Samsung Electronics

Chord–Unity Plug-in *User Guide*

July 26, 2013

Copyright

Contents

List of Figures

Chord Unity - Introduction and Overview

Chord Unity Plug-in Framework allows the Unity 3D developers to develop proximity based gaming applications that require multi-player gaming support.

This document provides Chord Unity Extension usage instructions and example code for

Assisting developers in using the Chord Unity Extension as easily as possible

This document introduces Chord Unity Extension and demonstrates the API using example code.

This document does not include information about building basic Android & Unity development environment and settings.

Prerequisites

Development machine should have the latest Unity SDK installed (v 4.5.1.5F1 and above)

Chord is supported by Android 4.0(Ice Cream Sandwich), API level 14 or above.

Chord Unity Extension

The following figure shows the Chord Unity Framework:

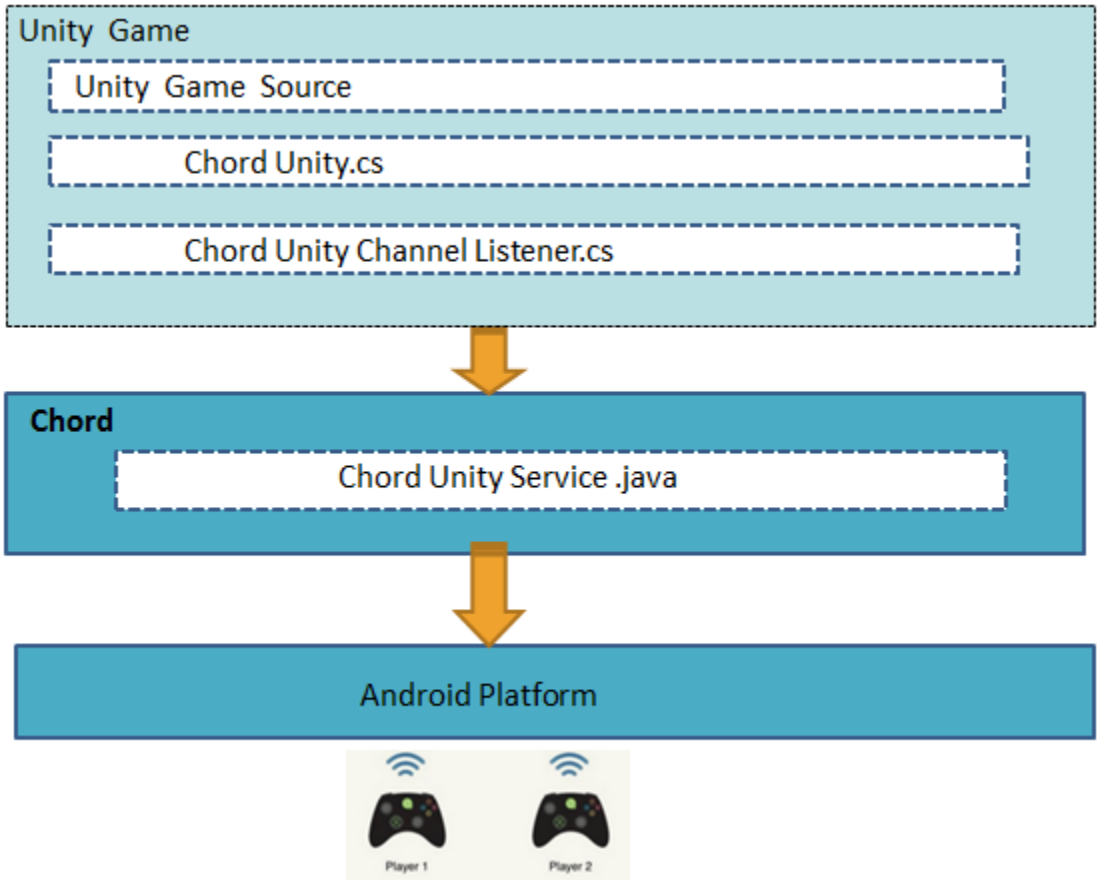


Figure 1: Chord Unity Framework

CHORD

Provides application developers with a simple, high-level set of APIs that can be used to build new exciting distributed applications. Chord intends to make the development environment easier, by providing simple API's. Application developer using Chord can develop network applications, without the knowledge of networking.

Chord Unity Plugin

This module provides Wrapper interfaces for Unity-3D applications to interact with Chord Framework

- Discover other peers in subnet and facilitate multi-player gaming
- Support real time data transfer with in subnet group

Installation of Chord Unity Plugin

1 Chord Unity3D Plugin needs to be deployed in unity 3D IDE project workspace

2 Components of ChordUnity3D plugin are

- ChordUnityInterface.jar
- chord-v2.0.0.jar
- sdk-v1.0.0.jar
- libchord-v2.0.so

3 Open Unity 3D IDE. Import the chord Unity package to unity-3D project by installing chordunity.unitypackage. This will copy the following jar & so Library files and source (C# interface) folder to the following path in the application unity-3d project.

Assets->Plugins->Android folder.

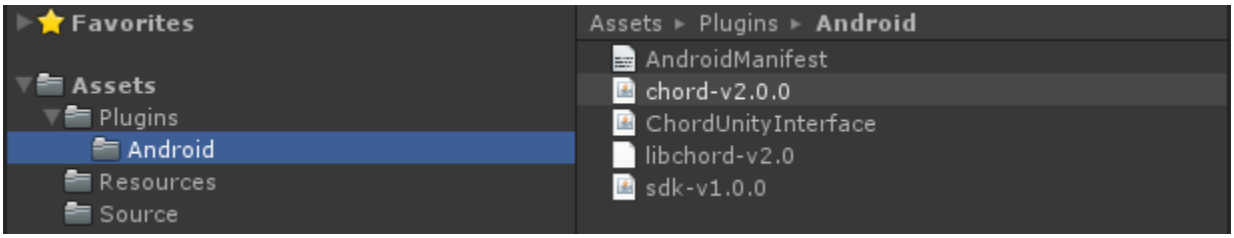
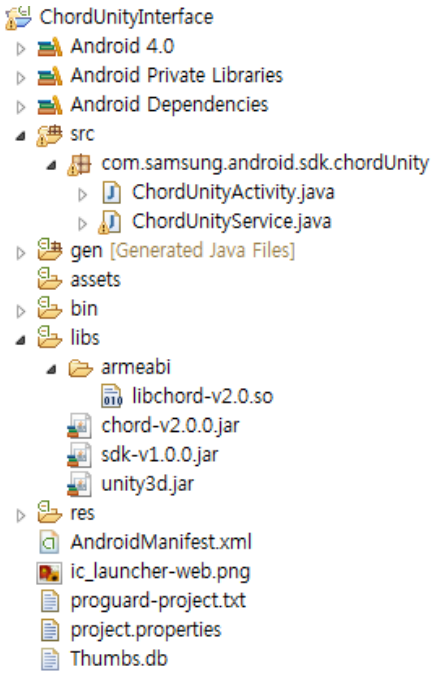


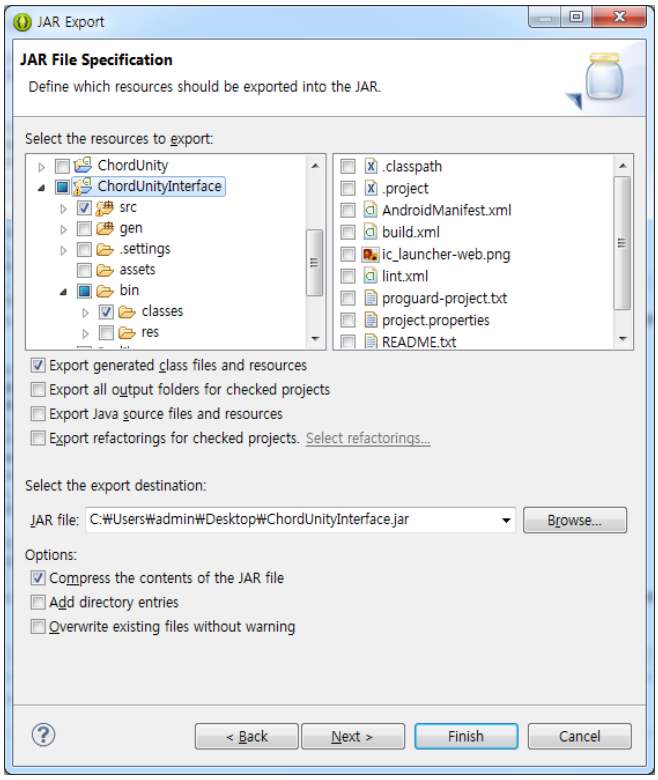
Figure 2: Copy Plugin jars

- How to make ChordUnityInterface jar file.
  - ChordUnityInterface needs to be deployed in project as below.

- chord-v2.0.0.jar
- sdk-v1.0.0.jar
- libchord-v2.0.so
- unity3d.jar(Unity3D library)



- Jar Export.



API Usage Details

3.1 Chord Unity Interface is defined in the following files

1. ChordUnity.CS is c# wrapper for interfacing with Chord
2. ChordUnityChannelListener.cs is c# interface for receiving callback data from Chord for channel related events.
3. ChordUnityNetworkListener.cs is C# interface for receiving call back data from Chord for network related events.
4. ChordUnityStatusListener.cs is C# interface for receiving call back data related to chord status events.
5. ChordUnityConstants.cs has the constants.

Application developers should implement all listener interfaces in the application code and register them for call back events with ChordUnity before starting it.

Chord Unity Listener Interfaces.

1. **ChordUnityStatusListener:** Listens for the connection status of the node by start & stop events.

```
public interface ChordUnityStatusListener
{
    void onStarted(string nodeName, int reason);
    void onStopped(int reason);
}
```

2. **ChordUnityNetworkListener**: Listens for the status of the network for different interfaces, regardless of whether Chord starts or not.

```
public interface ChordUnityNetworkListener
{
    void onConnected(int interfaceType);
    void onDisconnected(int interfaceType);
}
```

3. **ChordUnityChannelListener**: Listens for joining and leaving channels, and data and file transfers, browse path.

```
public interface ChordUnityChannelListener
{
    void onNodeJoined(string fromNode, string fromChannel);

    void onNodeLeft(string fromNode, string fromChannel);

    void onDataReceived(string fromNode, string fromChannel, string payloadType, byte[][] payload);

    void onFileWillReceive(string fromNode, string fromChannel, string fileName, string hash, string fileType, string exchangeId, long fileSize);

    void onFileChunkReceived(string fromNode, string fromChannel, string fileName, string hash, string fileType, string exchangeId, long fileSize, long offset);

    void onFileReceived(string fromNode, string fromChannel, string fileName, string hash, string fileType, string exchangeId, long fileSize, string tmpFilePath);

    void onFileChunkSent(string toNode, string toChannel, string fileName, string hash, string fileType, string exchangeId, long fileSize, long offset, long chunkSize);

    void onFileSent(string toNode, string toChannel, string fileName, string hash, string fileType, string exchangeId);

    void onFileFailed(string node, string channel, string fileName,
```

```
string hash, string exchangeId, int reason);

    void onMultiFilesWillReceive(string fromNode, string fromChannel, string fileName, string taskId, int totalCount, string fileType, long fileSize);


    void onMultiFilesChunkReceived(string fromNode, string fromChannel, string fileName, string taskId, int index, string fileType, long fileSize, long offset);


    void onMultiFilesReceived(string fromNode, string fromChannel, string fileName, string taskId, int index, string fileType, long fileSize, string tmpFilePath);



    void onMultiFilesChunkSent(string toNode, string toChannel, string fileName, string taskId, int index, string fileType, long fileSize, long offset, long chunkSize);


    void onMultiFilesSent(string toNode, string toChannel, string fileName, string taskId, int index, string fileType);


    void onMultiFilesFailed(string node, string channel, string fileName, string taskId, int index, int reason);


    void onMultiFilesFinished(string node, string channel, string taskId, int reason);

    void onBrowseResult(string[] fileList);

}
```

Debug Logging Support:

A wrapper chord class (CLog) for Debug logging, with debug level support is provided in the ChordUnity package.

Setting debug Level to > 0, will enable the Logging, else disable.

```
void setDebugLevel(int userLevel);

int getDebugLevel();

void Log(String msg);

void LogWarning(String msg);

void LogError(String msg);

void LogException(Exception exception);
```

Example code:

```
// ChordUnityStatusListener onStarted Event sample code

public void onStarted(string nodeName, int reason)   {
       CLog.Log("Spin onStarted");
       ChordUnityStarted = true;
       myNodeName = nodeName;
       myIp = mChordWrapper.getIp();
       // Join to public Channel ;
       mChordWrapper.joinChannel(PUBLIC_CHANNEL);
       OnGUI();
    }
```

//API to receive player nodes information joining gaming session

```
public void onNodeJoined (String nodeName, String fromChannel)
{
    CLog.Log("Spin onNodeJoined " + nodeName + ":" + fromChannel);
    if (fromChannel == UNITY_TEST_CHANNEL)
    {
        mPrivateNodeList.Add(nodeName);
    }
    else
        mNodeList.Add(nodeName);
    OnGUI();
}
```

//API to receive player nodes information leaving gaming session

```
public void onNodeLeft(string nodeName, string fromChannel)
{
    CLog.Log("Spin onNodeLeft start" + nodeName + ":" + fromChannel);
    if (fromChannel == UNITY_TEST_CHANNEL)
    {
        mPrivateNodeList.Remove(nodeName);
    }
    else
        mNodeList.Remove(nodeName);
    OnGUI();
}
```

//Sample code to demonstrate onDataRecieved API for the data received from other peer nodes

```
public void onDataReceived(string fromNode, string fromChannel, string
payloadType, byte[][] payload)
{
    CLog.Log("Spin onDataReceived" + fromNode + "chanel:" +
    fromChannel + "payloadType: " + payloadType);
   OnGUI();
}
```

Data Send and Receive Sequence

When a node sends data or a file to another node, there is an acknowledgement to verify that the information was successfully received. For files, the information is broken down into chunks and the successful receipt of each chunk is acknowledged. When the file is complete, the sender receives a message saying so.

1. Node A's app calls sendData()with a message to indicate it would like to send a message to Node B.

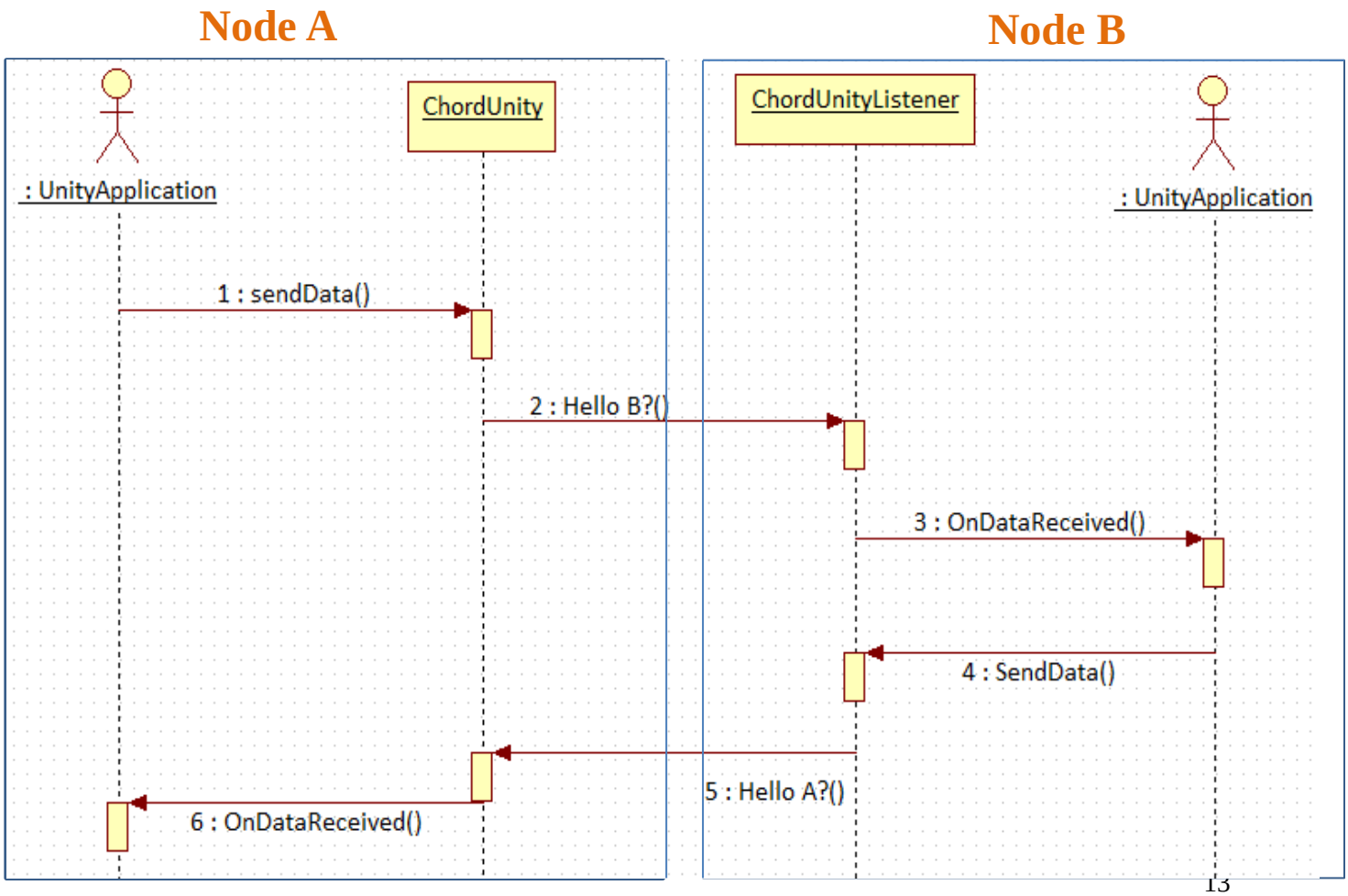
2. Node A's ChordUnity passes the message "Hello B?" to Node B's ChordUnityListener.

3 Node B's ChordUnityListener calls onDataReceived () and the message to Node B's app.

4. Node B's app replies by calling sendData() and message of its own to its ChordUnity.

5. Node B's ChordUnityListener passes the message "Hello A?" to Node A's ChordUnity.

6. Node A's ChordUnityListener passes onDataReceived () to Node A's app.



Assumptions and Limitations

- **Supported APIs:** Chord SDK Unity3D Plugin supports all APIs to Unity applications which are provided in Chord SDK Java APIs.
- **Private Channel Name:** Private channel names must not contain any leading & trailing spaces. In case added by the application, leading & trailing spaces in the channel name will be omitted.