expected failures of renew interaction

Type	HTTP status code	Title
http://readium.org/license-status-document/error/renew	4xx	Your publication could not be renewed properly.
http://readium.org/license-status-document/error/renew/date	403	Incorrect renewal period, your publication could not be renewed.

Table 27 *(continued)*

Type	HTTP status code	Title
http://readium.org/license-status-document/error/server	5xx	An unexpected error has occurred.

EXAMPLE An example with two renewal links: one meant for human interactions (HTML) and another for clients that can send the proper information to the server (returns a status document).

```
{
  "links": [
    {
         "rel": "renew",
         "href": "https://example.org/license/aaa-bbbb-ccc/renew",
         "type": "text/html"
   },
    {
         "rel": "renew"
         "href": "https://example.org/license/aaa-bbbb-ccc/renew?end,id,name",
         "type": "application/vnd.readium.license.status.v1.0+json",
         "templated": true
   }
  ]
}
```

8 Encryption profile

8.1 General

In order to maintain maximum flexibility, no specific algorithms are specified by this document. Instead, the design of both encryption.xml and the license document allow for the identification of encryption algorithms to be discovered by reading systems when presented with a protected publication.

In order to simplify the discovery process, LCP defines the notion of encryption profile, which is the set of encryption algorithms used in a specific protected publication and associated license document. Reading systems that implement the algorithms identified in the encryption profile will be able to decrypt protected publications encoded using such encryption profile. The identification of the encryption profile in the license document eases the discovery of these requirements by reading systems.

This document defines the Basic encryption profile 1.0, composed from a set of associated algorithms extracted from W3C XML Encryption or W3C XML Signature. The Basic encryption profile is for test only, as it does not provide the level of obfuscation required by a reliable protection mechanism.

Other encryption profiles are (or will be) defined for use in production; these profiles are referenced in the LCP encryption profiles registry.

8.2 Encryption profile requirements

All encryption profiles shall identify algorithms for the following targets:

a) publication resources;

b) content key and user fields (when encrypted);

c) user passphrase;

d) signature.

All algorithms used in an encryption profile should be defined in W3C XML Encryption or W3C XML Signature.

All encryption profiles shall use a URI to identify themselves in the profile property of the encryption object in the license document.

8.3 Basic encryption profile 1.0

The Basic encryption profile 1.0 is officially identified by the URL http://readium.org/lcp/basic-profile.

The algorithms defined in Table 28 are associated with the Basic encryption profile 1.0.

Table 28 — Algorithm list for the basic encryption profile 1.0

Encryption target	Algorithm (name)	Algorithm (URI)	Identified in
Publication resources	AES 256 bits CBC	http://www.w3.org/2001/04/xmlenc#aes256-cbc	encryption.xml
Content key, user fields (if encrypted)	AES 256 bits CBC	http://www.w3.org/2001/04/xmlenc#aes256-cbc	License document
User passphrase	SHA-256	http://www.w3.org/2001/04/xmlenc#sha256	License document
Signature	RSA with SHA-256	http://www.w3.org/2001/04/xmldsig-more#rsa-sha256	License document

9 Integration in EPUB

9.1 General

LCP follows the EPUB OCF 3.2 specification to identify encrypted resources within the EPUB container. Specifically, LCP uses META-INF/encryption.xml to indicate which resources listed in the package document are encrypted and to reference the content key.

9.2 Encrypted resources

In order to make sure that protected publications remain identifiable by reading systems, a certain number of resources are prohibited from being encrypted.

LCP inherits the following list of files prohibited from being encrypted from the EPUB OCF 3.2 specification:

— mimetype;

— META-INF/container.xml;

— META-INF/encryption.xml;

— META-INF/manifest.xml;

— META-INF/metadata.xml;

— META-INF/rights.xml;

— META-INF/signatures.xml;

— EPUB rootfiles (the package document for any rendition).

In addition, this document defines that the following files shall not be encrypted:

— META-INF/license.lcpl;

— navigation documents referenced in any package document from the publication (all publication resources listed in the publication manifest with the “nav” property defined in EPUB Packages 3.2, subclause E.2.3);

— NCX documents referenced in any package document from the publication (all publication resources listed in the publication manifest with the media type “application/x-dtbncx+xml”);

— cover images (all publication resources listed in the publication manifest with the “cover-image” property defined in EPUB Packages 3.2, subclause E.2.1).

9.3 Using META-INF/encryption.xml for LCP

As defined in the EPUB OCF 3.2 specification, all encrypted publication resources are identified in the META-INF/encryption.xml file using W3C XML Encryption.

Publication resources with non-codec content types should be compressed and the Deflate compression algorithm shall be used. This practice ensures that file entries stored in the container have a smaller size.

Resources with codec content types should be stored without compression. In such case, compression would introduce unnecessary processing overhead at production time (especially with large resource files) and would impact audio/video playback performance at consumption time.

In publications protected using LCP, there are additional requirements for determining the length of the full resource ahead of media playback, especially when using partial content requests:

Streams of data that are compressed before they are encrypted shall provide additional EncryptionProperties metadata to specify the size of the initial resource (i.e., before compression and encryption), as per the Compression XML element defined in EPUB OCF 3.2, subclause 3.5.2.2.1).

Streams of data that are not compressed before they are encrypted should provide such additional EncryptionProperties metadata.

In publications protected using LCP, the following XML syntax identifies the key used to encrypt resources (the LCP content key).

a) The ds:KeyInfo element shall point to the content key using the ds:RetrievalMethod element.

b) The URI attribute of ds:RetrievalMethod shall use a value of “license.lcpl#/encryption/content_key” to point to the encrypted content key stored in the license document. This URI follows JSON Pointer defined in RFC 6901.

c) The Type attribute shall use a value of “http://readium.org/2014/01/lcp#EncryptedContentKey” to identify the target of the URI as an encrypted content key.

EXAMPLE encryption.xml expresses that main.html (original size 35K) is deflated and aes256-cbc encrypted using an LCP content key.

```
<encryption
    xmlns ="urn:oasis:names:tc:opendocument:xmlns:container"
    xmlns:enc="http://www.w3.org/2001/04/xmlenc#"
    xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
    <enc:EncryptedData Id="ED1">
        <enc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#aes256-cbc"/>
        <ds:KeyInfo>
            <ds:RetrievalMethod URI="license.lcpl#/encryption/content_key" Type="http://
readium.org/2014/01/lcp#EncryptedContentKey"/>
        </ds:KeyInfo>
        <enc:CipherData>
            <enc:CipherReference URI="main.html"/>
        </enc:CipherData>
        <enc:EncryptionProperties>
           <enc:EncryptionProperty xmlns:ns="http://www.idpf.org/2016/
encryption#compression">
              <ns:Compression Method="8" OriginalLength="35000"/>
            </enc:EncryptionProperty>
         </enc:EncryptionProperties>
    </enc:EncryptedData>
</encryption>
```

10 Reading system behavior

10.1 Detecting LCP protected publication

reading systems can detect that a publication is protected using LCP by either of these means:

a) the presence of the license document (META-INF/license.lcpl);

b) the presence of a ds:KeyInfo/ds:RetrievalMethod element in META-INF/encryption.xml that references an LCP content key (i.e., Type attribute value is "http://readium.org/2014/01/lcp#EncryptedContentKey")

If encryption.xml references an LCP content key but the license document is missing, the reading system should report this to the user.

10.2 License document processing

10.2.1 Overall

In processing a license document, reading systems shall ignore all name/value pairs they do not understand.

Reading systems shall not store unencrypted versions of the content key and/or unencrypted user information.

10.2.2 Validating the license document

Reading systems shall:

a) validate the syntax and completeness of the license document;

b) validate the signature as defined in 6.5.5.

10.2.3 Acquiring the publication

When license documents are delivered independently of the protected publication, reading systems shall download the protected publication at the URL given in links/publication/href.

A reading system that will make the protected publication file accessible to the user shall add the license document to the downloaded protected publication at META-INF/license.lcpl.

Reading systems should verify the integrity of the downloaded protected publication, if a hash is provided.

A reading system should report any failure to acquire the protected publication to the user.

10.2.4 License status processing

If a link with the relation status is not present in the links object, the client shall process the license document according to the core LCP specification.

If a link with the relation status is present in the links object, the client shall process the status document according to the rules listed in 7.4.3.

10.3 User key processing

Reading systems shall:

— show the text hint and the URL of the hint page (if any) when prompting the user for their user passphrase.

Reading systems should:

— store the user key in a secured manner;

— try previously stored user keys before prompting the user to enter their user passphrase for processing a license document.

Reading systems may:

— use an alternate technique to discover and exchange the user key in the background;

— associate specific user keys with specific providers and user IDs in order to optimize the discovery of the correct user key for processing a license document.

Reading systems shall not:

— store the user passphrase, but only the user key.

10.4 Signature processing

Reading systems shall:

— validate the signature and provider certificate as described in 6.5.5;

— update the certification revocation list on a regular basis.

Reading systems should:

— be able to update their root certificate.

Reading systems may:

— choose to open a publication even if their own root certificate is deprecated.

Reading systems shall not:

— open a publication with an invalid signature or provider certificate;

— block a user from opening a publication if a network connection is unavailable or the certificate revocation list can't be reached.

10.5 Publication processing

Reading systems shall:

— respect all rights limitations given in the license document.

Reading systems may:

— delete a protected publication if its License expired.

Reading systems shall not:

— store unencrypted publication resources.