



**VisionLabs**  
MACHINES CAN SEE

# **VisionLabs LUNA SDK Licensing**

**written for LUNA SDK Mobile iOS version 5.30.2**

## Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>General information</b>	<b>4</b>
<b>3</b>	<b>Licensed features</b>	<b>5</b>
<b>4</b>	<b>License activation</b>	<b>7</b>
4.1	Online activation . . . . .	7
4.1.1	Online activation process . . . . .	7
4.2	License file parameters description . . . . .	7
4.3	Parameter details . . . . .	9
4.3.1	Server, EID, and ProductID . . . . .	9
4.3.2	Filename . . . . .	9
4.3.3	Test . . . . .	9
4.3.4	ActivationMode . . . . .	9
4.3.5	ContainerMode . . . . .	9
4.3.6	FingerprintSource . . . . .	10
4.3.7	ConnectionTimeout . . . . .	10
4.3.8	ServerRetriesCount . . . . .	10
4.3.9	licenseModel . . . . .	10

## 1 Introduction

This document includes information about LUNA SDK licensing. This information is required for any engineer or developer who wants to utilize LUNA SDK features.

Here you can find:

- General information about LUNA SDK licensing
- The list of licensed features
- Online and offline license activation description

## 2 General information

FaceEngine supports per-features node-locked licensing for every supported platform. This means that a final customer version of FaceEngine might be customized by activating and deactivating the licensed features. For that reason, not all algorithms and modules described in this book might be available to you.

Each SDK instance should be activated on every device. License, which was activated for one device, could not be used on some other device.

Interface for License objects `ILicense` (see file `ILicense.h`) gives you possibility to:

- check, if license is already activated;
- save license for next usage to some file;
- load license from file;
- check if some feature of FaceEngine is available for this license.

Typical usage scenario:

- Create a `IFaceEngine` object (see file `FaceEngine.h`);
- Get license pointer through `fsdk::IFaceEngine::getLicense`;
- Make activation for that license object through `fsdk::activateLicense`. This method requires full or relative path to the “license.conf” file.

### 3 Licensed features

To utilize license features in the code, please refer to the `LicenseFeature` enum defined in the `ILicense.h` file. Note that certain features may not be available on all platforms.

The table below lists features and their associated estimators available on mobile platforms:

Feature	Description	Available instances
BestShot	Enables you to select the best shot from the provided images.	IAGSEstimator
		IBestShotQualityEstimator
		IHeadPoseEstimator
Descriptor	Enables you to extract face features of a particular person and work with them. You can create a maximum of 1000 descriptors on mobile platforms.	IQualityEstimator
		IDescriptor
		IDescriptorBatch
Detection	Enables you to find a person's face in a frame.	IDescriptorExtractor
		IDescriptorMatcher
		IDetector
FaceFeatures	Enables you to estimate face features.	IEyeEstimator
		IFaceOcclusionEstimator
		IGlassesEstimator
Liveness	Enables you to get the best shot from a video.	IMouthEstimator
		IBestShotMobile
MedicalMaskDetection	Enables you to detect a medical face mask on the face in the source image.	IMedicalMaskEstimator

Feature	Description	Available instances
MobileLiveness	Enables you to predict whether a person's face is real or fake (photo, printed image).	ILivenessOneShotRGBEstimator
NIRLiveness	Enables you to perform the NIR Liveness estimation.	INIRLivenessEstimator
TrackEngine	Enables you to track a person in a video.	ITrackEngine

**Important:** Other features such as Attributes, Emotions, Liveness, DescriptorIndex, HumanDetection are not available on mobile platforms. See [FeatureMapMobile.htm](#) for additional information about these features.

If the license for the selected feature is invalid, the factory instantiation method will return a result with an error code and an empty object. For example, method `IFaceEngine::createAGSEstimator` will return `ResultValue` with `FSDKError::LicenseError` empty `IAGSEstimatorPtr` object in case if `LicenseFeature::BestShot` is not available.

## 4 License activation

This section describes license activation.

License activation should be performed after the distribution package is unarchived.

The generated license can only be used on the device where it was activated as the fingerprint of the device is used for its creation.

**NOTE!** Always remember that incorrect config may huck the things up very badly. Pay attention to what you configure and how. Always double-check what you deploy.

### 4.1 Online activation

Online activation is performed when you have an Internet connection on the device where the LUNA SDK license is activated.

Licensing configuration options are specified via the “[license.conf](#)” file which is an XML document with special tag formatting.

You can find the “license.conf” file in the “Frameworks/fsdk.framework/data” directory in the distribution package.

This file is mandatory for license activation. You must fill it with the correct values before launching LUNA SDK.

Fill in this file carefully. Incorrectly entered data may cause problems with activation on the device.

#### 4.1.1 Online activation process

The activation process is as follows:

- Request the **Server**, **EID**, and **ProductID** from VisionLabs
- Go to the “Frameworks/fsdk.framework/data” directory
- Open the “license.conf” file
- Enter the received parameters
- Save changes in the “license.conf” file

The license key will be generated and saved to the “Frameworks/fsdk.framework/data” directory.

Now you can use LUNA SDK.

### 4.2 License file parameters description

License activation and next processing requires parameters listed below.

Parameter	Description	Type	Default value
Server	Activation server URL	"Value::String"	(empty)
EID	Entitlement ID	"Value::String"	(empty)
ProductID	Product ID	"Value::String"	(empty)
Filename	Default license filename	"Value::String"	license.dat
Test	If run in a test environment	"Value::Int1"	0
ActivationMode	Activation mode	"Value::String"	Hardware
FingerprintSource	Generation option	"Value::Int1"	0
ContainerMode	If run in container	"Value::Int1"	0
ConnectionTimeout	Request timeout (in seconds)	"Value::Int1"	15
ServerRetriesCount	Retry attempts count	"Value::Int1"	3
licenseModel	License type	"Value::Int1"	1

For more information on the parameters, see [Parameter details](#).

By default, the file is created in the *"Frameworks/fsdk.framework/data"* directory. The file has a binary format. On subsequent launches of the product on the same device, the license will be automatically read from this file.

An example of the "license.conf" file:

```
?xml version="1.0"?>
<settings>
  <section name="Licensing::Settings">
    <param name="Server" type="Value::String" text=""/>
    <param name="EID" type="Value::String" text=""/>
    <param name="ProductID" type="Value::String" text=""/>
    <param name="Filename" type="Value::String" text="license.
      dat"/>
    <param name="Test" type="Value::Int1" x="1" />
    <param name="ActivationMode" type="Value::String" text="
      Hardware"/>
    <param name="FingerprintSource" type="Value::Int1" x="1"/>
    <param name="ContainerMode" type="Value::Int1" x="1"/>
    <param name="ConnectionTimeout" type="Value::Int1" x="15"/>
    <!-- Currently, the available values are: 1 and 2. Default
      is 1 -->
  </section>
</settings>
```



```
        <param name="licenseModel" type="Value::Int1" x="2" />
    </section>
</settings>
```

## 4.3 Parameter details

### 4.3.1 Server, EID, and ProductID

These parameters are mandatory for license activation. Obtain them from VisionLabs.

The EID parameter is required for both online and offline activation.

The Server and ProductID are only required for online activation.

### 4.3.2 Filename

The `Filename` parameter specifies the name of the file where the license will be saved after activation.

The maximum length is 64 characters.

**Important:** Do not change the default name, as this may lead to unexpected behavior during activation or runtime.

### 4.3.3 Test

Applies to FIT license only.

The `Test` parameter determines whether the system operates with a test entitlement in terms of FIT. It specifies whether the license being used is a test license or a production license. Possible values:

- 0 - The system operates with a production license.
- 1 - The system operates with a test license.

### 4.3.4 ActivationMode

The `ActivationMode` parameter specifies which path to choose when retrieving device UID during license activation stage. Possible values:

- Hardware
- JNI

### 4.3.5 ContainerMode

The `ContainerMode` parameter has the following possible values:

- 0 - Ignore any virtual network interfaces in the formation of the fingerprint.

- 1 - Take into account all virtual interfaces in the formation of a fingerprint.

The 1 value is used while running in docker container due to physical network interface is not available by default. Although, you can make physical interface available with `--net=host` and set `ContainerMode` to 0. In this case you can be sure fingerprint will not be changed among containers. For details, see “Licensing in docker container”.

#### 4.3.6 FingerprintSource

The `FingerprintSource` parameter has the following possible values:

- 0 - Default. First fingerprint generation option.
- 1 - Second fingerprint generation option.

#### 4.3.7 ConnectionTimeout

The `ConnectionTimeout` specifies the maximum time, in seconds, allowed for the transfer operation during license activation.

Behavior:

- A value of 0 disables the timeout, allowing the operation to proceed indefinitely.
- Negative values or values exceeding the maximum limit (300 seconds) are invalid.

While longer timeouts reduce the risk of aborting valid operations, they may increase activation delays. Balance this parameter based on your network conditions.

#### 4.3.8 ServerRetriesCount

The `ServerRetriesCount` parameter defines the number of retry attempts for connecting to the activation server. It helps handle transient failures, improving application stability.

The parameter must be a non-negative integer, with a maximum value of 100 attempts.

#### 4.3.9 licenseModel

The `licenseModel` defines the license to be used. Possible values:

- 1 - The system uses FIT license.
- 2 - The system uses Zeus license.