Development Guidance for Multi-Application

Published 2014-10-28 | (Compatible with SDK 5.0,5.1 and 2014 models)

> Development Guidance for Multi-Application

Contents

About document

This document explain about basic structure of Samsung SMART TV Web App RunTime and environment of operating. For developing Multi-tasking Application, refer to this guidance document. If App is supporting Multi-Tasking Application, App performance is expected, because App re-launch speed will be improved. You should consider caution of this document and apply source code. From now on, Multi-tasking Application be referred to as Multi-app.

Structure of Web Application RunTime

Basic Structure

Web App RunTime is driven onto the Samsung SMART TV platform which designed based on Linux. And **Web App RunTime** has Web Engine with Webkit2. It provides a variety of functions like Webkit.

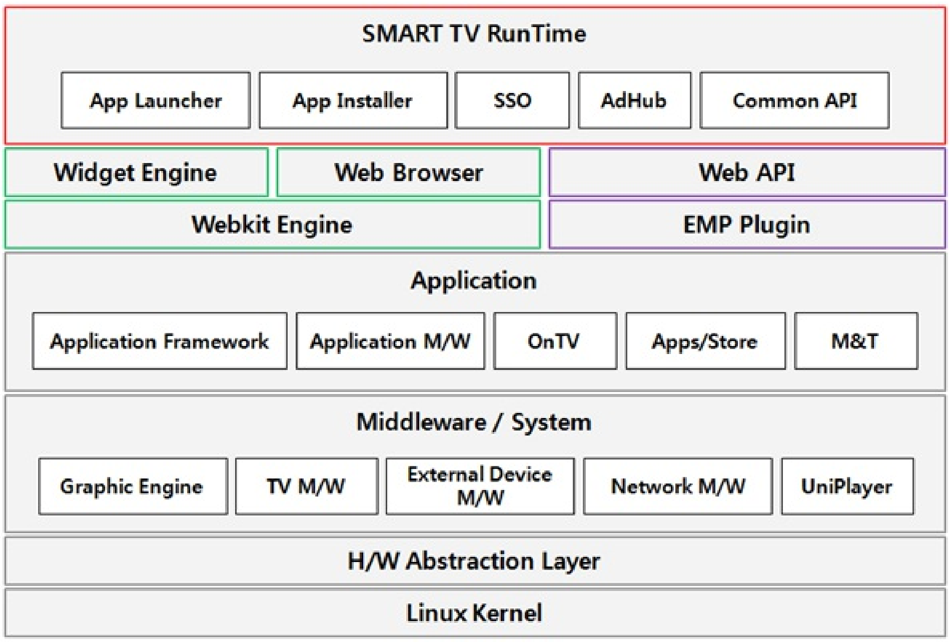

Figure 1: Basic structure

Web Engine based on Webkit, Widget Engine, EMP Plugin is important part of run App on Smart TV, and RunTime is driven with these things. RunTime manages Lifecycle of App which related to run, exit, install, delete, update.

Operation Flow

App executes from Widget Engine. And, Widget Engine provides **Send Event** between Apps. From here, Web Engine renders UI.

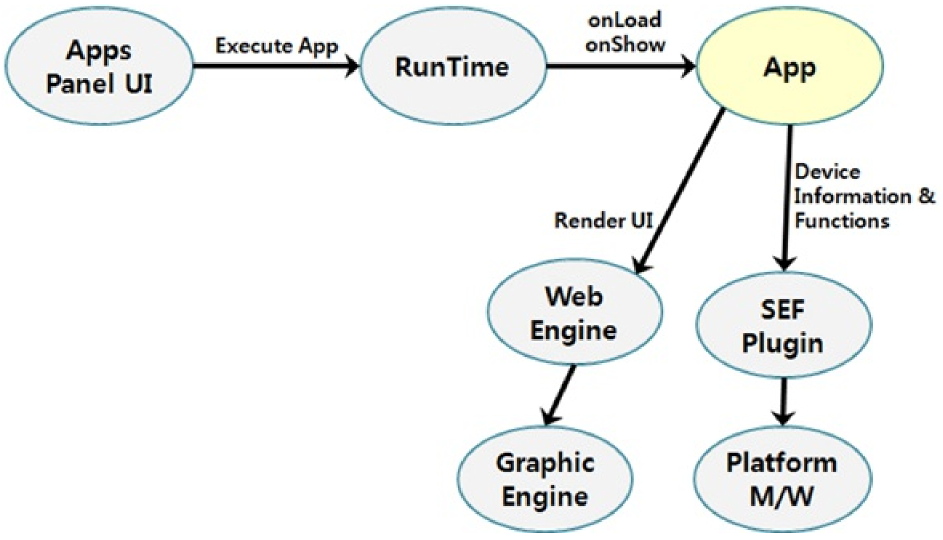

Figure 2: Operation flow

Device Specification

Specification of Samsung Smart TV is as follows.

Features	Specifications
OS	Linux kernel 2.6
Resolution	Graphic: 1280x720, 960x540 (32bit) Video: 1920x1080
Web Engine	Support HTML5, CSS3, DOM3

Which device do support “Multi-app”?

Supporting Multi-app device is as follows.

Years	Model	Support

Years	Model	Support
2012	6400 Series above	X
2013	6400 Series above	X
2014	6400 Series above	O

Multi-Tasking Application (Multi-app)

Basic Concept

App is in memory when destroy, so App run directly when run again.

All device is not supporting Multi-app. Supported device is displayed above.

Benefit of Applying

If you rerun, it runs faster. So it improves performance.

Remember previous App page, so easy to switch over Apps.

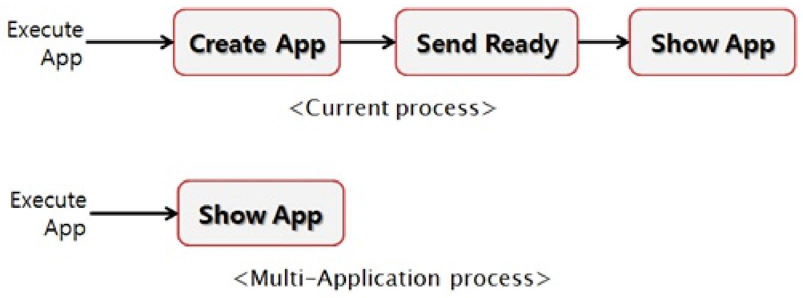

Figure 3: Benefit of applying

Basic Policy

App State

App has onload, onShow, onPause, onResume, onHide, onunload State.

- onload : App is initially driven into memory.
- onShow : Show the App which loaded into memory.
- onPause : App will hide and become Background status.
- onResume : App show on Background status.
- onHide : App is hidden before destroyed.
- onunload : App is completely removed on the memory.

App receives event, need to implementation according to State.

Flow of App State

When exit App, it will just onPause(Hide) without Destroy. So if App run again preview screen will onResume(Show) without Create.
Playing VOD will onPause(Hide) after Deactivate, When replaying it will Activate on the point which was playing.
User Terminate App with Exit button, App will destroy.
With limited memory, App will destroy.

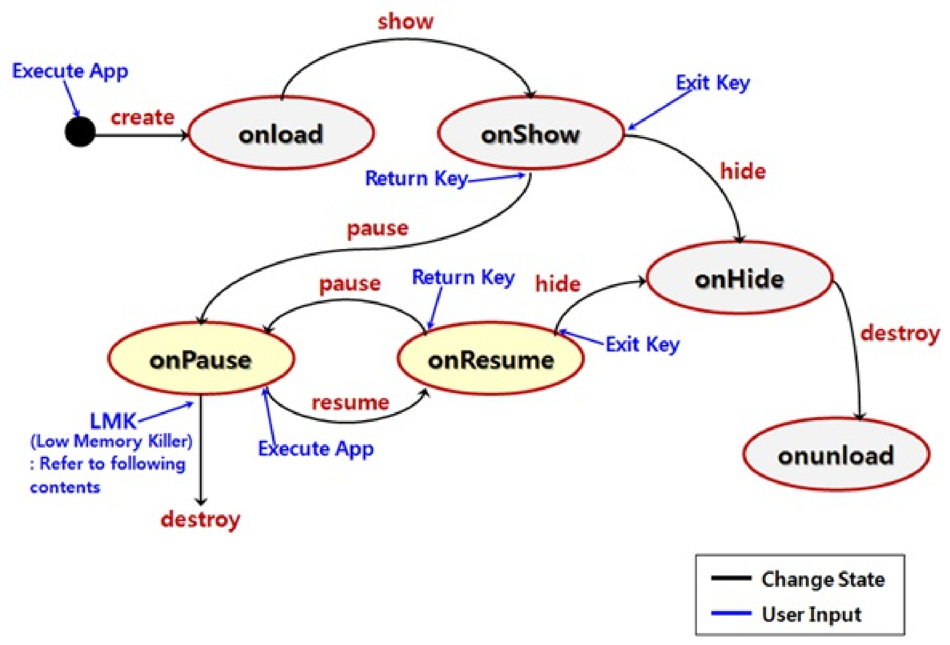

Figure 4: Flow of app state

Sequence of Execution App

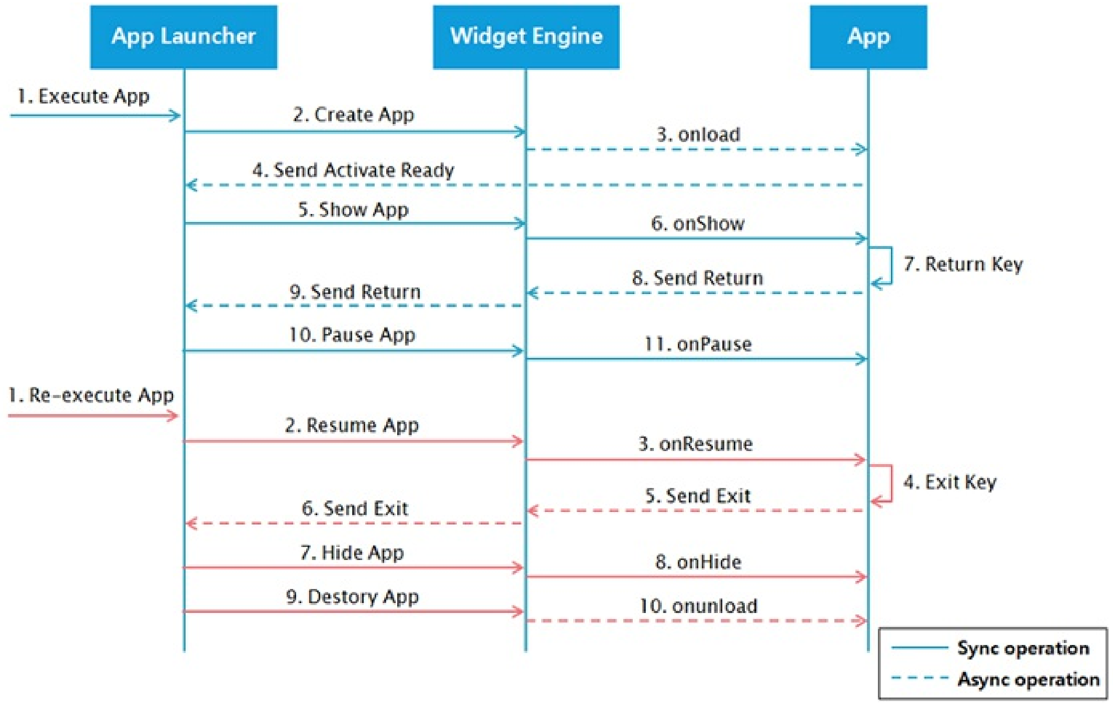

Figure 5: Sequence of execution app

How to apply?

Pre-Setting

For supporting Multi-app, **<multiapp>** tag is required in 'config.xml'.
<multiapp> tag indicates whether Multi-app or not. **y** is “support” , **n** is “do not support”. Default value is **n**.

Example

```
Example
<?xml version="1.0" encoding="UTF-8"?>
<widget>
    ...
    <multiapp>y</multiapp>
    ...
</widget>
```

App State for Implementation

App State for Implementation

onload/onunload

Description

onload

onload Event is called when Apps are pre-loaded, it is App’s Entry point. onload status is maintained until ready to Show. Using "window.location.search", you can get parameters.

onunload

onunload Event is called when App is removed in memory and completely terminated. When App rerun, it starts with onload status.

Caution

The following which control H/W resource will generate problem. Therefore you have to remove these things at onload state.

- Source change
- PIG setting
- Audio Mute/UnMute
- Video Mute/UnMute
- Channel change
- Media (Video, Music...) play by Uni-player
- Voice Help setting
- Gesture setting
- Register/Unregister Key

Prohibit animations such as Loading Bar.
Prohibit action which continuously connect network and connect server.
Prohibit change source at onunload

Usage

index.html

```
<body onload="Main.onLoad();" onunload="Main.onUnload();">
</body>
```

JavaScript

```
var Main = {};

Main.onLoad = function () {
    alert("Create App");
    alert("Params: " + window.location.search);
};

Main.onUnload = function () {
    alert("Destroy App");
};
```

onShow(event)

Description

The Event is called when App is first run.

State flow

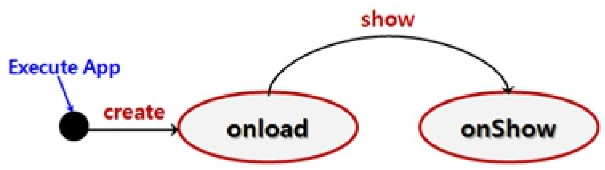

Figure 6: onShow state flow

Usage

Exit button usage

```
window.onShow = function (event) {
    alert("Event type = " + event.type); // deliver to "onShow"
    alert("Parameter = " + event.data);  // deliver to same form as window.location.search
};
```

onPause (event)

Description

When show status (onShow, onResume) App is hidden, Event is called to process something. If playing video, it will be paused automatically. And when enter onResume status, it will be resumed.

Caution

The same as onload state, after change onPause state, you have to use for controlling H/W resources.

- Source change
- PIG setting
- Audio Mute/UnMute
- Video Mute/UnMute
- Channel change
- Media (Video, Music...) play by Uni-player
- Voice Help setting
- Gesture setting
- Register/Unregister Key

Stop animations such as Loading Bar.
Stop connect to server continuously.
Stop action which continuously check network connection.
Because onPause state can be remained long time and disconnected network, we recommend to implement the following operations.

- Disconnect session with server and logout service account.

If there is no connection with server during the long time, video playing can be in trouble, even though you press pause while a video is playing. You need to prepare in this case, so the following methods are recommended.

- Return to previous screen, when change to onPause state.
- Display play error message and then return to previous screen, if playing video is failed in onResume state.

Stop timer (setTimeout, setInterval)
Close IME

State Flow

When being the App showing, all case except pushing Exit key, status will go back onPause status. When App is exited by Return key or SMART HUB key, status will change to onResume.

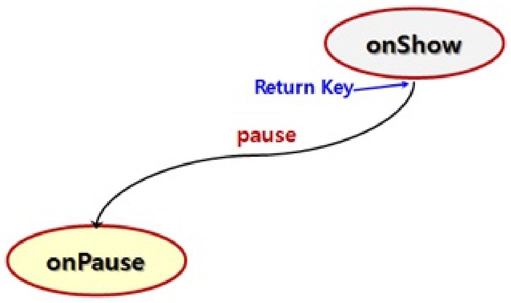

Figure 7: onPause state flow

Usage

```
onPause usage
window.onPause = function (event) {
    alert("Event type = " + event.type);  // deliver to "onPause"
};
```

onResume(event)

Description

onResume Event is called when Hide status App is shown. At this time if you have something to process, you can use onResume event. If necessary, you perform the stopped action again when onPause event is called. You can bind it same as onShow event. But if app have to process only one time, you should separate onShow and onResume.

State flow

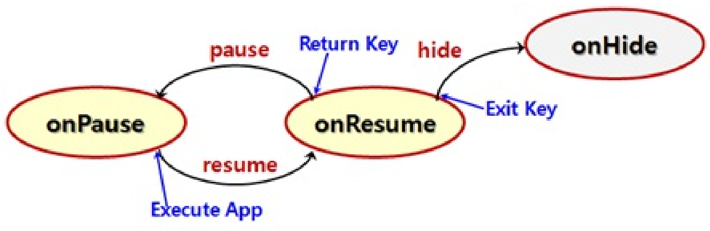

Figure 8: onResume state flow

Caution

It is need re-start implementations that stopped operation at onPause state.

Re-connect session with server and login service account.

We recommend when occurred error timing of resume video, return to previous screen.

If display error popup at onPause, should be close popup.

Need to confirm connection with server.

If needs time, update time information.

If use SSO, check up SSO and bind service account state.

If support multi-language, check up language.

Usage

```
onResume usage
window.onResume = function (event) {
   alert("Event type = " + event.type); // deliver to "onResume"
   alert("Parameter = " + event.data);  // deliver to same form as window.location.search
};
```

onHide(event)

Description

This event is called when App is hidden before destroyed. Even though App is shown it is called before Destroying.

State Flow

When Exit key is pushed on onShow, onResume status, onShow will change **onUnload** go through onHide status.

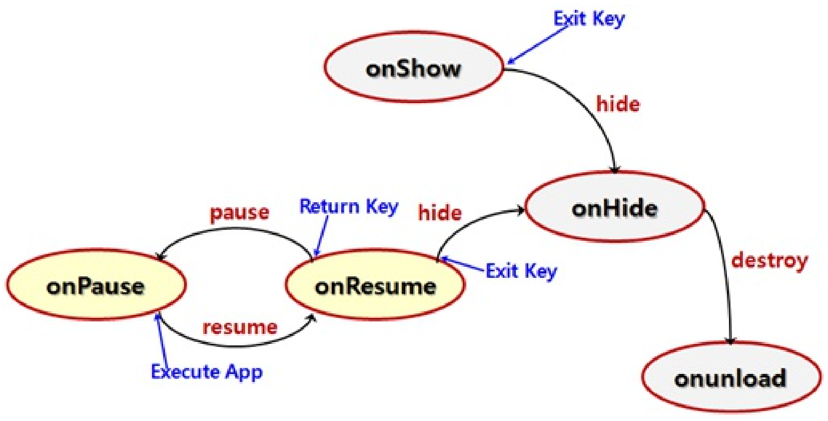

Figure 9: onHide state flow

Caution

It is below implementations that stopped operation at onHide state.

Close IME

Close SSO screen

Usage

```
onHide usage
window.onHide = function (event) {
    alert("Event type = " + event.type);  // deliver to "onHide"
};
```

Case of state flow

Case 1

After execute App, destroy by LMK.

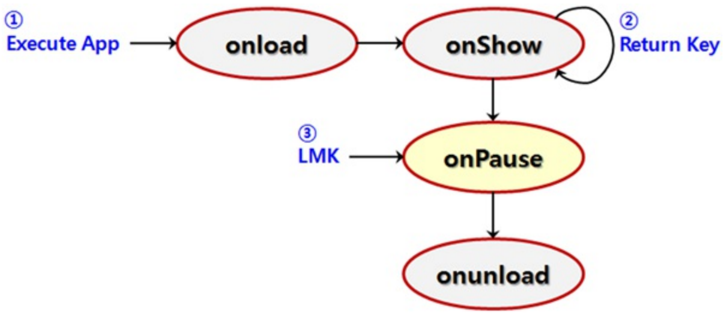

Figure 10: State flow - case 1

Case 2

After execute App, terminated by user Exit key.

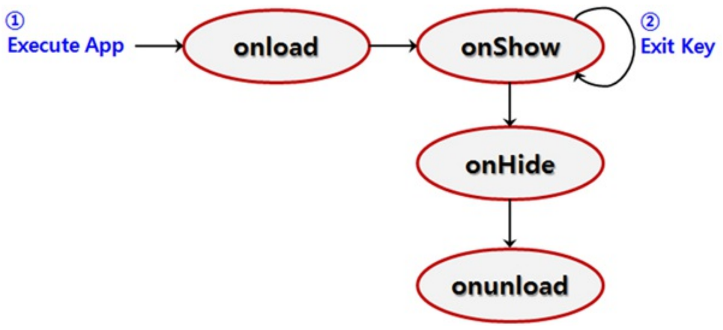

Figure 11: State flow - case 2

Case 3

After preload is completed, execute App.

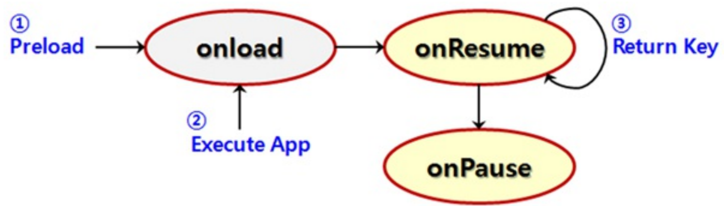

Figure 12: State flow - case 3

Case 4

After preload is completed, destroy by LMK.

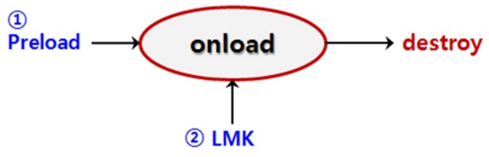

Figure 13: State flow - case 4

Example of Implementation

Description

To minimize modification, recommend this method. If there are no differences to process on 'onResume/onShow, onPause/onHide', don't need to distinguish event.

Example

config.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>

<widget xmlns="http://www.samsung.com/">
    <cpname itemtype="string"/>
    <cplogo itemtype="string"/>
    <cpauthjs itemtype="string"/>
    <ThumbIcon itemtype="string">icon/sampleIcon_106_87.png</ThumbIcon>
    <BigThumbIcon itemtype="string">icon/sampleIcon_115_95.png</BigThumbIcon>
    <ListIcon itemtype="string">icon/sampleIcon_85_70.png</ListIcon>
    <BigListIcon itemtype="string">icon/sampleIcon_95_78.png</BigListIcon>
    <category itemtype="string"/>
    <autoUpdate itemtype="boolean">n</autoUpdate>
    <ver itemtype="string">0.100</ver>
    <mgrver itemtype="string"/>
    <fullwidget itemtype="boolean">y</fullwidget>
    <multiapp itemtype="boolean">y</multiapp>
    <type itemtype="string">user</type>
    <srcctl itemtype="boolean">y</srcctl>
    <ticker itemtype="boolean">n</ticker>
    <childlock itemtype="boolean">n</childlock>
    <videomute itemtype="boolean">n</videomute>
    <dcont itemtype="boolean">y</dcont>
    <widgetname itemtype="string">MultiApp</widgetname>
    <description itemtype="string"/>
    <width itemtype="string">960</width>
    <height itemtype="string">540</height>
    <author itemtype="group">
        <name itemtype="string"/>
        <email itemtype="string"/>
        <link itemtype="string"/>
        <organization itemtype="string"/>
    </author>
</widget>
```

Main.js

```javascript
var widgetAPI = new Common.API.Widget();
var tvKey = new Common.API.TVKeyValue();

var Main = {};

Main.onLoad = function () {
    // Enable key event processing
    this.enableKeys();

    // Parsing parameter 'window.location.search', if you need.

    // Add for supporting Multi-app
    window.onShow = Main.onShow;
    window.onResume = Main.onShow;
    window.onHide = Main.onHide;
    window.onPause = Main.onHide;

    // Prohibit changing source
    // Prohibit set PIG
    // Prohibit Mute/UnMute audio
    // Prohibit Mute/UnMute video
    // Prohibit Mute/UnMute video
    // Prohibit change channel
    // Prohibit use media(Video, Music…) play by Uni-player
    // Prohibit set voice help
    // Prohibit set gesture
    // Prohibit register/unregister keys

    widgetAPI.sendReadyEvent();
};

Main.onUnload = function () {};

Main.onShow = function (event) {
    alert("Event type = " + event.type);
    alert("Event data = " + event.data);

    if (event.type == "onShow") {
        // onShow state

        // Parsing parameter 'event.data', if you need
        // Register keys, if you need
    } else if (event.type == "onResume") {
        // onResume state

        // Parsing parameter 'event.data', if you need
        // Register keys, if you need
    }
};

Main.onHide = function (event) {
    alert("Event type = " + event.type);

    if (event.type == "onHide") {
        // onHide state
    } else if (event.type == "onPause") {
        // onPause state

        // Stop timer (e.g. setTimeout, setInterval)
    }
};

Main.enableKeys = function () {
    document.getElementById("anchor").focus();
};

Main.keyDown = function () {
    var keyCode = event.keyCode;
    alert("Key pressed: " + keyCode);

    switch (keyCode) {
        case tvKey.KEY_RETURN:
        case tvKey.KEY_PANEL_RETURN:
            alert("RETURN");
            widgetAPI.sendReturnEvent();
            break;
        case tvKey.KEY_LEFT:
          alert("LEFT");
          break;

        case tvKey.KEY_RIGHT:
          alert("RIGHT");
          break;

        case tvKey.KEY_UP:
          alert("UP");
          break;

        case tvKey.KEY_DOWN:
          alert("DOWN");
          break;

        case tvKey.KEY_ENTER:
        case tvKey.KEY_PANEL_ENTER:
          alert("ENTER");
          break;

        default:
          alert("Unhandled key");
          break;
      }
    };
```

LMK (Low Memory Killer)

What's LMK?

Concept

LMK stands for Low Memory Killer. When extra memory of available memory drops below a certain value, LMK removed App from memory in order to secure the memory.

Condition

Being generated in the onPause condition.
Occurs when the available memory is less than 120M.

Priorities

'onPause' status App
App running from the old time.

Flow of App State

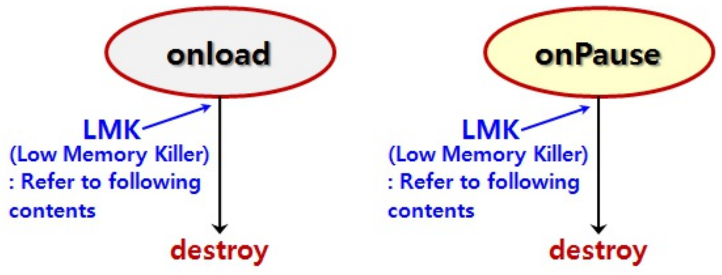

Figure 14: Flow of app state

In onPause status, when occuring LMK, perform end logic.
When occurred LMK, removed on memory, but do not process App state.