# The OpenCV Manager Manual

*Release 2.4.3*

December 25, 2012

# CONTENTS

CHAPTER

ONE

# ANDROID OPENCV MANAGER

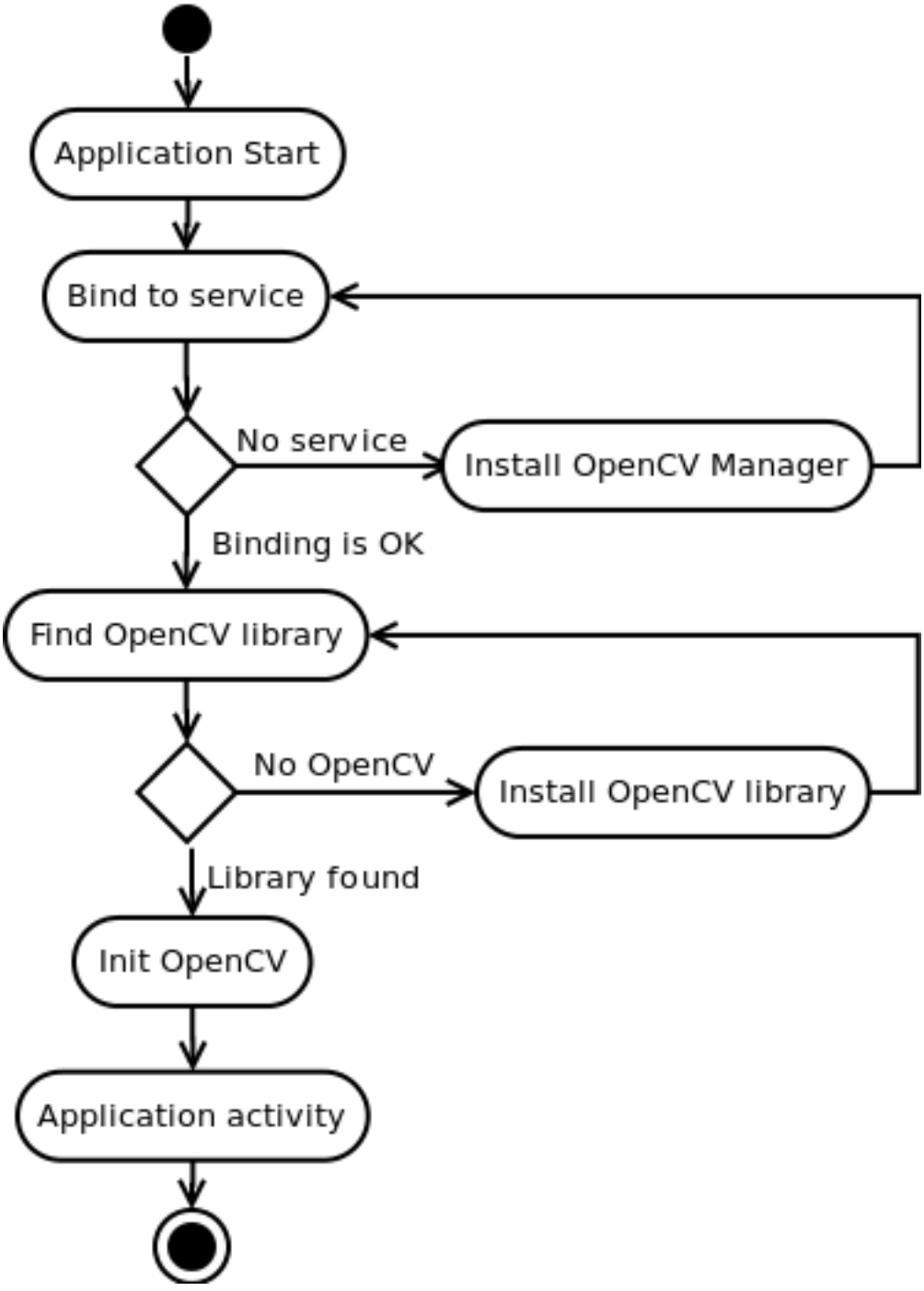
Contents:

## 1.1 Introduction

OpenCV Manager is an Android service targeted to manage OpenCV library binaries on end users devices. It allows sharing the OpenCV dynamic libraries between applications on the same device. The Manager provides the following benefits:

1. Less memory usage. All apps use the same binaries from service and do not keep native libs inside themselves;
2. Hardware specific optimizations for all supported platforms;
3. Trusted OpenCV library source. All packages with OpenCV are published on Google Play market;
4. Regular updates and bug fixes;

## Usage model for end user




First OpenCV app:

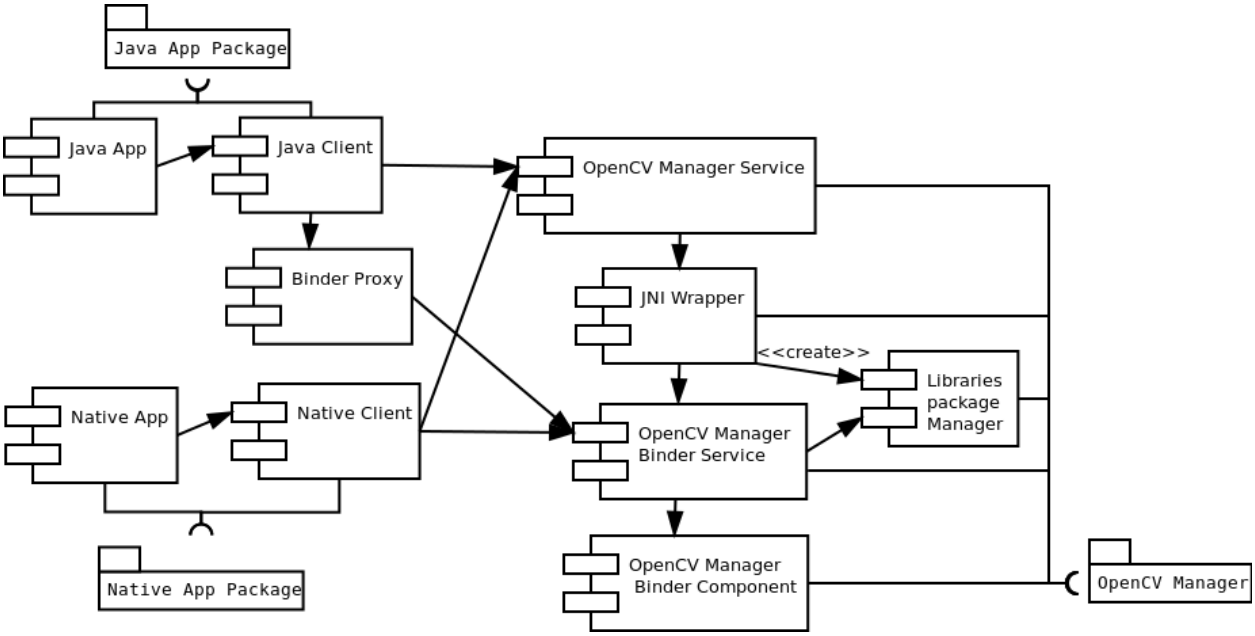
1. Any OpenCV-dependent app is installed from Google Play marketplace or manually;
2. At the first launch, it suggests installation of OpenCV Manager;
3. Then OpenCV Manager is downloaded and installed, using the Google Play application.

4. When Manager has been started, the application suggests installation of OpenCV library for the target device architecture if it is necessary;
5. After the installation is finished, the app may be launched.

Subsequent launches of OpenCV apps:

1. Any OpenCV-dependent app is installed from Google Play market or manually;
2. At the first launch, the app starts as usually;
3. If the selected OpenCV version is not installed, OpenCV Manager suggests installing OpenCV library for the target device through Google Play marketplace;
4. After the installation is finished, the app may be launched.

### Architecture of OpenCV Manager




## 1.2 Manager Workflow

### How to select the proper version of OpenCV Manager

Since version 1.7 several packages of OpenCV Manager are built. Every package is targeted for some specific hardware platform and includes corresponding OpenCV binaries. So, in most cases OpenCV Manager uses built-in version of OpenCV. Separate package with OpenCV binaries is currently used in a single rare case, when an ARMv7-A processor without NEON support is detected. In this case an additional binary package is used. The new package selection logic in most cases simplifies OpenCV installation on end user devices. In most cases OpenCV Manager may be installed automatically from Google Play.

If Google Play is not available (i.e. on emulator, developer board, etc), you can install it manually using adb tool:

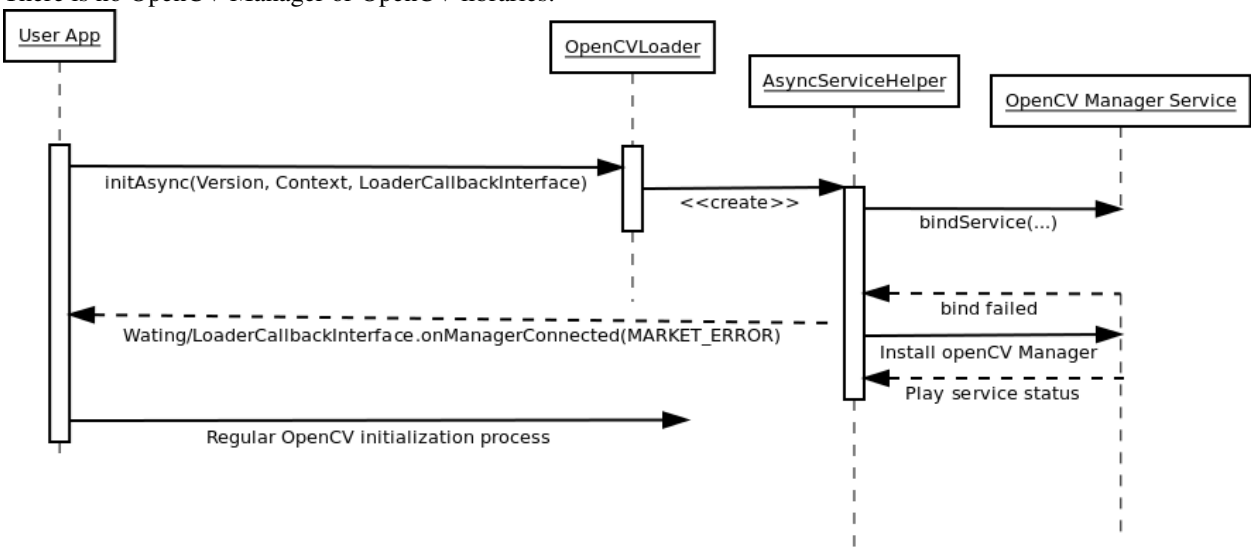
```
adb install OpenCV-2.4.3-android-sdk/apk/OpenCV_2.4.3.2_Manager_2.4_<platform>.apk
```

Use the table below to determine proper OpenCV Manager package for your device:

| Hardware Platform | Android ver. | Package name |
|---|---|---|
| armeabi-v7a (ARMv7-A + NEON) | >= 2.3 | OpenCV_2.4.3.2_Manager_2.4_armv7a-neon.apk |
| armeabi-v7a (ARMv7-A + NEON) | = 2.2 | OpenCV_2.4.3.2_Manager_2.4_armv7a-neon-android8.apk |
| armeabi (ARMv5, ARMv6) | >= 2.3 | OpenCV_2.4.3.2_Manager_2.4_armeabi.apk |
| Intel x86 | >= 2.3 | OpenCV_2.4.3.2_Manager_2.4_x86.apk |
| MIPS | >= 2.3 | OpenCV_2.4.3.2_Manager_2.4_mips.apk |

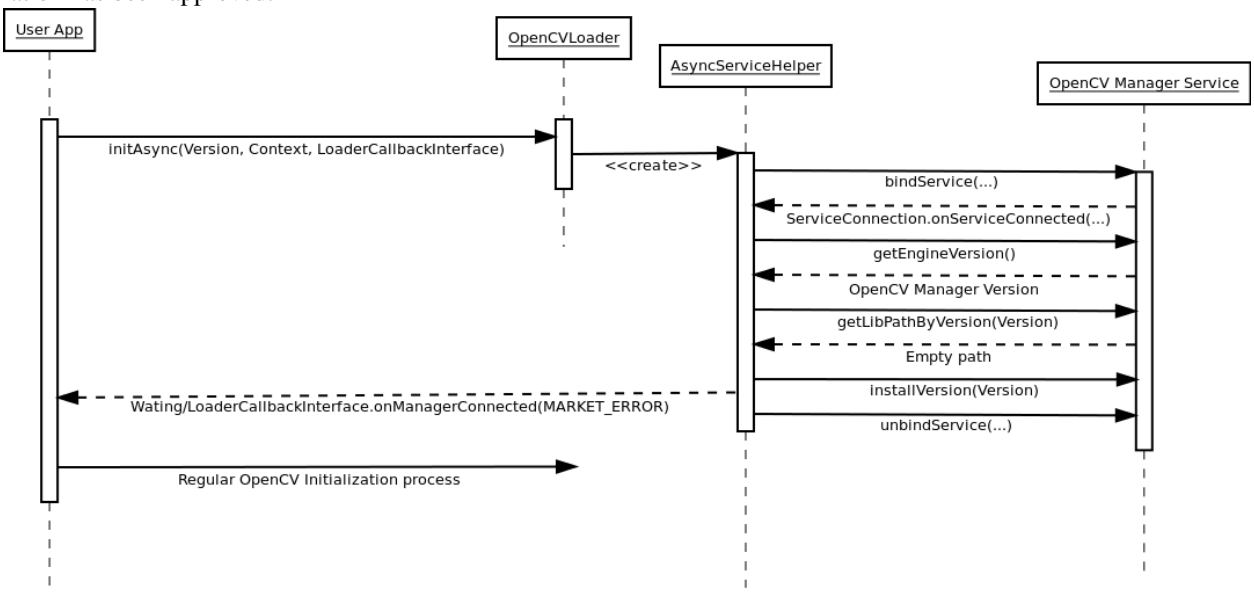
## First application start

There is no OpenCV Manager or OpenCV libraries:

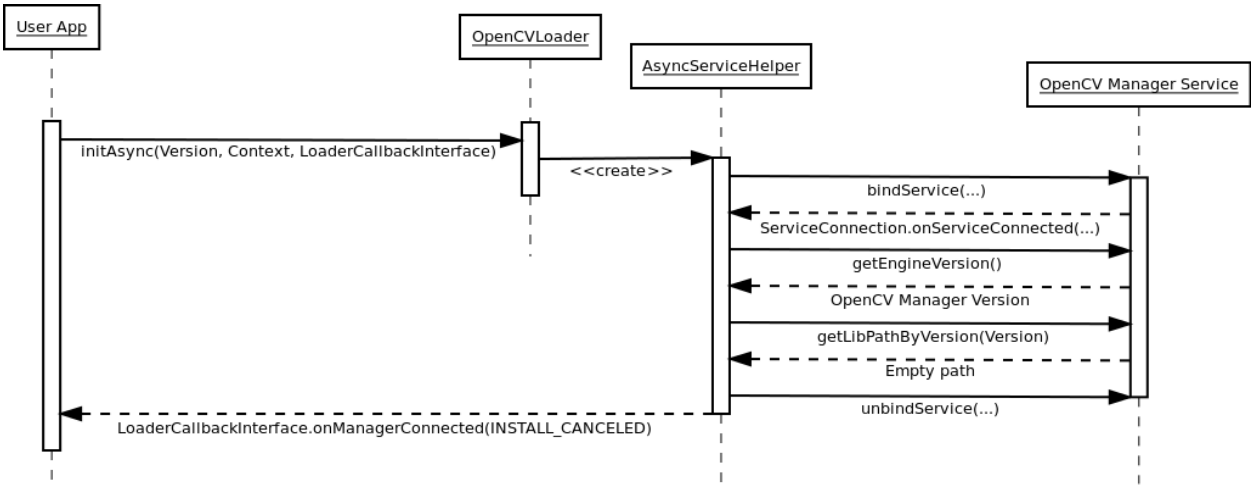



## Additional library package installation

There is an OpenCV Manager service, but it does not contain appropriate OpenCV library. If OpenCV library installation has been approved:

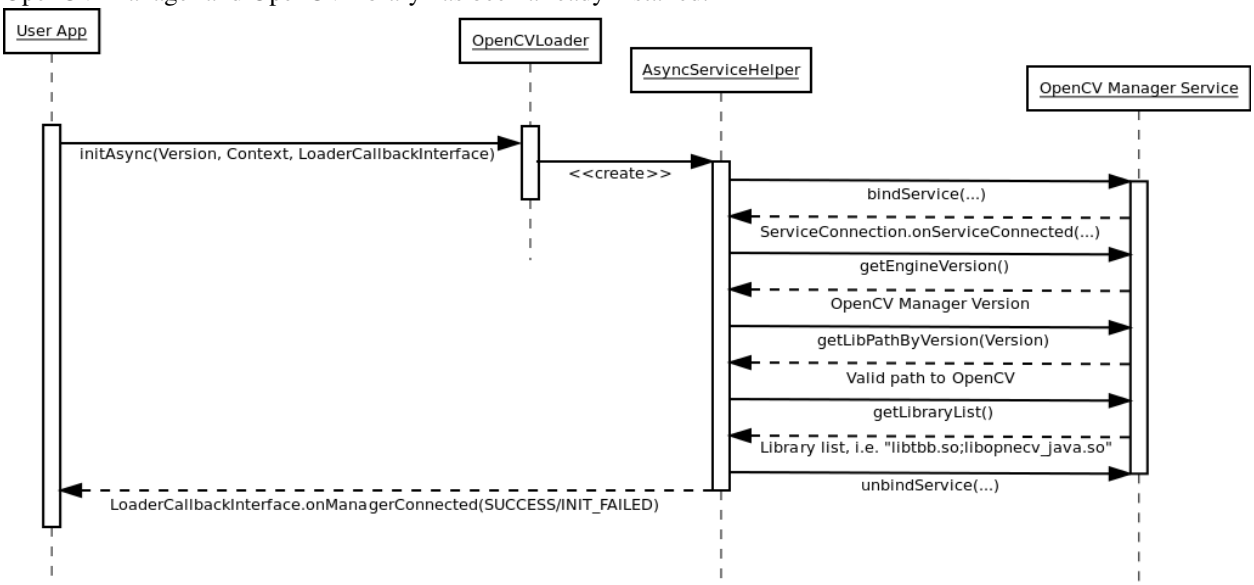



If OpenCV library installation has been cancelled:




### Regular application start

OpenCV Manager and OpenCV library has been already installed.




## 1.3 Java OpenCV Loader

**class `OpenCVLoader`**

Helper class provides common initialization methods for OpenCV library.

### boolean initDebug()

**`static boolean initDebug()`**

Loads and initializes OpenCV library from within current application package. Roughly it is analog of `system.loadLibrary("opencv_java")`.

**Return type** boolean;

**Returns** returns true if initialization of OpenCV was successful.

---

**Note:** This method is deprecated for production code. It is designed for experimental and local development purposes only. If you want to publish your app use approach with async initialization.

---

### boolean initAsync()

**static boolean initAsync(String Version, Context AppContext, LoaderCallbackInterface Callback)**
Loads and initializes OpenCV library using OpenCV Manager.

**Parameters**

**Version** – OpenCV Library version.

**AppContext** – application context for connecting to the service.

**Callback** – object, that implements `LoaderCallbackInterface` for handling connection status (see `BaseLoaderCallback`).

**Return type** boolean;

**Returns** returns true if initialization of OpenCV starts successfully.

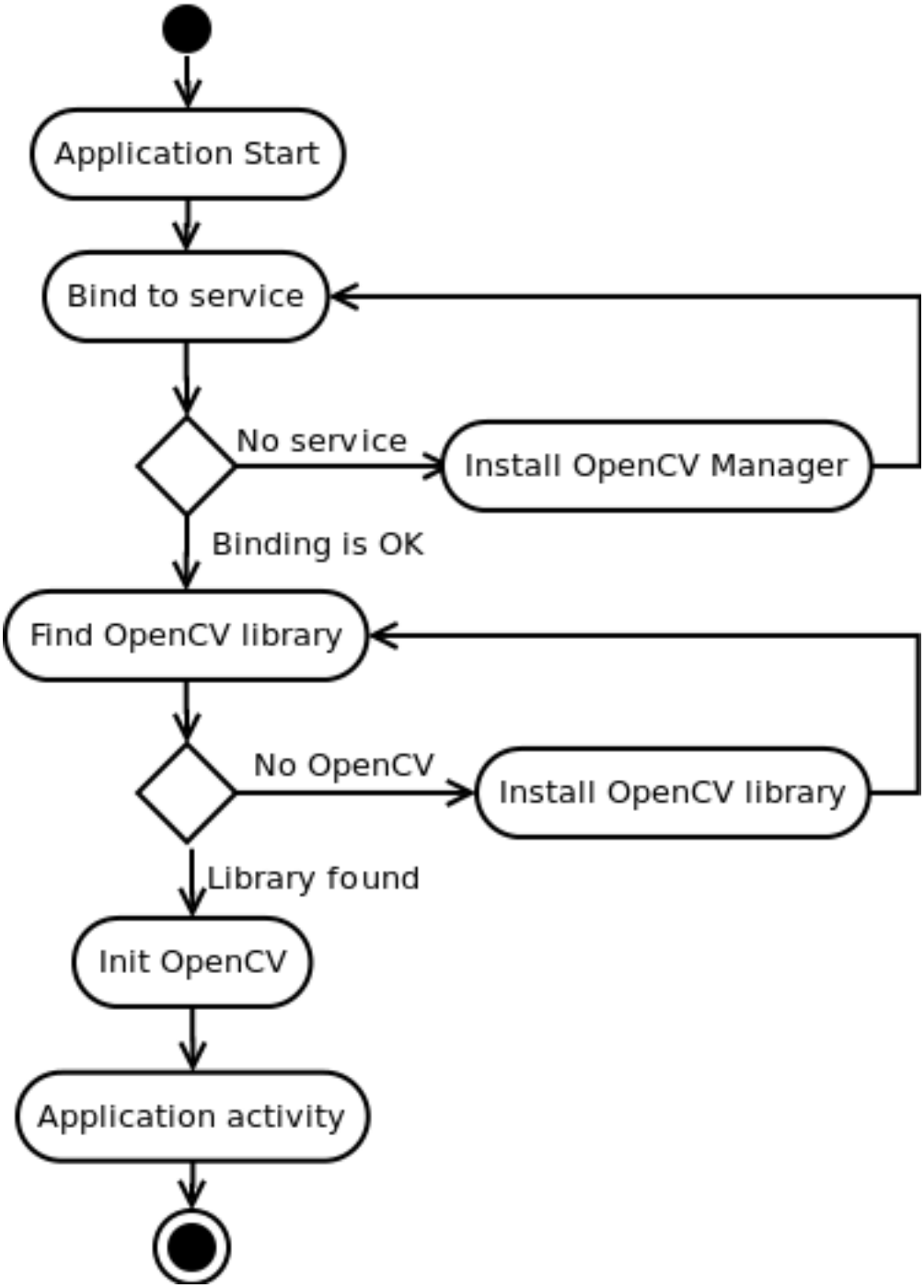
### OpenCV version constants

**OPENCV_VERSION_2_4_2**
OpenCV Library version 2.4.2

**OPENCV_VERSION_2_4_3**
OpenCV Library version 2.4.3

## 1.4 Base Loader Callback Interface Implementation

**class BaseLoaderCallback**
Basic implementation of `LoaderCallbackInterface`. Logic of this implementation is well-described by the following scheme:




## Using in Java Activity

There is a very base code snippet implementing the async initialization with `BaseLoaderCallback`. See the "15-puzzle" OpenCV sample for details.

```
1 public class MyActivity extends Activity implements HelperCallbackInterface
2 {
3 private BaseLoaderCallback mOpenCVCallBack = new BaseLoaderCallback(this) {
```

```
    @Override
    public void onManagerConnected(int status) {
      switch (status) {
        case LoaderCallbackInterface.SUCCESS:
        {
           Log.i(TAG, "OpenCV loaded successfully");
           // Create and set View
           mView = new puzzle15View(mAppContext);
           setContentView(mView);
        } break;
        default:
        {
           super.onManagerConnected(status);
        } break;
      }
    }
};

/** Call on every application resume **/
@Override
protected void onResume()
{
    Log.i(TAG, "Called onResume");
    super.onResume();

    Log.i(TAG, "Trying to load OpenCV library");
    if (!OpenCVLoader.initAsync(OpenCVLoader.OPENCV_VERSION_2_4_3, this, mOpenCVCallBack))
    {
        Log.e(TAG, "Cannot connect to OpenCV Manager");
    }
}
```

### Using in Service

Default `BaseLoaderCallback` implementation treats application context as `Activity` and calls `Activity.finish()` method to exit in case of initialization failure. To override this behavior you need to override `finish()` method of `BaseLoaderCallback` class and implement your own finalization method.

## 1.5 Loader Callback Interface

**class `LoaderCallbackInterface`**
Interface for a callback object in case of asynchronous initialization of OpenCV.

### void onManagerConnected()

**`void onManagerConnected(int status)`**
Callback method that is called after OpenCV library initialization.

**Parameters**

**status** – status of initialization (see "Initialization Status Constants" section below).

### void onPackageInstall()

**void onPackageInstall(InstallCallbackInterface Callback)**
Callback method that is called in case when package installation is needed.

**Parameters**

**callback** – answer object with `install` and `cancel` methods and package description.

### Initialization status constants

**SUCCESS**
OpenCV initialization finished successfully

**MARKET_ERROR**
Google Play (Android Market) application cannot be invoked

**INSTALL_CANCELED**
OpenCV library installation was cancelled by user

**INCOMPATIBLE_MANAGER_VERSION**
Version of OpenCV Manager is incompatible with this app. Manager update is needed.

**INIT_FAILED**
OpenCV library initialization failed

## 1.6 Install Callback Interface

**class InstallCallbackInterface**
Callback interface for package installation or update.

### String getPackageName()

**String getPackageName()**
Get name of a package to be installed.

**Return type** string;

**Returns** returns package name, i.e. “OpenCV Manager Service” or “OpenCV library”.

### void install()

**void install()**
Installation of package has been approved.

### void cancel()

**void cancel()**
Installation of package has been cancelled.

## void wait_install()

**void wait_install()**
Wait for package installation.

CHAPTER

TWO

# JAVA API

Java API reference (JavaDoc): external link.