Creating a Microphone Application

Published 2014-10-27 | (Compatible with SDK 3.5,4.5,5.0,5.1 and 2012,2013,2014 models)

> This tutorial describes the use of microphone class of the External Interworking API. This class is needed to create various kinds of TV applications, like karaoke using different types of microphones. The application being developed in this tutorial demonstrates the development of TV application using microphone. The application connects to a USB microphone device and gets data from the device through TV platform.

Contents

Programming of applications using microphones is mostly based on the External Interworking API of the Hardware API.

Prerequisites

To create applications that run on a TV screen, you need:

Samsung TV connected to the Internet

SDK **(Samsung Smart TV SDK is recommended)** or a text editor for creating HTML, JavaScript and CSS files

Environment

Use Samsung Smart TV SDK to create the application. Emulator provided along with SDK could be used to debug and test the application before deploying it onto TV. Refer Testing Your Application on a TV. Note that applications may perform better on the TV than on the emulator.

You can find general instructions for creating applications in Implementing Your Application Code.

Source Files

> Note
>
> The files needed for the sample application are **here**.

The tutorial Microphone Application is located under Tutorial_Microphone. Source files in directory are explained in the table:

Directory	Description
css	Contains the StyleSheet file TutorialMicrophone.css
resource	Contains the image folder which contains background.jpg

Directory	Description
js	Contains the JavaScript file TutorialMicrophone.js which does the following: initializes codes registers events key handling displays the results

VirtualBox (SDK 4.5)

USB Mic's pcm data are not delivered correctly on the VirtualBox. If you want to hear the voice, you should record it as mic.pcm before run the SDK. And copy the file to Apps directory.

> Note
>
> E.g.: arecord –f S16_LE –r 48000 –c 1 > mic.pcm (linux record example)

Microphone Application

Starting the Microphone Application

To start the microphone application,

1. Select the **Open App** on the right-bottom menu (refer to the figure below).

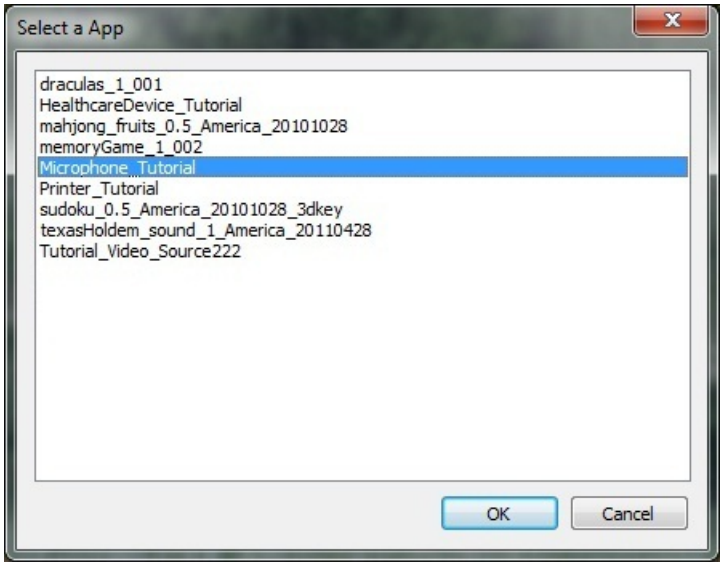

Figure: Selecting the application

2. Select Microphone_Tutorial. If the following display appears on the emulator, then it means that application has started successfully:

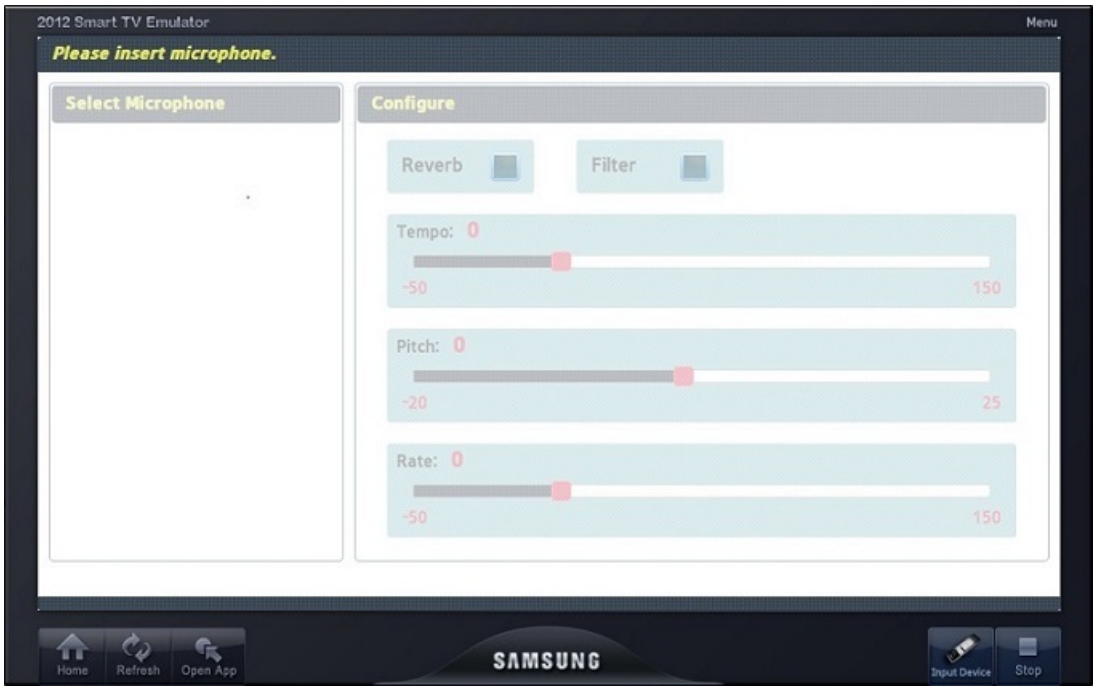

Figure: Microphone application started

3. Connect an usb microphone to TV. You can see the connected microphone list on the left side.

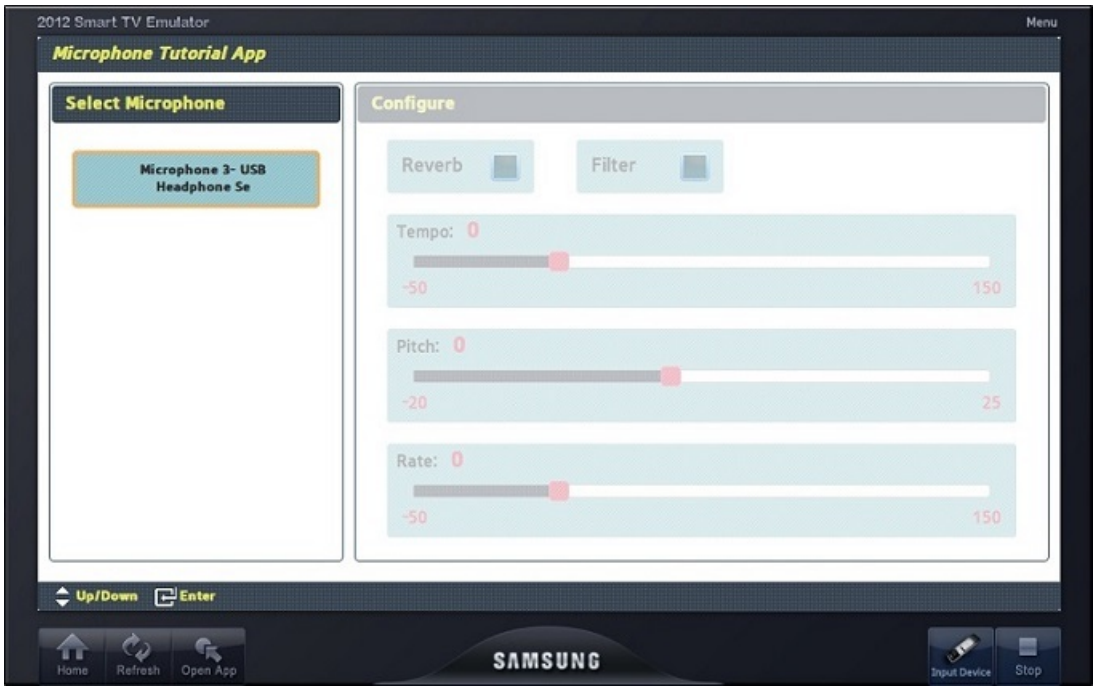

Figure: Figure: connected Microphone

Using the Microphone

Following sections explain how to use the connected microphone.

1. If you want to enable and play the connected microphone, just press **Enter** key of the remote control. After pressing enter key, you can see the configure screen enabled as below figure.

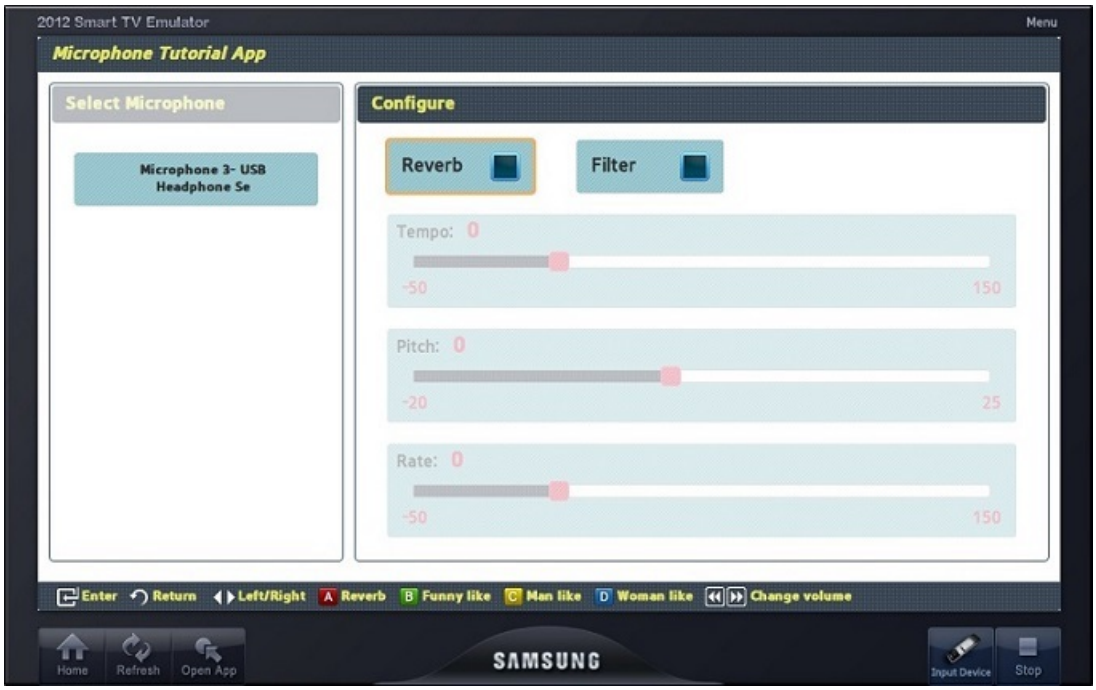

Figure: Control the volume of microphone

Please just press **Return** key of remote control, if you want to stop and disable the connected microphone.

2. If you want to increase the mic volume, press **|>>** key of the remote control. Current volume level can be seen as in below figure.

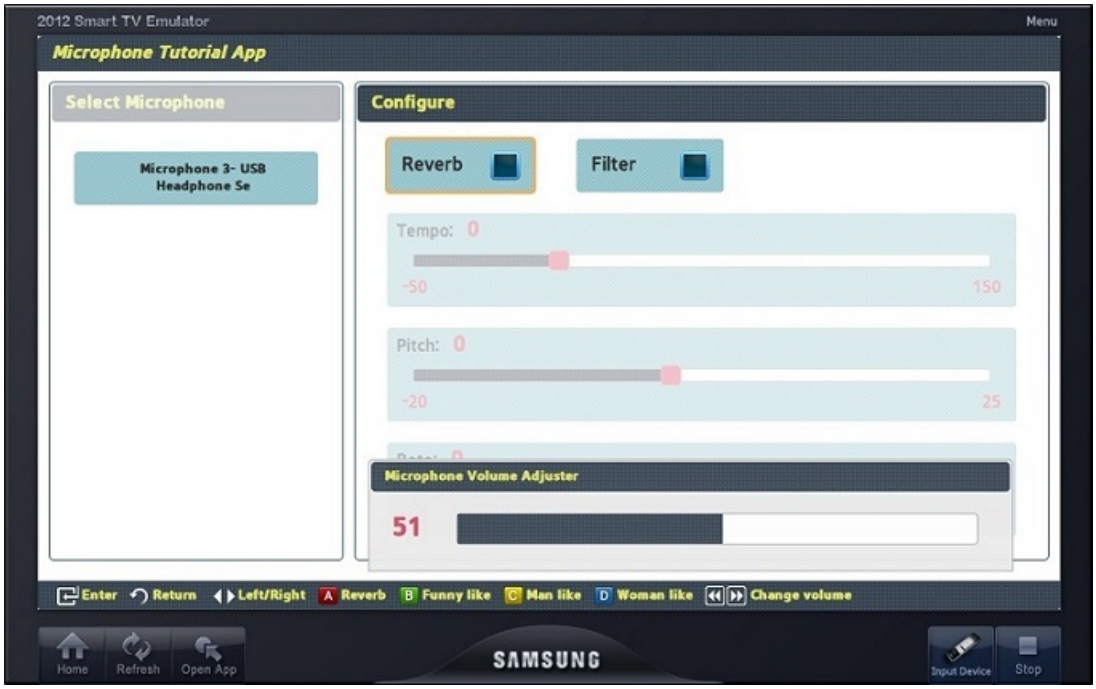

Figure: Control the volume of microphone

If you want to decrease the mic volume, press **<<|** key of the remote control.

3. The reverb effect of the voice can be applied by pressing **A(RED)** key of the remote control or **Enter** key of remote control on the Reveb menu as in the below figure.

> Note
>
> Currently, SDK doesn't support real reverb effect, but just displays the message as below.

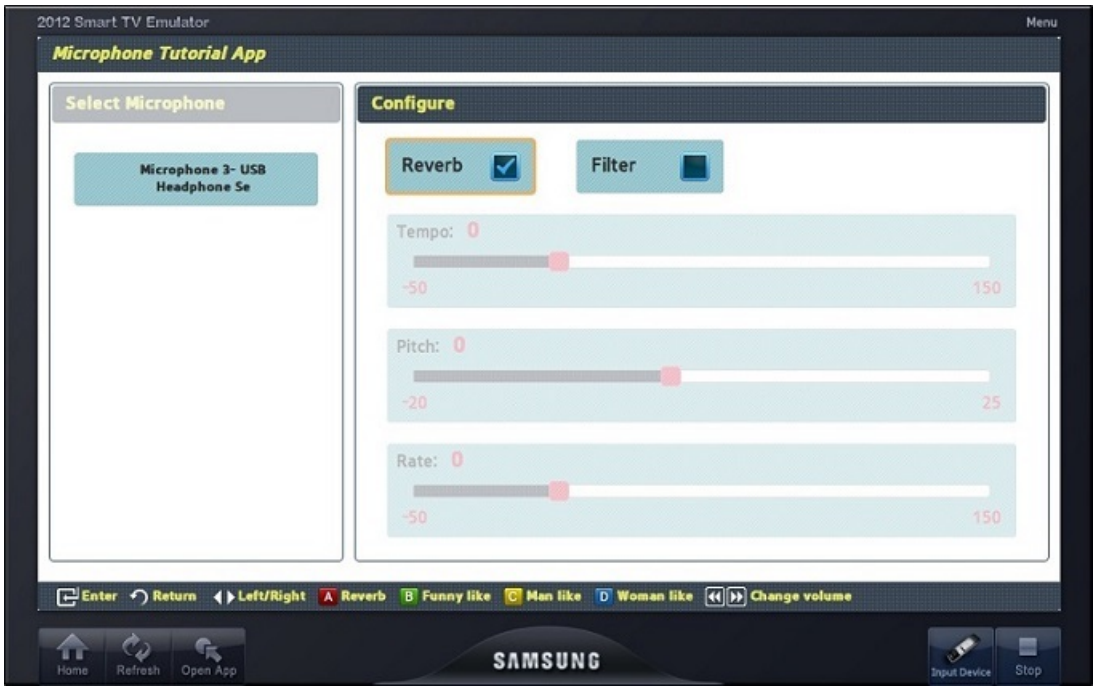

Figure: Control the reverb effect

Cancel the reverb effect by pressing **A(RED)** key or **Enter** key of the remote control again. This result in display change as below follow. (only message printed with SDK)

4. You can set the filter effect by selecting **Filter** menu. Below figure shows the display change on screen:

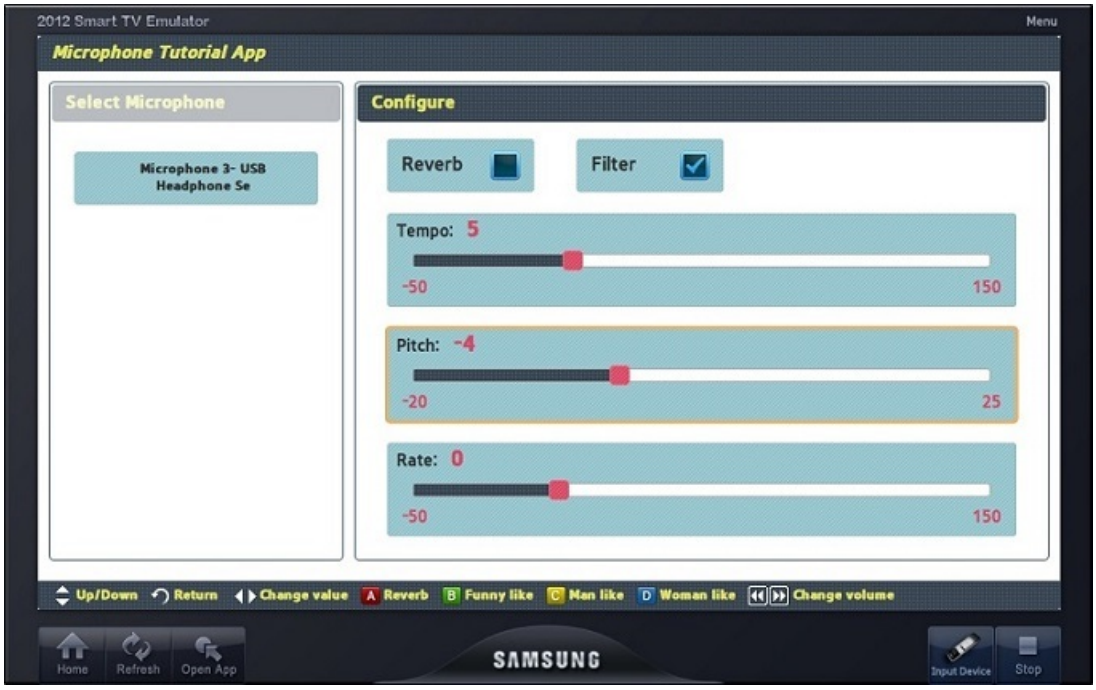

Figure: Turn on the Filter Effect

> Note
>
> This feature is only enabled on the supported models. Please check it with getSupportedEffects function before using it.

You can set the filter value by pressing **left / right key** key of the remote control on each items..

Getting an Available Microphone

If microphones are connected, window.webapis.microphone.getMicrophones() function will be return each Microphone class object of connected microphones.

Function	Description
getMicrophones	Get Microphones which are connected to TV. When it called for the first time, it will initialize all the internal modules. Even though several microphones instances are returned, only one microphone can be used. So, get only one microphone instance from this function

```
var microphone = window.webapis.microphone || {};

Main.keyDown = function () {
    var keyCode = event.keyCode;

    switch (keyCode) {
        case gTVKey.KEY_ENTER:
            microphone.getMicrophones(Main.onMicrophoneObtained);
            break;
            ...
    }
}

Main.onMicrophoneObtained = function (mics) {
    var enabledEffect;

    if (mics.length > 0) {
        if (mics[0] != null) {
            micDevice = mics[0];
            if (micDevice.enableDevice(microphone.MICROPHONE_FORMAT_SIGNED_16BIT_LITTLE_ENDIAN,
                                microphone.MICROPHONE_FRAMERATE_48000) == false) {
                return;
            }

            if (micDevice.play() == true) {
                document.getElementById("enable").innerHTML = "Your microphone ON.";
            }
            else {
                document.getElementById("enable").innerHTML = "Can't be ON, Error happened!";
                return;
            }

    ...
        }
    } else {
        ...
    }
}
```

Registering a Callback Function for a Device Connection Event

To receive a device connection event, register a callback function with window.webapis.microphone.registerManagerCallback() function. Once the callback registration is done, callback functions are called at connecting or disconnecting a microphone. The callback function can get window.webapis.microphone.ManagerEvent class object including event type and microphone name as an input parameter.

Function	Description
registerManagerCallback	Register callback function to get the event about CONNECT/DISCONNECT

```
Main.onLoad = function () {
    var nDevNum = 0;

    gWidgetAPI = new Common.API.Widget();
    // Create Common module
    gTVKey = new Common.API.TVKeyValue();
    gWidgetAPI.sendReadyEvent();
    // Send ready message to Application Manager

    microphone.registerManagerCallback(Main.onDeviceStatusChange);
    ...
}

Main.onDeviceStatusChange = function (sParam) {
    switch (sParam.eventType) {
        case microphone.MGR_EVENT_DEV_DISCONNECT:
            if (micDevice != null) {
                if (micDevice.getUniqueID() == sParam.deviceUID) {
                    alert("Your microphone is disconnected right now. ");
                    micDevice = null;
                }
            }
            ...
            break;
        default:
            break;
    }

    return;
}
```

Enable or Disable the Microphone

Microphone can be enabled to use, by calling Microphone.enableDevice (unsigned short format, unsigned short framerate) function. Once the call is made, the microphone is ready to use. So, this function should be called before calling Microphone.play() function. Call Microphone.disableDevice(), if microphone is no longer needed and need to be disabled.

Function	Description
enableDevice	Make the microphone device enable.
disableDevice	Make the microphone device disable.

```
Main.onMicrophoneObtained = function (mics) {
    var enabledEffect;

    if (mics.length > 0) {
        if (mics[0] != null) {
            micDevice = mics[0];
            if (micDevice.enableDevice(microphone.MICROPHONE_FORMAT_SIGNED_16BIT_LITTLE_ENDIAN,
                                microphone.MICROPHONE_FRAMERATE_48000) == false) {
                return;
            }

            if (micDevice.play() == true) {
                document.getElementById("enable").innerHTML = "Your microphone ON.";
            } else {
                return;
            }

            micVolume = micDevice.getVolumeLevel();

            alert("Currently,mic volume is " + micVolume + ".");

            ...
        }
    }
    ...
}
```

Play or Stop the Microphone

To start hearing voice from speaker, call the Microphone.play() function. Call Microphone.stop() to stop hearing voice from speakers.

Function	Description
play	Start to send the voice from the microphone to the TV speaker
stop	Stop sending the voice from the microphone to the TV speaker.

```
Main.keyDown = function () {
    var keyCode = event.keyCode;

    switch (keyCode) {
        case gTVKey.KEY_RED: {
            if (micDevice != null) {
                if (micDevice.play() == true) {
                    document.getElementById("enable").innerHTML = "Your microphone ON.";
                } else {
                    document.getElementById("enable").innerHTML = "Can't be ON, Error happened!";
                }
            }
            break;
        }
        case gTVKey.KEY_BLUE: {
            if (micDevice != null) {
                if (micDevice.stop() == true) {
                    document.getElementById("enable").innerHTML = "Your microphone OFF.";
                } else {
                    document.getElementById("enable").innerHTML = "Can't be OFF, Error happened!";
                }
            }
            break;
        }

    }
}
```

Set or get volume level

If the microphone supports the volume control, you can set or get the volume of the microphone.

Function	Description
setVolumeLevel	Sets the current volume level of microphone
getVolumeLevel	Gets the current volume level of microphone.

```
Main.keyDown = function () {
    var keyCode = event.keyCode;

    switch (keyCode) {
        case gTVKey.KEY_LEFT: {
            if (micDevice != null) {
                micVolume--;
                if (micVolume < 0) {
                    micVolume = 0;
                }
                micDevice.setVolumeLevel(micVolume);
                alert("__volume DOWN . level = " + micVolume + "___");
                document.getElementById("volume").innerHTML = "volume down: " + micVolume;
            }
            break;
        }
        case gTVKey.KEY_RIGHT: {
            if (micDevice != null) {
                micVolume++;
```

```
            if (micVolume > 100) {
                micVolume = 100;
            }
            micDevice.setVolumeLevel(micVolume);
        }
        break;
    }
  }
}

Main.onMicrophoneObtained = function (mics) {
  var enabledEffect;

  if (mics.length > 0) {
    if (mics[0] != null) {
      micDevice = mics[0];
      if (micDevice.enableDevice(microphone.MICROPHONE_FORMAT_SIGNED_16BIT_LITTLE_ENDIAN,
                    microphone.MICROPHONE_FRAMERATE_48000) == false) {
        return;
      }

      if (micDevice.play() == true) {
        document.getElementById("enable").innerHTML = "Your microphone ON.";
      } else {
        return;
      }

      micVolume = micDevice.getVolumeLevel();

      alert("Currently,mic volume is " + micVolume + ".");

      ...
    }
  }
  ...
}
```

Set or get effects

A few models of TVs support several effects. To use those effects, effects functions such as getSupportedEffects, getEnabledEffects, setEffect are provided.

Function	Description
setEffect	Set effect ON/OFF. By default, all effects are Off. Currently, Reverb and Filter effect is supported by TV.
getSupportedEffects	Get the all kind of effect type that TV can support. If you want to use some effect, you are supposed to use setEffect function. Supported effects should be queried by calling getSupportedEffects() before calling setEffect(). For example, some TV model may support reverb effect, while some other TV model may not support reverb effect.
getEnabledEffects	Get the all effects that is now enabled.

```
Main.keyDown = function () {
    var keyCode = event.keyCode;

    switch (keyCode) {
        ...
        case gTVKey.KEY_GREEN:
            if (micDevice != null) {
                if (supportedEffects & microphone.MICROPHONE_EFFECT_REVERB) {
                    micDevice.setEffect(microphone.MICROPHONE_EFFECT_REVERB, true);
                    document.getElementById("reverb").innerHTML = "Reverb effect ON.";
                }
            }
            break;
        case gTVKey.KEY_YELLOW:
            if (micDevice != null) {
                if (supportedEffects & microphone.MICROPHONE_EFFECT_REVERB) {
                    micDevice.setEffect(microphone.MICROPHONE_EFFECT_REVERB, false);
                    document.getElementById("reverb").innerHTML = "Reverb effect OFF.";
                }
            }
            break;
    }
    ...
}

Main.onMicrophoneObtained = function (mics) {
    var enabledEffect;

    if (mics.length > 0) {
        if (mics[0] != null) {
            micDevice = mics[0];
            if (micDevice.enableDevice(microphone.MICROPHONE_FORMAT_SIGNED_16BIT_LITTLE_ENDIAN, \
                                       microphone.MICROPHONE_FRAMERATE_48000) == false) {
                return;
            }

            if (micDevice.play() == true) {
                document.getElementById("enable").innerHTML = "Your microphone ON.";
            } else {
                document.getElementById("enable").innerHTML = "Can't be ON, Error happened!";
                return;
            }

            supportedEffects = micDevice.getSupportedEffects();

            enabledEffect = micDevice.getEnabledEffects();
            if (enabledEffect != 0) {
                alert("Currently, sound effects[" + enabledEffect + "] are on. ");
            }
        }
    }
    ...
}
```