# OnStream MediaPlayer+ SDK Project Setup

# for Android Platforms SDK Version 3.5 and later



March, 2013

Version 1.1 201303001



# **Copyright/Confidentiality Notice**

© 2013 VisualOn, Inc. All rights reserved.

VisualOn, Inc., 4675 Stevens Creek Blvd, Santa Clara, CA 95051, USA

#### VisualOn Trademarks

Trademarks and service marks of VisualOn, Inc. (VisualOn) contained in this document are attributed to VisualOn with the appropriate symbol. For queries regarding VisualOn's trademarks, contact the corporate legal department at the address above or call 408.244.8801.

VisualOn® OnStream™

All other trademarks are the property of their respective holders.

# CONFIDENTIALITY NOTICE

No part of this publication may be reproduced in whole or in part by any means (including photocopying or storage in an information storage/retrieval system) or transmitted in any form or by any means without prior written permission from VisualOn, Inc. (VisualOn).

Information in this document is subject to change without notice and does not represent a commitment on the part of VisualOn. The information contained herein is the proprietary and confidential information of VisualOn or its licensors, and is supplied subject to, and may be used only by VisualOn's customer in accordance with, a written agreement between VisualOn and its customer. Except as may be explicitly set forth in such agreement, VisualOn does not make, and expressly disclaims, any representations or warranties as to the completeness, accuracy or usefulness of the information contained in this document. VisualOn does not warrant that use of such information will not infringe any third party rights, nor does VisualOn assume any liability for damages or costs of any kind that may result from use of such information.

RESTRICTED RIGHTS LEGEND Use, duplication, or disclosure by the Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013.

UNPUBLISHED This document contains unpublished confidential information and is not to be disclosed or used except as authorized by written contract with VisualOn. Rights reserved under the copyright laws of the United States.



# **Table of Contents**

1	ABOUT THIS MANUAL	1
	1.1 Overview	1
	1.2 Scope	1
	1.3 REVISION HISTORY	1
	1.4 RELATED DOCUMENTS	1
	1.5 ABBREVIATIONS	2
	1.6 Typographic Conventions	2
2	ANDROID DEVELOPMENT ENVIRONMENT SETUP	3
	2.1 Android SDK/ADT Download and Installation	3
	2.2 ANDROID NDK DOWNLOAD AND INSTALLATION	3
	2.3 ECLIPSE DOWNLOAD AND INSTALLATION	3
3	SDK PACKAGE CONTENTS	4
	3.1 OnStream MediaPlayer+ SDK Release Structure	4
4	SDK PROJECT SETUP	5
	4.1 DECOMPRESS THE SDK PACKAGE	5
	4.2 CREATE/OPEN AN ANDROID PROJECT	5
	4.3 MODIFY THE ANDROID MANIFEST	7
	4.4 ADD THE SDK LIBRARIES	8
	4.4.1 Create a libs Folder	8
	4.4.2 Import Java Libraries	8
	4.4.3 Create a libs/armeabi Folder	10
	4.4.4 Import C/C++ Shared Libraries	10
	4.4.5 Update Project Properties	11
	4.5 IMPORT THE SDK PACKAGES	12
	4.6 ADD THE ASSETS	13
	4.6.1 Create an assets Folder	
	4.6.2 Import Assets	14
5	NEXT STEPS	15



# **Table of Figures**

FIGURE 3-1: ONSTREAM MEDIAPLAYER+ SDK DIRECTORY STRUCTURE	4
FIGURE 4-1: ECLIPSE NEW PROJECT DIALOG BOX	5
FIGURE 4-2: ECLIPSE NEW ANDROID APPLICATION DIALOG BOX	6
FIGURE 4-3: ECLIPSE PACKAGE EXPLORER VIEW WITH SAMPLE PROJECT	7
FIGURE 4-4: UPDATED ANDROIDMANIFEST.XML FILE IN ECLIPSE EDITOR WINDOW	8
FIGURE 4-5: ECLIPSE IMPORT DIALOG BOX	9
FIGURE 4-6: SELECTED JAVA LIBRARIES	9
FIGURE 4-7: SELECTED JAVA LIBRARIES	10
FIGURE 4-8: PROPERTIES DIALOG BOX	11
FIGURE 4-9: JAR SELECTION DIALOG BOX	12
FIGURE 4-10: FILE OPERATION DIALOG BOX	14



# 1 About This Manual

#### 1.1 OVERVIEW

This manual describes the project setup and installation of OnStream<sup>™</sup> MediaPlayer+ SDK (or "SDK") releases for Android platforms. This document includes the following topics:

- Android Development Environment Setup
- SDK Packages
- SDK Project Setup

# 1.2 SCOPE

This manual is intended for Android developers who need to create a flexible and high-performance media player that supports playback of live or VOD streaming, progressive download, and local media sources.

Android developers are assumed to be familiar with: the Android SDK/ADT; the Eclipse IDE; the Java Native Interface (JNI) and Android NDK; and the Java and C/C++ programming languages.

# 1.3 REVISION HISTORY

Rev	Product Version	Date	Description
1.1	V3.5	2013-03-25	Document Creation

### 1.4 RELATED DOCUMENTS

The following documents (included with your installation package) provide additional information related to this user guide:

- OnStream MediaPlayer+ Introduction and Documentation Guide for Android Platforms
- OnStream MediaPlayer+ Player SDK Integration Guide for Android Platforms
- OnStream MediaPlayer+ Sample Player Tutorial for Android Platforms



# 1.5 ABBREVIATIONS

Acronym	Description
API	Application Programming Interface
DRM	Digital Rights Management
IDE	Integrated Development Environment
JNI	Java Native Interface
OSMP+	OnStream MediaPlayer+
SDK	Software Development Kit

# 1.6 Typographic Conventions

- **Directory Contents** are shown in "Calibri" font in blue.
- File and Directory Names are shown in "Calibri" font in *blue italics*.
- File Contents and Source Code are shown single-spaced in "Courier New" font.
- Menu Options, Commands, and Windows/Views are shown single-spaced in bold.
- **Project and Document Titles** are shown in *italics*.

# Examples:

- 1. Select the **Select root directory** radio button, and input or browse to <SDK\_INSTALL\_DIR>\Android\SamplePlayer.
- 2. Under **Projects**, make sure that the *STSamplePlayer* checkbox is selected. Click **Finish** to complete the import.
- 3. Customer module integration is discussed in the *OnStream MediaPlayer+ Engine API Reference Manual for Android Platforms*.
- 4. Set the format for the surface using

SurfaceHolder.setFormat(PixelFormat.RGBA 8888);.



# 2 Android Development Environment Setup

This section describes the setup of a general Android application development environment. If you have already set up the development environment and are familiar with the Android application development process, please skip this section and proceed to the next one.

# 2.1 ANDROID SDK/ADT DOWNLOAD AND INSTALLATION

Please use the following web links to download and install the Android SDK:

- http://developer.android.com/sdk/index.html
- http://developer.android.com/sdk/installing.html

The Android SDK is required for the OnStream MediaPlayer+ SDK.

## 2.2 ANDROID NDK DOWNLOAD AND INSTALLATION

Please use the following web link to download and install the Android NDK:

• <a href="http://developer.android.com/tools/sdk/ndk/index.html">http://developer.android.com/tools/sdk/ndk/index.html</a>

The Android NDK is recommended for the OnStream MediaPlayer+ SDK, and required for use with the *SampleEngine* application.

# 2.3 ECLIPSE DOWNLOAD AND INSTALLATION

Please use the following web links to download and install the Eclipse IDE.

- http://www.eclipse.org/downloads
- http://wiki.eclipse.org/Eclipse/Installation

The Eclipse Classic 4.2 IDE is recommended for the OnStream MediaPlayer+ SDK.



# 3 SDK Package Contents

This section describes the contents of the SDK release packages. SDK release packages are contained in a zip compressed file, and include:

- Java (\* .jar) and C/C++ (\*.so) library files
- Documentation (\*.pdf and \*.html) files
- Sample application source code and resource files
- Integration labs

The specific library files, documentation, and sample applications included will vary depending on your SDK configuration.

# 3.1 OnStream MediaPlayer+ SDK Release Structure

After decompression, the OnStream MediaPlayer+ SDK releases, including the Engine, Data Source, and Player packages, have the directory structure shown in Figure 3-1.

OSMP+ <version></version>			
Android			
Bin			
<empty folder=""></empty>			
Doc			
BasePlayer			
<base documentation="" files="" player=""/>			
DataSource			
< Data Source documentation files>			
Engine			
<pre>  &lt; Engine documentation files&gt;</pre>			
Labs			
<integration labs=""></integration>			
<u></u> Jar			
<sdk files="" java="" library=""></sdk>			
Libs			
<pre> <sdk *.so="" c="" c++="" files="" library="" shared=""></sdk></pre>			
SamplePlayer <sample media="" player="" using=""></sample>			
< Eclipse project files such as .project, AndroidManifest.xml, and .classpath>			
<sampleplayer files="" project=""></sampleplayer>			
SampleEngine <sample engine="" media="" player="" using=""></sample>			
< Eclipse project files such as .project, AndroidManifest.xml, and .classpath>			
<sampleengine files="" project=""></sampleengine>			

Figure 3-1: OnStream MediaPlayer+ SDK Directory Structure



# 4 SDK Project Setup

This section describes the SDK project setup for Eclipse/Android platforms.

# 4.1 DECOMPRESS THE SDK PACKAGE

As described in section 3 (SDK Package Contents), SDK packages are contained in a zip compressed file. To install the release:

- 1. Decompress the SDK using an unzip tool.
- 2. Note the path for later use.

# 4.2 CREATE/OPEN AN ANDROID PROJECT

From the Eclipse IDE, open your Android project or create a new one. To create a new project:

- 1. Select **File->New->Project** from the top menu.
- 2. In the **New Project** dialog box, select the **Android->Android Application Project** wizard. Click **Next** to continue.

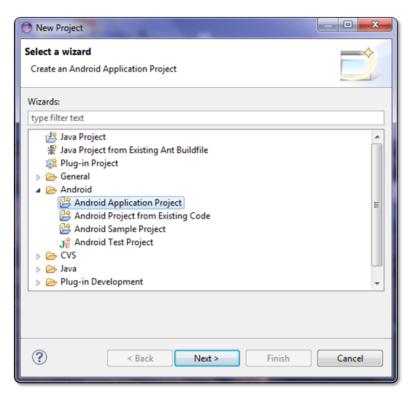


Figure 4-1: Eclipse New Project Dialog Box



- 3. In the **New Android Application** dialog box, set:
  - a. Application Name to Sample.
  - b. Project Name to Sample.
  - c. Package Name to com. visualon. sample.
  - d. Minimum Required SDK to API 8: Android 2.2 (Froyo) or above.

Click **Next** to continue.

**Note:** The values used for the **Application Name**, **Project Name**, and **Package Name** are examples only, and may be modified according to your project specifications.

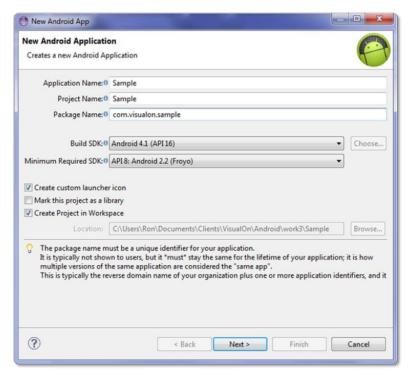


Figure 4-2: Eclipse New Android Application Dialog Box

4. Configure the remaining **New Android Application** wizard settings as desired.

The Sample Android project should now be visible in the Eclipse **Package Explorer**.





Figure 4-3: Eclipse Package Explorer View with Sample Project

### 4.3 Modify the Android Manifest

Usage of the SDK will require internet access and the ability to write to external storage. These capabilities are enabled through the project's manifest. To update the manifest settings:

- 1. Right-click on the *AndroidManifest.xml* file in the **Project Explorer** and select **Open With->Text Editor** from the context-sensitive menu. The manifest file should appear in the **Editor** window.
- 2. Add the following text to the *AndroidManifest.xml* file, before the <uses-sdk...> entry:

```
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
<uses-permission android:name="android.permission.CHANGE_WIFI_STATE"/>
<uses-permission
    android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.INTERNET"/>
```

3. Save the file/project (**Ctrl-S**).



```
AndroidManifest.xml 🛭
                                                                                                                           <manifest xmlns:android="http://schemas.android.com/apk/res/android"</pre>
         package="com.visualon.sample"
         android:versionCode="1'
         android:versionName="1.0"
         <uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
<uses-permission android:name="android.permission.CHANGE_WIFI_STATE"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.INTERNET"/>
              android:minSdkVersion="4"
              android:targetSdkVersion="15" />
         <application
               android:icon="@drawable/ic_launcher"
               android:label="@string/app_name"
               android:theme="@style/AppTheme" >
               <activity
                    android:name=".Player"
                    android:label="@string/title_activity_player" >
                    <intent-filter>
                          <action android:name="android.intent.action.MAIN" />
```

Figure 4-4: Updated AndroidManifest.xml File in Eclipse Editor Window

The Sample project is now ready for use with the SDK.

# 4.4 ADD THE SDK LIBRARIES

To incorporate the SDK into the *Sample* project, the Java library and C/C++ shared library files must be included.

### 4.4.1 Create a libs Folder

If your project does not include a *libs* folder, you need to create it. To create the libs folder in Eclipse:

- 1. Right-click on the *Sample* project root folder in the **Package Explorer** and select **New->Folder** from the context-sensitive menu.
- 2. In the **New Folder** dialog box, set **Folder Name** to libs.
- 3. Click **Finish** to create the folder.

# 4.4.2 Import Java Libraries

To import the SDK Java libraries:

- 1. Right-click on the *libs* folder in the **Package Explorer** and select **Import...** from the context-sensitive menu.
- 2. In the **Import** dialog box, select **General->File System** and click **Next** to continue.



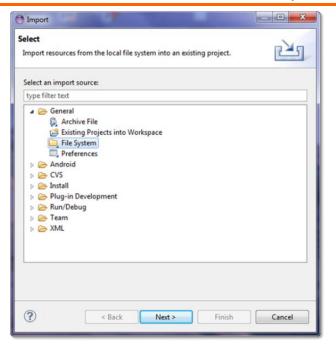


Figure 4-5: Eclipse Import Dialog Box

- 3. In the next dialog box, click **Browse...** and browse to the location of the decompressed SDK package. Browse to the *<SDK\_INSTALL\_DIR>\Android\Jar\*.
- 4. Activate the checkboxes for all of the \*.jar files listed in the directory.
- 5. Expand the Jar (or libs) directory and verify that the debug directory is not selected.

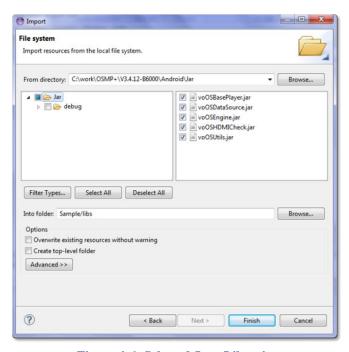


Figure 4-6: Selected Java Libraries

6. Click **Finish** to complete the import.



#### 4.4.3 Create a libs/armeabi Folder

If your project does not include a *libs/armeabi* folder, you need to create it. To create the *libs/armeabi* folder in Eclipse:

- 1. Right-click on the *libs* folder in the **Package Explorer** and select **New->Folder** from the context-sensitive menu.
- 2. In the New Folder dialog box, set Folder Name to armeabi.
- 3. Click **Finish** to create the folder.

# **4.4.4** Import C/C++ Shared Libraries

To import the SDK Java libraries:

- 1. Right-click on the *libs/armeabi* folder in the **Package Explorer** and select **Import...** from the context-sensitive menu.
- 2. In the **Import** dialog box, select **General->File System** and click **Next** to continue.
- 3. In the next dialog box, click **Browse...** and browse to the location of the decompressed SDK package. Browse to the *<SDK\_INSTALL\_DIR>\Android\Libs\* directory.
- 4. Click **Select All** to check all of the library files for import.
- 5. Expand the *Libs* (or *armeabi*) directory and deselect the *debug* directory.

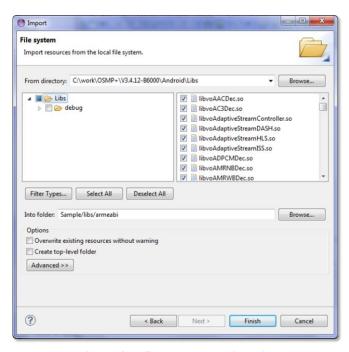


Figure 4-7: Selected Java Libraries

6. Click **Finish** to complete the import.



# **4.4.5** Update Project Properties

With the Java library files now imported into the project, the project build path must be configured to use them. To add the Java library files to the project build path:

- 1. Left-click on the *Sample* project root folder in the Package Explorer and then select **Project->Properties** from the top menu; or right-click on the *Sample* root folder in the **Package Explorer** and select **Properties** from the context-sensitive menu.
- 2. In the **Properties** dialog box, select **Java Build Path** in the left panel, and then select the **Libraries** tab.

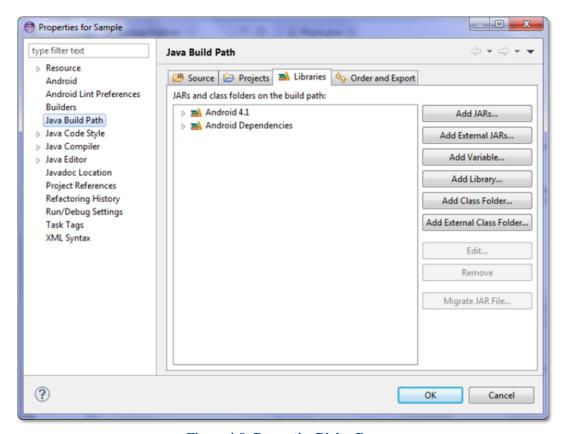


Figure 4-8: Properties Dialog Box

3. Click **Add JARs**. In the **JAR Selection** dialog box, browse to the *libs* directory and select the available \*.*jar* files (use SHIFT/CTRL to select multiple files). Click **OK** to complete the library addition.



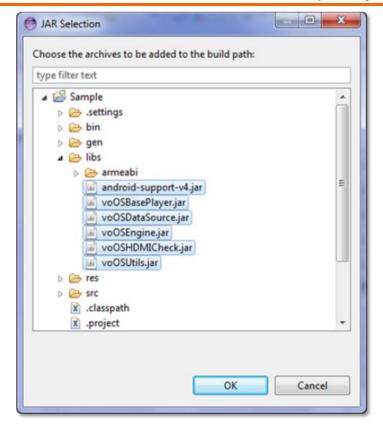


Figure 4-9: JAR Selection Dialog Box

### 4.5 IMPORT THE SDK PACKAGES

The SDK release includes two packages that define the interfaces and classes to be used by an SDK client project. The <code>com.visualon.OSMPPlayer</code> package includes the declarations of the SDK interfaces, classes, and enumerated types. The <code>com.visualon.OSMPPlayerImpl</code> package includes the SDK implementations. For every file in the project that uses the SDK, the appropriate package file should be imported.

To import the entire SDK release:

1. Add the following import directives:

```
import com.visualon.OSMPPlayer.*;
import com.visualon.OSMPPlayerImpl.*;
```

For additional clarity and to reduce the local namespace, individual interfaces and classes can be specified for import. Table 4-1 lists the interfaces and classes included with the OSMPPlayer and OSMPPlayerImpl packages.



Table 4-1: SDK Package Interface and Classes

Package	Interfaces and Classes
VOCommonPlayer	VOCommonPlayer VOCommonPlayerAssetSelection VOCommonPlayerConfiguration VOCommonPlayerControl VOCommonPlayerDeviceInfo VOCommonPlayerHDMI VOCommonPlayerListener VOCommonPlayerSubtitle VOOSMPAnalyticsFilter VOOSMPAnalyticsInfo VOOSMPChunkInfo VOOSMPInitParam VOOSMPTSPStatistics VOOSMPType VOOSMPVerificationInfo
VOCommonPlayerImpl	VOCommonPlayerHDMIImpl VOCommonPlayerImpl VOOSMPAnalyticsInfoImpl VOOSMPAssetIndexImpl VOOSMPAssetPropertyImpl VOOSMPEnumUtils VOOSMPInitParamImpl

# 4.6 ADD ASSETS

Using the SDK will require a local license file (provided by VisualOn), which can be transferred to the mobile device as an asset. A device capability file, which optimizes playback by hardware platform, may also be provided.

**Note:** A temporary evaluation license and sample device capability file are included with the sample player project at *<SDK\_INSTALL\_DIR>\Android\SamplePlayer\assets\*. These examples may be used for test projects and training labs.

#### 4.6.1 Create an assets Folder

If your project does not include an *assets* folder, you need to create it. To create the *assets* folder in Eclipse:

- 1. Right-click on the *Sample* project root folder in the **Package Explorer** and select **New->Folder** from the context-sensitive menu.
- 2. In the **New Folder** dialog box, set **Folder Name** to assets.
- 3. Click **Finish** to create the folder.



# 4.6.2 Import Assets

To import each asset (license or device capability file):

- 1. Using a **File Explorer**, browse to the directory containing the asset.
- 2. Click on the asset file, drag it into the **Package Explorer**, and release it over the *assets* directory.
- 3. In the **File Operation** dialog box, ensure that **Copy files** radio button is selected. Click **OK** to complete the import.

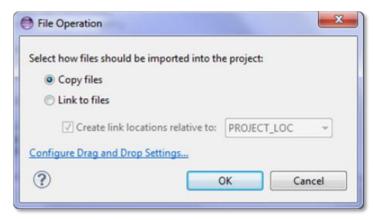


Figure 4-10: File Operation Dialog Box



# 5 Next Steps

You have successfully installed the SDK into your first project, and are ready to proceed with SDK integration, as described in the *OnStream MediaPlayer+ Player SDK Integration Guide for Android Platforms*.