



Radium LCP (IRead-Rafeeki) V 1.0

Prepared By Dev Team

THE "NO PROBLEM" PEOPLE

TABLE OF CONTENTS

Document Purpose.....	4
Audience.....	4
Introduction.....	4
What is Radium LCP:.....	5
Radium LCP Components:	5
Radium LCP Setup:	5
• Radium LCP Installation:	5
• Radium LCP Configuration:	6
READIUM LCP Components Integration:.....	7
• LCP Encrypt (Encryption Utility):.....	7
• LCP Server (License Server):.....	7
▪ General Notes:	7
▪ Integration API Endpoints.....	7
1. Add Publication (content):	7
2. Get Encrypted Publication (content):	9
3. Create License:.....	10
4. Get an Existing License:	12
5. List All Existing Licenses:.....	13
6. List All Existing content Licenses:	14
7. Update License:	15
• LSD Server (License Status Document)	16
▪ General Notes:	16
▪ Integration API Endpoints.....	18
1. Create a license status document:	18
2. Get license status document:	18
3. Register Device:.....	18
4. Return License:.....	19
5. Renew License:.....	20
6. Get Licenses Registered Devices Count:.....	21
7. Get License Registered Devices:	22
8. Revoke/Cancel License:.....	23

Revision History

Date	Description	Author	Comments
27 March 2022	Initial Radium Doc	Aly Salem	

DOCUMENT PURPOSE

The purpose from this document is to present a detailed description of Radium LCP and to show and demonstrate how to install and integrate with API Endpoints provided by Radium LCP to achieve the desired level of DRM (Digital Rights Management) through LCP (License Content Protection).

AUDIENCE

The audience of this document, whom are interested and concerned with the technical aspects of Radium LCP and its integration.

INTRODUCTION

This document will explain the details of Radium LCP, its usage, benefits, components, installation and API Endpoints to achieve the integration. Also it will explain the specific integration cycle and process of IRead(Rafeeki) with Radium LCP.

WHAT IS RADIUM LCP:

- Radium LCP (Licensed Content Protection) is a DRM (Digital Rights Management) solution that enhances the process of distributing, purchase and loan of protected content eBooks based on encryption and user passphrase which may be his password, email, mobile number ext....
- It efficiently protects the content against over-sharing. It generates licenses to users so that they can purchase, loan, renew the loan or even cancel the license which provides them the authority to view the protected content.

READIUM LCP COMPONENTS:

1. Lcpencrypt (Encryption utility):
 - A command line utility for content encryption.
 - It stores the encrypted file into a file system or S3 bucket
 - It also notifies the License Server (lcpserver) with the newly added protected content to store its data including the encryption key (Content Key).
2. Lcpserver (License Server):
 - Server that generates a license or a protected publication (i.e. an encrypted publication in which a license is embedded).
 - It also updates the rights associated with a license, get an updated license or a set of licenses.
 - This server should be in a private LAN protected from internet.
3. Lsdserver (License Status Server):
 - Server that can revoke, cancel or check the license update status and its registered devices.
 - This server is public to internet since it has a number of public functionalities accessible from the web, like checking the status document of a specific license, device registration, or even lending renewal.

READIUM LCP SETUP:

• RADIUM LCP INSTALLATION:

1. Install Golang, 1.13 or higher, however it is **highly recommended** to avoid reaching or exceeding 1.16 since the upcoming commands are not supported in those versions (previously used version on Dev environment is 1.14).
 - Install Link: <https://go.dev/dl/>
 - Install Path: C:\Program Files\Go\
2. Install Git.
 - Install Link: <https://git-scm.com/download/win>
3. Install a GCC compiler:
 - Install Link: <https://jmeubank.github.io/tdm-gcc/>
4. Open the CMD:

- run the following commands:
 1. `cd %GOPATH%`
 2. `go get -v github.com/readium/readium-lcp-server/...`
- GOPATH is a path set as environment variable which lists places to look for Go code, it contains a values list of paths separated by semicolon ";". The default value is "%USERPROFILE%\go" e.g (C:\Users\deployadmin\go)
- Change GOPATH if needed:
 1. Via CMD using GoLang: `go env -w GOPATH=c:\go-work`
 2. Via UI: search for environment and select "Edit environment variables for your account" then you will find GOPATH and change its value.
 3. Reference Link: <https://github.com/golang/go/wiki/SettingGOPATH>
- After Running the second command:
 1. Three folders will be created in GOPATH which are (bin, pkg, src)
 2. The bin folder should contain the exe files for (lcpencrypt, lcpserver, lsdserver, frontend).exe files which will be running later to run readium servers.
 3. If the bin folder doesn't contain them, you can get each separately using this command for each component e.g "go get -v github.com/readium/readium-lcp-server/lcpserver "
- 5. Install NodeJs (NPM), **in case of running the test FrontEnd** provided by Radium:
 - Install Link: <https://nodejs.org/en/download/>
 - Install Path: C:\Program Files\nodejs\
 - After installing run the below CMD commands
 1. `cd $GOPATH/src/github.com/readium/readium-lcp-server/frontend/manage`
 2. `npm install`
 3. `npm start`

• READIUM LCP CONFIGURATION:

1. The LCP, LSD Servers and Frontend test server will search their configuration file in the go bin directory by default, but the path to the file can be changed using an environment variable
 - **READIUM_LCPSERVER_CONFIG** for the License server
 - **READIUM_LSDSERVER_CONFIG** for the License Status server
 - **READIUM_FRONTEND_CONFIG** for the Frontend test server
2. Each server may have its own config file, or all server have only one config file (i.e if they are all deployed on the same server).
3. The LCP and LSD servers also require authenticated API requests for some of their functionalities. So, a password file formatted as an Apache "htpasswd" file is used for such authentication data and its path is referenced from the config file.
 - It is recommended to name the file "htpasswd" without any extension.
 - To generate the file content, use the below link: <https://hostingcanada.org/htpasswd-generator>
4. A sample Config for all and each server with description and clarification of its properties is attached to this Document, links:

- [ConfigForAllServers](#)
 - [LCPConfig](#)
 - [LSDConfig](#)
 - [TestFrontEndConfig](#)
5. After setting configuration in "bin" folder run lcpencrypt.exe, lsdserver.exe, lcpserver.exe.

READIUM LCP COMPONENTS INTEGRATION:

• LCP ENCRYPT (ENCRYPTION UTILITY):

- A command line utility for content encryption.
- It stores the encrypted file into a file system or S3 bucket
- It also notifies the License Server (lcpserver) with the newly added protected content to store its data including the encryption key (Content Key).
-

• LCP SERVER (LICENSE SERVER):

▪ GENERAL NOTES:

- **All** API Endpoints provided by LCP Server are private which means it requires authentication via (Basic Authorization). It is implemented by Sending the Username and password in the request header after applying base64 encoding.
 - Encoding String is "{Username}:{Password}"
 - Ex test:123456 generates "dGVzdDoxMjM0NTYg"
- A sample complete license with description and clarification of its properties is attached to this document, Link:
 - [SampleLicenseDescription.txt](#)
- All Sent and received Date Time are ISO 8601 Formatted, Which is compatible with C# DateTime Data Type.

▪ INTEGRATION API ENDPOINTS

1. Add Publication (content):

- PUT <LCPbaseURL>/contents/<content_id>
- This method is used primarily by "lcpencrypt" to store data about generated encrypted publication.
- Request:

Parameter Name	Description	Type
content-id	Unique Identifier for publication "GUID". The content_id in the URL must be equal to the content-id in the json.	String

content-encryption-key	encryption key used to encrypt the publication	String
protected-content-location	absolution URL or file path of the encrypted content	String
protected-content-length	size of the encrypted content	Integer
protected-content-sha256	hash of the encrypted content	String
protected-content-disposition	original file name	String

FIGURE 1: ADD CONTENT REQUEST SAMPLE

```
{
  "content-id": "string",
  "content-encryption-key": "string",
  "protected-content-location": "string",
  "protected-content-length": "string",
  "protected-content-sha256": "string",
  "protected-content-disposition": "string"
}
```

- Response:

HTTP Status Code	Description
200	Ok DB Updated
201	Content was added (created)
401	Bad request
404	File not found
500	internal server error

2. Get Encrypted Publication (content):

- GET <LCPbaseURL>/contents/<content_id>
- Fetch encrypted publication (not licensed publication).
- Response:

HTTP Status Code	Description
200	Content found and returns the content as file stream
404	File not found
500	internal server error

3. Create License:

- POST <LCPbaseURL>/contents/<content_id>/license
- Hash online tool link (for generating user passphrase hash):
<https://passwordsgenerator.net/sha256-hash-generator/>.
- Note: LCP can create the license for the same request (doesn't validate duplicate license request on the same content)
- Request:

Parameter Name	Description	Type
provider	provider identifier (a URL).	String
user	Json object that contains user Information	Object
user/ id	User identifier in provider DB, Mandatory.	String
user/email	User Email, optional.	String
user/ encrypted	Encrypted values the provider wants to embed in the license, optional.	String array
encryption/user_key	User Key hash data	Object
encryption/user_key /text_hint	Hint to be shown for User to ask for passphrase	String
encryption/user_key /hex_value	Value of hashed user passphrase with below algorithm as hex-encoded string	String
encryption/user_key /algorithm	The URI of the hashing algorithm, if this property is removed the value will be sha256 (Default)	String
rights	the set of rights associated with the license, Optional	Object
rights/start	Start Date of license	DateTime
rights/end	End Date of license	DateTime
rights/ print	Maximum number of pages that can be printed over the lifetime of the license, if zero copy will not be allowed, to be unlimited don't send it	Integer
rights/ copy	Maximum number of characters that can be copied over the lifetime of the license, if zero copy will not be allowed, to be unlimited don't send it	Integer

FIGURE 2 CREATE LICENSE REQUEST SAMPLE

```
{
  "provider": "http://www.iRead.com",
  "user": {
    "id": "d9f298a7-7f34-49e7-8aae-4378ecb1d597",
    "email": "user@mymail.com",
    "encrypted": ["email"]
  },
  "encryption": {
    "user_key": {
      "text_hint": "The title of the first book you ever read",
      "hex_value": "4981AA0A50D563040519E9032B5D74367B1D129E239A1BA82667A57333866494",
      "algorithm": "http://www.w3.org/2001/04/xmlenc#sha256"
    }
  },
  "rights": {
    "print": 10,
    "copy": 2048,
    "start": "2019-11-04T01:08:15+01:00",
    "end": "2019-11-25T01:08:15+01:00"
  }
}
```

▪ Response:

HTTP Status Code	Description
201	License created and returns license sample response license is provided in SampleLicenseDescription.txt file attached to this document.
400	Bad Request
404	Content not found

4. Get an Existing License:

- POST <LCPbaseURL>/licenses/<license_id>
- Request:

Parameter Name	Description	Type
user	Json object that contains user Information	Object
user/email	User Email, optional.	String
user/ encrypted	Encrypted values the provider wants to embed in the license, optional.	String array
encryption/user_key	User Key hash data	Object
encryption/user_key /text_hint	Hint to be shown for User to ask for passphrase	String
encryption/user_key /hex_value	Value of hashed user passphrase with below algorithm as hex-encoded string	String
encryption/user_key /algorithm	The URI of the hashing algorithm, if this property is removed the value will be sha256 (Default)	String

FIGURE 3 GET LICENSE REQUEST SAMPLE

```
{
  "user": {
    "email": "user@mymail.com",
    "encrypted": ["email"]
  },
  "encryption": {
    "user_key": {
      "text_hint": "The title of the first book you ever read",
      "hex_value": "4981AA0A50D563040519E9032B5D74367B1D129E239A1BA82667A57333866494",
      "algorithm": "http://www.w3.org/2001/04/xmlenc#sha256"
    }
  }
}
```

- Response:

HTTP Status Code	Description
200	returns license. sample response license is provided in SampleLicenseDescription.txt file attached to this document.
400	Bad Request
404	License not found

5. List All Existing Licenses:

- GET <LCPbaseURL>/licenses{page?, per_page?}
- List licenses ordered by creation Date (issue date)
- Page numbering is 1-based (starts with 1 not 0), and the default per_page is 30
- Example: http://lcpserver.my.org/licenses?page=2&per_page=50
- Response:

HTTP Status Code	Description
200	returns license. sample response license is provided in SampleLicenseDescription.txt file attached to this document.
400	Bad Request (Invalid Pagination Parameters)

FIGURE 4 LIST LICENSES RESPONSE SAMPLE

```
[
  {
    "provider": "http://www.iRead.com",
    "id": "b19a0012-0af0-4a9a-bab7-fb09a5849e4f",
    "issued": "2022-03-23T12:04:15Z",
    "user": {
      "id": "d9f298a7-7f34-49e7-8aae-4378ecb1d597"
    },
    "rights": {
      "start": "2022-03-23T11:24:34Z",
      "end": "2022-03-26T11:24:34Z",
      "start": "2016-11-04T01:08:15+01:00",
      "end": "2016-11-25T01:08:15+01:00"
    }
  }
]
```

6. List All Existing content Licenses:

- GET <LCPbaseURL>/contents/<content_id>/licenses{page?, per_page?}
- List licences associated with specific content ordered by creation Date (issue date)
- Page numbering is 1-based (starts with 1 not 0), and the default per_page is 30
- Response:

HTTP Status Code	Description
200	returns license. sample response license is provided in SampleLicenseDescription.txt file attached to this document.
400	Bad Request (Invalid Pagination Paramters)

FIGURE 5 LIST CONTENT LICENSES RESPONSE SAMPLE

```
[
  {
    "provider": "http://www.iRead.com",
    "id": "b19a0012-0af0-4a9a-bab7-fb09a5849e4f",
    "issued": "2022-03-23T12:04:15Z",
    "user": {
      "id": "d9f298a7-7f34-49e7-8aae-4378ecb1d597"
    },
    "rights": {
      "start": "2022-03-23T11:24:34Z",
      "end": "2022-03-26T11:24:34Z",
      "start": "2016-11-04T01:08:15+01:00",
      "end": "2016-11-25T01:08:15+01:00"
    }
  }
]
```

7. Update License:

- PATCH <LCPbaseURL>/licenses/<license_id>
- This method called from the License Status server (for lending return or renewal).
- Request:

Parameter Name	Description	Type
provider	provider identifier (a URL).	String
user	Json object that contains user Information	Object
user/email	User Email, optional.	String
user/ encrypted	Encrypted values the provider wants to embed in the license, optional.	String array
rights	the set of rights associated with the license, Optional	Object
rights/start	Start Date of license	DateTime
rights/end	End Date of license	DateTime
rights/ print	Maximum number of pages that can be printed over the lifetime of the license, if zero copy will not be allowed, to be unlimited don't send it	Integer
rights/ copy	Maximum number of characters that can be copied over the lifetime of the license, if zero copy will not be allowed, to be unlimited don't send it	Integer

FIGURE 6 UPDATE LICENSE REQUEST SAMPLE

```
{
  "provider": "http://www.imaginaryebookretailer.com",
  "user": {
    "email": "user@mymail.com",
    "encrypted": ["email"]
  },
  "rights": {
    "print": 10,
    "copy": 2048,
    "start": "2016-11-04T01:08:15+01:00",
    "end": "2016-11-25T01:08:15+01:00"
  }
}
```

- Response:

HTTP Status Code	Description
200	Success
400	Bad Request
404	License not found

• LSD SERVER (LICENSE STATUS DOCUMENT)

▪ GENERAL NOTES:

- **Some** API Endpoints provided by LSD Server are private which means it requires authentication via (Basic Authorization). It is implemented by Sending the Username and password in the request header after applying base64 encoding.
 - Encoding String is "{Username}:{Password}"
 - Ex test:123456 generates "dGVzdDoxMjM0NTYg"
- A sample complete license status document with description and clarification of its properties is attached to this document, Link:
 - [SampleLicenseStatusDocumentDescription](#)
- All Sent and received Date Time are ISO 8601 Formatted, which is compatible with C# DateTime Data Type.
- Each time the user opens the publication, if the license contains a link to a status document and the device is online, then the status document is requested.
- If the device wasn't registered before for this license and if the status document contains an registration link, then the device calls the registration link
- If the server rejected the registration, the device must not let the user open the book before a successful attempt is made.
- If the license was updated, the device must downloads and replace the old license with the updated license.
- License statuses are predefined they are the following:

Value	Description
ready	The License 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 returned when it was previously active, it has been invalidated by the User.
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.

- License events type are predefined they are the following:

Value	Description
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.
register	Signals a successful registration event by a device.

■ INTEGRATION API ENDPOINTS

1. Create a license status document:

- PUT <LSDbaseURL>/licenses
- This Method is private, needs authentication.
- This method is called by the License Server after generation of a license.
- Request: a complete License that is generated
- Response:

HTTP Status Code	Description
201	Created
400	Bad Request
500	internal server error

2. Get license status document:

- GET <LSDbaseURL>/licenses/<license_id>/status
- This method generates and returns a license status document, using license Id.
- Response:

HTTP Status Code	Description
200	Returns the requested license status
404	License not found

3. Register Device:

- POST <LSDbaseURL>/licenses/<license_id>/register{?id, name}
- Activate a device for a given license where license status must be **ready or active**.
- Parameters are **mandatory** where id is the device identifier in the form of a UUID while name is the device name in the form of a URL-encoded string. Also note parameters maximum length is 255 bytes
- Response:

HTTP Status Code	Description
200	Returns the updated license status
400	Bad Request when parameters are not provided
403	Forbidden, parameters are invalid
404	License not found
500	Internal server error

4. Return License:

- PUT <LSDBaseURL>/licenses/<license_id>/return{?id, name}
- It checks that the calling device is activated, then modifies the end date associated with the given license, it will be set to "now".
- This method calls the update license "rights" method of the LCP server.
- Parameters are **optional** where id is the device identifier in the form of a UUID while name is the device name in the form of a URL-encoded string.
- When the license status is "ready" the status will be changed to "cancelled", while if it is "active" it will be changed to "returned". And a new event will be added to the status document
- Response:

HTTP Status Code	Description
200	License updated and returns License Status Document as described in the sample also it adds an event for return
400	Bad Request
403	Rejected, license is already returned or expired.
404	License not found
500	Internal server error

5. Renew License:

- PUT <LSDBaseURL>/licenses/<license_id>/renew{?end, id, name}
- The method checks that the calling device is activated, then modifies the end date associated with the license.
- This method calls the update license "rights" method of the LCP server.
- Parameters are **optional** where id is the device identifier in the form of a UUID while name is the device name in the form of a URL-encoded string. End is the requested end date of license in **(YYYY-MM-DDThh:mm:ssTZD)** Format.
- If end is not provided, the default number of additional days will be (renew_days) found in the configuration file.
 - Ex if end date is 2022-03-01 and renew_days is 7 then after renewal it will be 2022-03-07
 - **Important Notes:**
 - License status document contains "potential_rights/end" property, which is initialized only once at the time the license is added to the LSD server by adding "renting_days" found in config file to issue date.
 - Renew is not possible if the license requested end date exceeds "potential_rights/end" date
 - On the previous example if you tried to renew the license again and "renting_days" value is 10 so the "potential_rights/end" is 2022-03-10 and the requested end date is 2022-03-14 the Endpoint will return 403
- Response:

HTTP Status Code	Description
200	returns license. sample response license is provided in SampleLicenseDescription.txt file attached to this document.
400	Bad Request (Invalid Pagination Parameters)
403	Forbidden, Incorrect renewal period
404	License not found
500	Internal server error

6. Get Licenses Registered Devices Count:

- GET <LSDBaseURL>/licenses{?devices, page, per_page}
- This method is a private method.
- It is used to filter the licenses that have been used by a large number of devices.
- The devices parameter represents the minimum number of devices and it is optional.
- Response:

HTTP Status Code	Description
200	Success, returns array of partial licenses
401	Unauthorized
400	Bad Request (Invalid Pagination Parameters)

FIGURE 7 LICENSES REGISTERED DEVICES COUNT SAMPLE RESPONSE

```
[
  {
    "id": "234-5435-3453-345354",
    "status": "active",
    "message": "Your license is currently active and has been used on at least one device.",
    "updated": {
      "license": "2014-08-05T00:00:00Z",
      "status": "2014-08-08T00:00:00Z"
    },
    "device_count": "100"
  }
]
```

7. Get License Registered Devices:

- GET <LSDBaseURL>/licenses /<license_id>/registered
- This method is a private method.
- It is used to get all devices registered to a specific license.
- Response:

HTTP Status Code	Description
200	Success, returns array of partial licenses
401	Unauthorized
404	License not found
500	Internal sever error

FIGURE8 LICENSES REGISTERED DEVICES COUNT SAMPLE RESPONSE

```
{
  "id": "423f4f9a-585e-4fde-811c-2f8b5d018af6",
  "devices": [
    {
      "id": "39d98920-6c8f-4601-9a08-195191d7cfbc",
      "name": "alysalem",
      "timestamp": "2022-03-24T10:38:51Z"
    }
  ]
}
```

8. Revoke/Cancel License:

- PATCH <LSDbaseURL>/licenses/<license_id>/status
- This method is a private method.
- This methods **revokes (after use)** or **cancels (before use)** a license.
- The LSD server internally calls the LCP server and updates the expiration time to "now".
- A "cancel" command returns a 400 error if the current status is not "ready" also "revoke" returns an error if the current status is not "active".
- Request:

FIGURE 9 REVOKE OR CANCEL SAMPLE REQUEST

```
{
  "status": "revoked",
  "message": "Your license has been revoked because it has been used on 589 devices in less than
a week.",
}
```

- Response:

HTTP Status Code	Description
200	Success, returns array of partial licenses
400	Bad Request, the new status is not compatible with current status.
401	Unauthorized
404	License not found
500	Internal server error