

# i.MX Android™ Wi-Fi Display Sink API User's Guide

## Contents

## 1 Overview

i.MX Android™ release includes support for Wi-Fi Display (WFD) Sink, which allows a device to act as Wi-Fi Display Sink and render the audio and video content from Wi-Fi Source Device. The application framework provides access to the Wi-Fi Display Sink functionality through the i.MX Android Wi-Fi Display Sink APIs. These APIs let applications configure the device as a Wi-Fi Display Sink device, which can be connected with Wi-Fi Display Source, and render the Real-Time Streaming Protocol (RTSP) streaming from Wi-Fi Display Source.

Using the i.MX Android Wi-Fi Display Sink APIs, an Android application can perform the following:

- Scan for other Wi-Fi Display Source devices.
- Get self peer name and MAC address.
- Set the device name that is displayed in other Wi-Fi Display devices.
- Set the display surface to render the RTSP streaming from the Wi-Fi Display Source Device.
- Start or stop the RTSP streaming between Wi-Fi Display Source and Sink.
- Disconnect Wi-Fi display connection from Peer to Peer (P2P) layer.
- Send the input event to the Wi-Fi Display Source device, which known as User Input Back Channel (UIBC).

1	Overview.....	1
2	Requirements.....	2
3	API Description.....	2
4	Examples.....	7



## 2 Requirements

- i.MX Android N7.1.1\_1.0.0 release
- Murata TypeZP Ver2.0 module based on Broadcom BCM4339

## 3 API Description

Below is a description of the i.MX Android WFD Sink API.

### 3.1 Package com.fsl.wfd

#### 3.1.1 com.fsl.wfd.WfdSink

##### Description

This class encapsulates the basic operation of the WFD Sink device.

##### Methods

**Table 1. com.fsl.wfd.WfdSink methods**

Method	Description	Parameter
void startSearch()	This function is used to launch Wi-Fi P2P scanning. After the function is called, Wi-Fi P2P peer discovery starts. When Wi-Fi P2P peers are discovered, it delivers the peers' information.	-
void setDeviceName(String name)	This function is used to set the Wi-Fi P2P device name. This function is called when the device needs to be renamed.	name: the name used to identify a Wi-Fi display device.
String getDeviceName()	This function is used to get the device name. You can display the self name in your UI and the source device gets it from the searched results.	-
String getDeviceMacAddress()	This function is used to get the device MAC address. You can display the MAC address in your UI.	-
void stopSearch()	This function is used to stop the Wi-Fi P2P scanning peers. This function is called after the Wi-Fi P2P connection is disconnected and the WFD Sink function needs to be quit.	-
void stopActive()	This function is used to stop Wi-Fi P2P connection. Call it to disconnect the P2P connection through the lower network layer.	-
void startRtsp(Surface surface)	This function is used to start the RTSP streaming when the Wi-Fi P2P network is connected.	surface: it represents the display area.

*Table continues on the next page...*

**Table 1. com.fsl.wfd.WfdSink methods (continued)**

Method	Description	Parameter
void sendKeyEvent(int KeyCode, KeyEvent event)	This function is used to send the key event through UIBC. If Sink and Source negotiate to support the UIBC, call this interface to send the key event. You can get this event from the key action listener of the activity.	-
void sendTouchEvent(MotionEvent event)	This function is used to send the touch event through the UIBC. If sink and source negotiate to support UIBC, call this interface to send the touch event. You can get this event from the activity view.	-
void setMiracastMode(int mode)	This function is used to set the role of the WifiDisplay. Set WifiP2pManager.MIRACAST_SINK in p2p_search stage to enable the target source device to find the sink device. Set WifiP2pManager.MIRACAST_DISABLED when the connection tears down and WifiDisplaySink Activity stops. This enables the Android Wi-Fi Display Source feature in the i.MX to be restored to the available status.	mode: <ul style="list-style-type: none"> <li>WifiP2pManager.MIRACAST_SINK: This is a standalone connection, which is not disturbed by another P2P device.</li> <li>WifiP2pManager.MIRACAST_DISABLED: The P2P connection is normal.</li> </ul>
void sendScrollEvent(MotionEvent event)	The MotionEvent of the mouse wheel from the onGenericMotionEvent(MotionEvent) can be sent through this API.	-
String getWfdSinkVersion()	This function returns the WfdSink API version.	-
void addRtspStateObserver(RtspStateObserver observer)	This function adds RtspStateObserver to WfdSink. These observers provide eight callbacks with RTSP state change (4 state: start, stop, pause, resume). It is already thread-safely.	observer - Different WifiStateObserver added to WfdSink
void addWifiStateObserver(WifiStateObserver observer)	This function adds WifiStateObserver to WfdSink. These observers provide four callbacks with the WIF state change. It is already thread-safe.	observer - Different WifiStateObserver added to WfdSink
void removeRtspStateObserver(RtspStateObserver observer)	This function removes WifiStateObserver from WfdSink. It is already thread-safe.	observer - RtspStateObserver removed from WfdSink. It should be already added with the WfdSink.
void removeWifiStateObserver(WifiStateObserver observer)	This function removes WifiStateObserver from WfdSink. It is already thread-safe.	observer - WifiStateObserver removed from WfdSink. It should be already added with the WfdSink.
Set<WifiP2pDevice> getPairedDevices()	This function gets the Wi-Fi P2P devices, which are paired with the Sink.	-

### 3.1.2 com.fsl.wfd.SinkView

#### Description

This class extends the SurfaceView. It is used to play the RTSP stream. The SinkView contains a WfdSink object and encapsulates some operations of WfdSink, WifiStateObserver, and RtspStateObserver. It provides a helper class to develop a widget and provide Wi-Fi Display Sink functions.

**Table 2. com.fsl.wfd.SinkView methods**

Method	Description	Parameter
protected void onSinkViewDeviceConnect(boolean isConnect)	Callback of the Wi-Fi P2P state change. It is called if a Wi-Fi P2P is connected or disconnected.	isConnect – Wi-Fi P2P connection state
protected void onSinkViewDeviceListChange(WifiP2pDevice[] devices)	Callback of the Wi-Fi P2P device change. It is called if a new Wi-Fi P2P device is found.	devices - Array of Wi-Fi P2P device found
protected void onSinkViewDeviceUpdate()	Callback of the device information updating.	-
protected void onSinkViewRtspPauseBegin()	Callback of the RTSP pause beginning. Code can be put here before RTSP pauses.	-
protected void onSinkViewRtspPauseFinished()	Callback of the RTSP pause finishing. Code can be put here after RTSP pauses.	-
protected void onSinkViewRtspResumeBegin()	Callback of the RTSP resume beginning. Code can be put here before RTSP resumes.	-
protected void onSinkViewRtspResumeFinished()	Callback of the RTSP resume finishing. Code can be put here after RTSP resumes.	-
protected void onSinkViewRtspStartBegin()	Callback of the RTSP start finishing. Code can be put here before RTSP starts.	-
protected void onSinkViewRtspStartFinished()	Callback of the RTSP start beginning. Code can be put here after RTSP starts.	-
protected void onSinkViewRtspStopBegin()	Callback of the RTSP stop beginning. Code can be put here after RTSP stops.	-
protected void onSinkViewRtspStopFinished()	Callback of the RTSP stop finishing. Code can be put here after RTSP stops.	-
protected void onSinkViewVideoSizeChange(int height, int width)	Callback of the WfdSink video size changing. It is called if the RTSP size changes.	height - video size height width – video size width
public void setOnRtspListener(SinkView. onRtspStateListener listener)	Set the rtspStateListener to get the callback of RTSP start, stop, pause, resume state.	listener - onRtspStateListener in SinkView, provides the callback
public void setWfdSink(WfdSink wfdSink)	Set a wfdSink to SinkView. The wfdSink object should not be null.	wfdSink - use this wfdSink to do some Wi-Fi display operation

### 3.1.3 com.fsl.wfd.SinkActivityBase

#### Description

This class is used to wrap the basic operation of WFD Sink in an android.app.Activity. It contains a default WFD Sink object and some callbacks. It provides a helper class to develop an activity and provide the Wi-Fi Display Sink functions.

## Methods

**Table 3. com.fsl.wfd.SinkActivityBase methods**

Method	Description	Parameter
public WfdSink getWfdSink()	Get the default WFD Sink object created by SinkActivityBase.	-
protected void handleDeviceChangeOnUIThread(WifiP2pDevice[] devices)	Callback of the Wi-Fi P2P device change. This callback runs on the UI thread.	devices - Array of Wi-Fi P2P device found
protected void handleDeviceUpdateOnUIThread()	Callback of the current device change. This callback runs on the UI thread.	-
protected void handleRtspPauseBeginOnUIThread()	Callback of the RTSP pause beginning. Code can be put here before RTSP pauses. This callback runs on the UI thread.	-
protected void handleRtspPauseFinishedOnUIThread()	Callback of the RTSP pause finishing. Code can be put here after RTSP pauses. This callback runs on the UI thread.	-
protected void handleRtspResumeBeginOnUIThread()	Callback of the RTSP resume beginning. Code can be put here before RTSP resumes. This callback runs on the UI thread.	-
protected void handleRtspResumeFinishedOnUIThread()	Callback of the Rtsp resume finishing. Code can be put here after Rtsp resumes. This callback runs on the UI thread.	-
protected void handleRtspStartFinishedOnUIThread()	Callback of the RTSP start finishing. Code can be put here after RTSP starts. This callback runs on the UI thread.	-
protected void handleRtspStoptBeginOnUIThread()	Callback of the RTSP stop beginning. Code can be put here before RTSP stops. This callback runs on the UI thread.	-
protected void handleRtspStoptFinishedOnUIThread()	Callback of the RTSP stop finishing. Code can be put here after RTSP stops. This callback runs on the UI thread.	-
protected void handleSinkConnectOnUIThread(boolean connected)	Callback of the WifiP2p connect state. This callback runs on the UI thread.	isConnect – Wi-Fi P2P connect state. • true: connect • false: disconnect
protected void handleSinkVideoSizeChangeOnUIThread(int height, int width)	Callback of the WfdSink video size change. This callback runs on the UI thread.	height - video size height width - video size width
protected void handleRtspStartBeginOnUIThread()	Callback of the RTSP start beginning. Code can be put here before RTSP starts. This callback runs on the UI thread.	-
protected abstract void initView()	initializes the root view and this method should be complete in sub class.	-
protected void onExit()	Callback of exit.	-
protected void onStartSearch()	Callback of searching for Wi-Fi P2P devices. Application can update the UI about searching status in this Callback.	-

*Table continues on the next page...*

**Table 3. com.fsl.wfd.SinkActivityBase methods (continued)**

Method	Description	Parameter
protected abstract void onWifiDisabled()	Callback of Wi-Fi disabled. Something should be done to open the Wi-Fi before Wi-Fi P2P sink.	-
protected abstract void onWifiEnabled()	Callback of Wi-Fi enabled. startSearch for Wi-Fi P2P devices can be done here.	-
protected void startSearch()	Start searching for Wi-Fi P2P devices. This API needs to be called actively.	-
protected void stopSearch()	Stop searching for Wi-Fi P2P devices. This API needs to be called actively.	-

## 3.2 Package com.fsl.wfd.observer

### 3.2.1 com.fsl.wfd.observer.WfdObserver

#### Description

This class is the base of the Wi-Fi state observer and the RTSP state observer.

#### Methods

**Table 4. com.fsl.wfd.observer.WfdObserver methods**

Method	Description	Parameter
WfdObserver(java.lang.String observerName)	Constructor	observerName - observer name
String getName()	Gets the observer name.	-

### 3.2.2 com.fsl.wfd.observer.RtspStateObserver

#### Description

This class provides the callbacks for the RTSP state change. WfdSink provides the interface addRtspStateObserver() to register this callback and removeRtspStateObserver() to unregister this callback.

#### Methods

**Table 5. com.fsl.wfd.observer.RtspStateObserver methods**

Method	Description	Parameter
RtspStateObserver(java.lang.String observerName)	Constructor	observerName - observer name
abstract void onRtspStartBegin()	Callback of the RTSP start beginning. Code can be put here before RTSP starts.	-

*Table continues on the next page...*

**Table 5. com.fsl.wfd.observer.RtspStateObserver methods (continued)**

Method	Description	Parameter
abstract void onRtspStartFinished()	Callback of the RTSP start finishing. Code can be put here after RTSP starts.	-
abstract void onRtspStopBegin()	Callback of the RTSP stop beginning.	-
abstract void onRtspStopFinished()	Callback of the RTSP stop finishing. Code can be put here after RTSP stops.	-
abstract void onRtspPauseBegin()	Callback of the RTSP pause beginning. Code can be put here before RTSP pauses.	-
abstract void onRtspPauseFinished()	Callback of the Rtsp pause finishing. Code can be put here after RTSP pauses.	-
abstract void onRtspResumeBegin()	Callback of the RTSP resume beginning. Code can be put here before RTSP resumes.	-
abstract void onRtspResumeFinished()	Callback of the RTSP resume finishing. Code can be put here after RTSP resumes.	-

### 3.2.3 com.fsl.wfd.observer.WifiStateObserver

#### Description

This class provides 4 callbacks for Wi-Fi state change. WFD Sink provides the interface – addWifiStateObserver() to register this callback and removeWifiStateObserver() to unregister this callback.

#### Methods

**Table 6. com.fsl.wfd.observer.WifiStateObserver methods**

Method	Description	Parameter
WifiStateObserver(java.lang.String observerName)	Constructor	observerName - observer name
abstract void onDeviceListChange(WifiP2pDevice[] devices)	Callback of Wi-Fi P2P device Change. After starting searching the Wi-Fi P2P device, call the callback when the Wi-Fi P2P device is found.	devices - Array of Wi-Fi P2P device found
abstract void onDeviceConnect(boolean isConnected)	Callback of Wi-Fi P2P connect State. This callback is called when the Wi-Fi P2P device is connected.	isConnect - WIFI P2P connect state, true is connect, false is disconnect.
abstract void onDeviceUpdate()	Callback of current device change.	-
abstract void onSinkVideoSizeChange(int height, int weight)	Callback of WfdSink video size change.	height - video size height weight – video size width

## 4 Examples

A demo application is provided in myandroid/device/wfd-proprietary/WfdSinkApp to show how to use the WFD Sink APIs.

## Examples

When writing Android.mk, add these two variables:

- Include fsl.imx as static Java library.
- Disable the ProGuard function.

```
LOCAL_STATIC_JAVA_LIBRARIES := fsl.imx
LOCAL_PROGUARD_ENABLED := disabled
```

## 4.1 Registering the callbacks

This example shows how to register the callbacks.

```
mWfdSink = new WfdSink(this);
mWifiStateObserver = new SinkActivityWifiStateObserver(TAG);
mRtspStateObserver = new SinkActivityRtspStateObserver(TAG);
mPWfdSink.addWifiStateObserver(mWifiStateObserver);
mPWfdSink.addRtspStateObserver(mRtspStateObserver);

.....
//Two observers should be implemented.
class SinkActivityWifiStateObserver extends WifiStateObserver {
    public SinkActivityWifiStateObserver(String observerName) {
        super(observerName);
    }

    @Override
    public void onDeviceListChange(final WifiP2pDevice[] devices) {
        .....
    }

    @Override
    public void onDeviceConnect(final boolean isConnect){
        .....
    }

    @Override
    public void onSinkVideoSizeChange(final int height, final int width){
        .....
    }

    @Override
    public void onDeviceUpdate(){
        .....
    }
}
class SinkActivityRtspStateObserver extends RtspStateObserver {
    public SinkActivityRtspStateObserver(String observerName) {
        super(observerName);
    }

    @Override
    public void onRtspStartBegin(){
    }
}
```

## 4.2 Enabling discoverability

Call the API startSearch() to start scanning Wi-Fi display source devices.



## 4.3 Connecting devices

The Wi-Fi P2P network broadcast is received. The Wi-Fi P2P source device list can be obtained when WFD\_DEVICE\_LIST\_CHANGED\_ACTION is received. Call the callback onDeviceConnect() of WifiStateObserver. Then, call the startRtsp() and stopRtsp() methods of WFD Sink.

To ensure that the P2P Stack is exclusively used by the Wi-Fi Display Sink Application, use setMiracastMode to set the role of WifiDisplay.

To manage the Wi-Fi P2P network connection state, execute the following commands:

```
private handleConnection(boolean connected) {
    if (connected) {
        mWfdSink.startRtsp(mSurfaceHolder.getSurface());
        mWfdSink.setMiracastMode(WifiP2pManager.MIRACAST_SINK);
    }
    else {
        mWfdSink.stopActive();
        mWfdSink.setMiracastMode(WifiP2pManager.MIRACAST_DISABLED);
    }
}
```

## 4.4 Stopping Wi-Fi Display Sink

The following is the example to stop Wi-Fi Display Sink:

```
mWfdSink.setMiracastMode(WifiP2pManager.MIRACAST_DISABLED);
mWfdSink.stopActive();
```

### NOTE

The default Android WFD Source requires the RTSP streaming to start within 15 seconds after the Wi-Fi P2P connection is built.

## 4.5 How to use sendKeyEvent, sendScrollEvent, and sendTouchEvent

The APIs sendKeyEvent, sendScrollEvent, and sendTouchEvent, have one event parameter. These events can be captured by Android input event-related APIs. The following are the examples to capture the Key and Motion event and send them to Wi-Fi Display Source through the Wi-Fi Display Sink APIs:

```
@Override
public boolean onKeyDown(int keyCode, KeyEvent event) {
    if (keyCode == KeyEvent.KEYCODE_BACK) {
        mHandler.removeMessages(SET_UI_FLAG_HIDE_NAVIGATION);
        this.exitDialog();
    } else if (keyCode == KeyEvent.KEYCODE_VOLUME_DOWN
        || keyCode == KeyEvent.KEYCODE_VOLUME_UP) {
        mWfdSink.sendKeyEvent(keyCode, event);
        return true;
    }
    return super.onKeyDown(keyCode, event);
}

mSurfaceView = (SinkView)findViewById(R.id.sink_preview);
SurfaceHolder holder = mSurfaceView.getHolder();
holder.addCallback(this);
```

## Examples

```
holder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);
mSurfaceView.setOnTouchListener(new OnTouchListener() {
    @Override
    public boolean onTouch(View v, MotionEvent event) {
        Log.d(TAG, "sink view on touch event x=" + event.getX() + "y=" +
event.getY());
        mWfdSink.sendTouchEvent(event);
        return true;
    }
});
@Override
public boolean onGenericMotionEvent(MotionEvent event) {
    if (event.isFromSource(InputDevice.SOURCE_MOUSE)) {
        switch (event.getAction()) {
            case MotionEvent.ACTION_SCROLL:
                mWfdSink.sendScrollEvent(event);
                break;
            case MotionEvent.ACTION_HOVER_MOVE:
                mWfdSink.sendTouchEvent(event);
                Log.i(TAG, "sendrolon GenerMotionEvent HoverMoved." +
event.toString());
                break;
        }
    }
    return super.onGenericMotionEvent(event);
}
```

## 4.6 How to use SinkView to display

SinkView is based on SurfaceView. It is used to play the RTSP stream in the surface automatically. It simplifies the developing work by wrapping the related operation of Wi-Fi Display Sink. Developers still can write a view based on SurfaceView to implement the Wi-Fi Display Sink. The following examples show how to use it.

To initialize SinkView:

```
SinkView sinkView = new SinkView(context);
```

or

```
SinkView sinkView = (SinkView) findViewById(R.id.sinkView);
```

To set WFD Sink to SinkView.

```
sinkView.setWfdSink(wfdSink);
```

To add an RTSP state listener:

```
sinkView.setOnRtspListener(new SinkView.onRtspStateListener{...});
```

There are two ways to get the RTSP state callback of SinkView:

- Extend the SinkView and then override the onSinkViewRtspStopBegin() and other seven callbacks.
- Set a listener as an onclick listener. This way is more convenient than the previous one.

## 4.7 How to use SinkActivityBase

SinkActivity provides a basic activity that wraps two WFD Observers. It helps develop an activity, which implements the Wi-Fi Display Sink. The following example shows how to use it.

```
public class MyWfdSinkActivity extends SinkActivityBase{
    @Override
```

```

protected void initView() {
    //init the UI widgets here.
    //getWfdSink or new WfdSink and register the wfdSink to the sinkView
    //Ensure that the WfdSink in Activity and SinkView is same.
    mWfdSink = getWfdSink();
    mSurfaceView = (SinkView)findViewById(R.id.sink_preview);
    mSurfaceView.setWfdSink(mWfdSink);
}

//Wifi Sink need wifi open, if the wifi opened, the activity will call this method
//else onWifiDisabled() will be called.
@Override
protected void onWifiEnabled() {
    if(!isExit)
        startSearch();
}

//Stop the WIFI P2P device search and handle wifi p2p connection
@Override
protected void handleSinkConnectOnUiThread(boolean connected) {
    super.handleSinkConnectOnUiThread(connected);
    stopSearch();
    handleConnected(connected)
}

//handle RTSP state
@Override
protected void handlRtspStartBeginOnUiThread() {
    super.handlRtspStartBeginOnUiThread();
}

//handle WIFI P2P state
@Override
protected void handleSinkConnectOnUiThread(boolean connected){
    super.handleSinkConnectOnUiThread();
}
}

```

## 4.8 Revision History

**Table 7. Revision history**

Revision number	Date	Substantive changes
0	03/2017	Initial release

---

**How to Reach Us:**

**Home Page:**  
[nxp.com](http://nxp.com)

**Web Support:**  
[nxp.com/support](http://nxp.com/support)

Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. Typical parameters that may be provided in NXP data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including typicals, must be validated for each customer application by customer's technical experts. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address:  
[nxp.com/SalesTermsandConditions](http://nxp.com/SalesTermsandConditions).

NXP, the NXP logo, Freescale, and the Freescale logo are trademarks of NXP B.V. All other product or service names are the property of their respective owners. All rights reserved.

© 2017 NXP B.V.

Document Number: WFDSINKAPIUG  
Rev. 0  
03/2017

