Demo Application UI Guide (Video Player)

Version 0.6.0

Display Audio

Solution Team

Release information

The following changes have been made to this document.

Change History

Date	Change
06 Dec. 2017	First release for v0.1.0
15 Dec. 2017	Update Storage Event v0.2.0
07 Feb. 2018	Update scenario v0.3.0
14 Nov 2018	PlayList null point fix v0.4.0 Storage Remove fix v0.4.0. Player library null point fix v0.4.0. Add Vide Mute Function v0.4.0. Add MakeThumbnail() v0.4.0.
15 Feb 2019	Automatically adjust UI Size (1024x600, 1920x720) v0.5.0 Add AvSync v0.5.0 Add VideoSpeed v0.5.0
28 May 2019	Get ThumbNail In Video File v0.6.0

Proprietary Notice

Information in this document is provided solely to enable system and software implementers to use Nexell products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

Nexell reserves the right to make changes without further notice to any products herein.

Nexell makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Nexell 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 which may be provided in Nexell 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. Nexell does not convey any license under its patent rights nor the rights of others. Nexell products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Nexell product could create a situation where personal injury or death may occur. Should Buyer purchase or use Nexell products for any such unintended or unauthorized application, Buyer shall indemnify and hold Nexell and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Nexell was negligent regarding the design or manufacture of the part.

Contact us

[11595] BundangYemiji Bldg. 12F, 31 Hwangsaeul-ro 258 beon gil, Bundang-gu, Sungnam-city, Gyeonggi-do, Korea.

TEL: 82-31-698-7400

FAX:82-31-698-7455

http://www.nexell.co.kr

Contents

Chap 1. Overview

1.1 Overview

This document describes how to use NxVideoPlayer application.

NxVideoPlayer makes video file list by using sqliteutils that reads database written by media scanner.

NxVideoPlayer also makes xml file that stores last playing information to play again from last state.

1.2 UI Guide

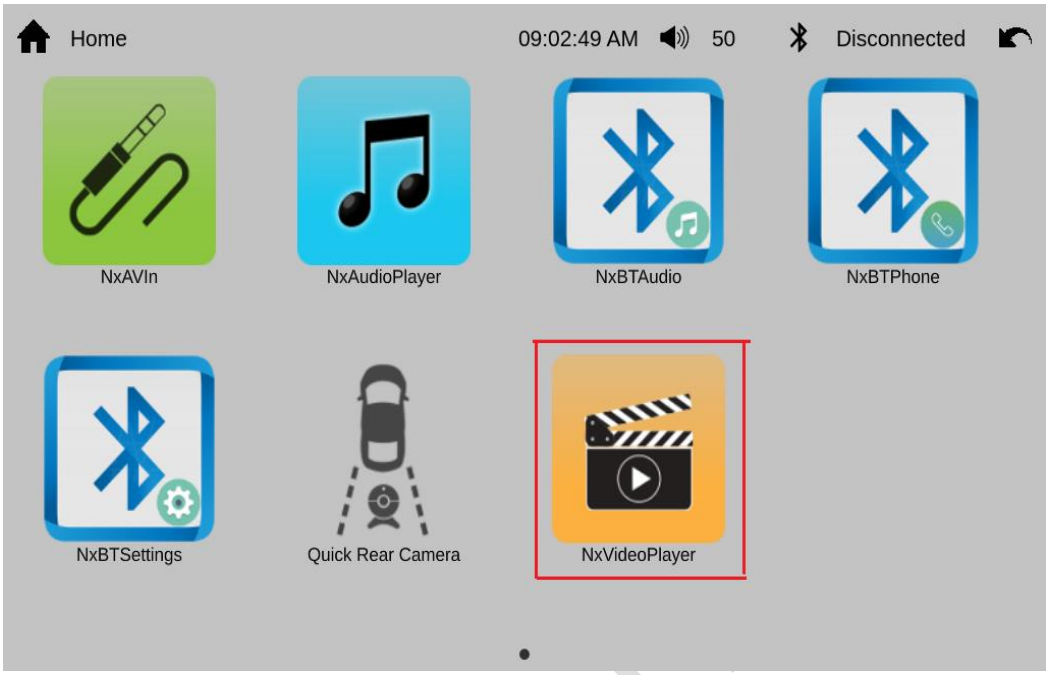
1.2.1 Features

- Video file play, seek, pause, and resume
- Video speed
- Make video file list including USB and SD-card.
- Auto play from last state, if available.
- Shows subtitle, if available.

1.2.2 Execute

NxVideoPlayer is executed by clicking icon in the launcher.

When some application is clicked, launcher hides from screen and clicked application appears.

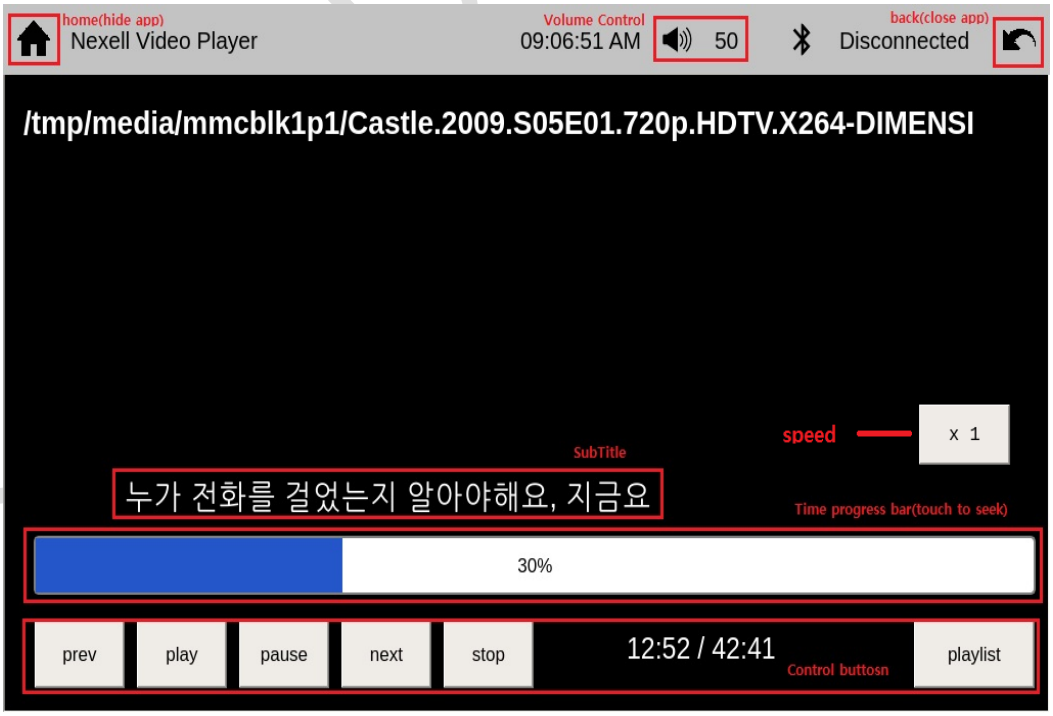


1.2.3 UI Details

GUI is shown like below picture after NxVideoPlayer is executed.

home (hide app) does not mean application window operates hide.

NxVideoPlayer disappears behind other application by changing Z-order.

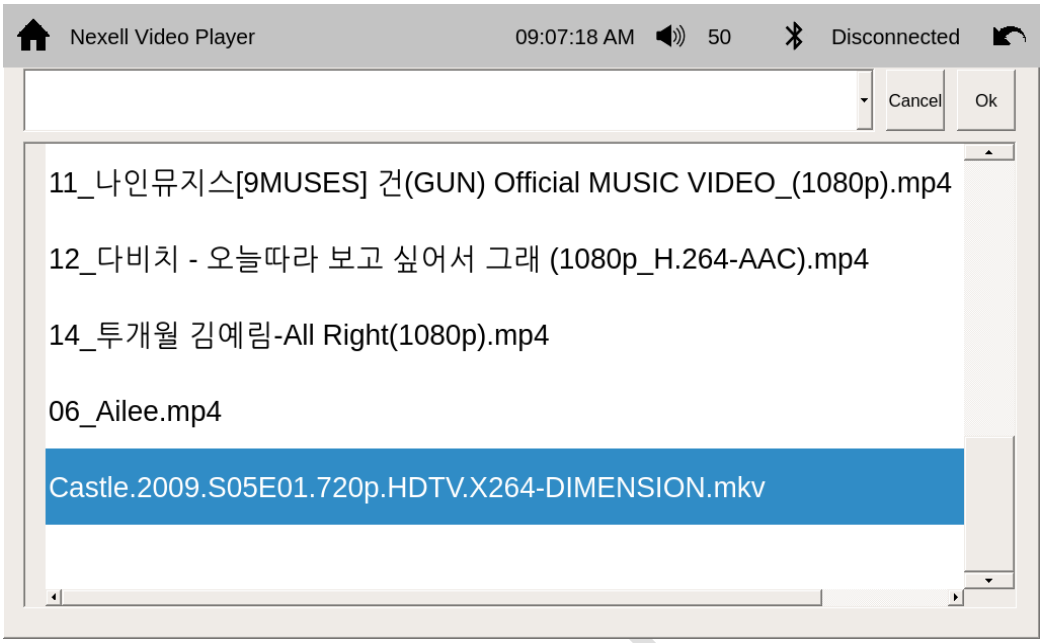


1.2.4 UI Playlist Details

Playlist UI is shown by clicking playlist button.

List can be scrolled down, and items in list can be played by double-clicking or

select(item click) and Ok button.



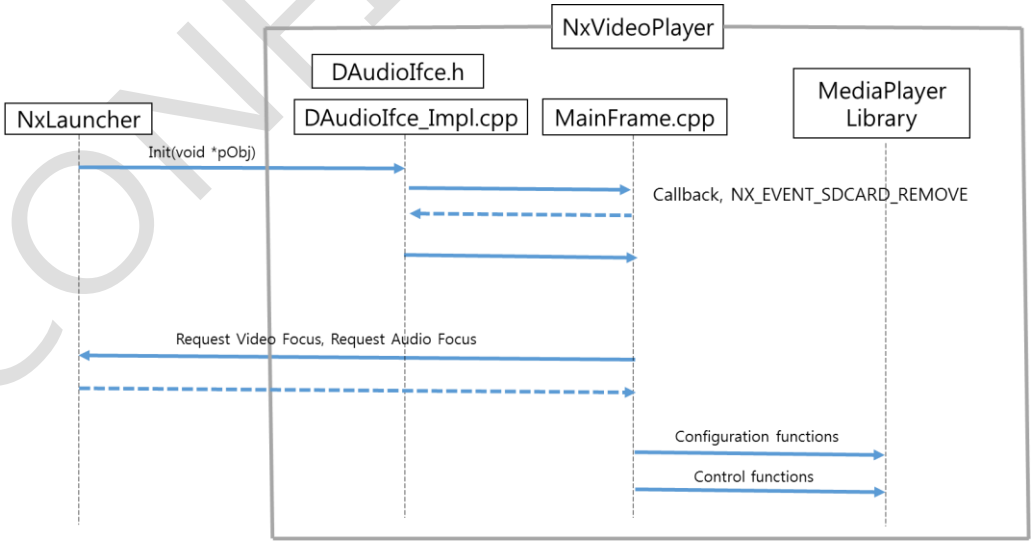
1.3 Flow

1.3.1 Application Overall

NxVideoPlayer communicates through NxLauncher .

RequestVideoFocus is needed, when application wants to appear on screen.

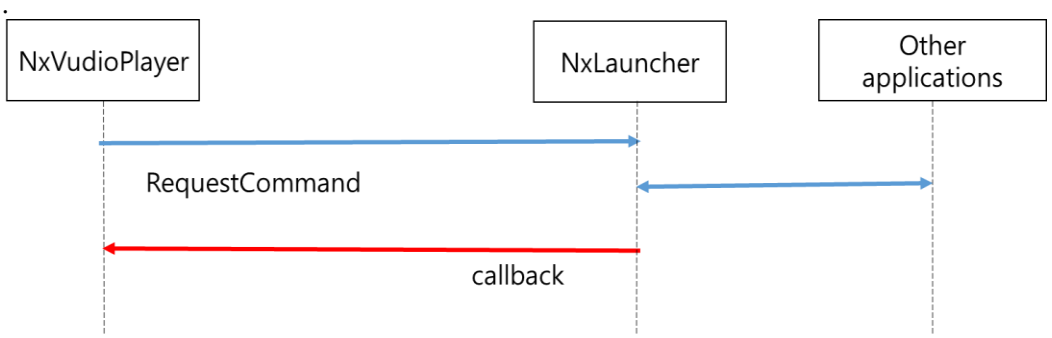
In particular, to use an Audio Device, must return RequestAudioFocus.



1.3.2 Application Switching

Each application communicates with NxLauncher .

NxLauncher is used for changing Z-order of application.



Chap 2. GUI Guide

2.1 Status Bar

NxStatusBar is used for Status Bar.

2.1.1 Home

When Home button clicked, current application disappears by changing launcher's Z-order to top.

In the home button callback function of NxStatusBar, NxVideoPlayer uses NxLauncher to request to display the program .

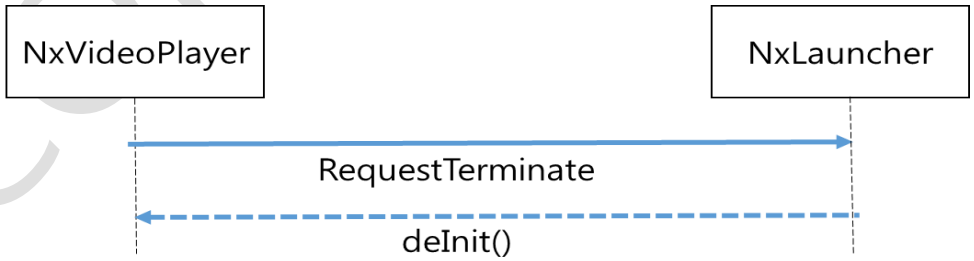
2.1.1.1 Flow

2.1.2 Back

When Back button clicked, NxVideoPlayer closes, and other application(previously top) appears.

In the nearest routine, NxVideoPlayer uses NxLauncher to signal loss of audio focus, loss of video focus, and process removal .

2.1.2.1 Flow



2.2 Media Player Control

In NxVideoPlayer application, Media Player library is wrapped by CNX_MoviePlayer class.

2.2.1 Initialization

NxVideoPlayer must get Audio Focus successfully before initialize Media Player lib.

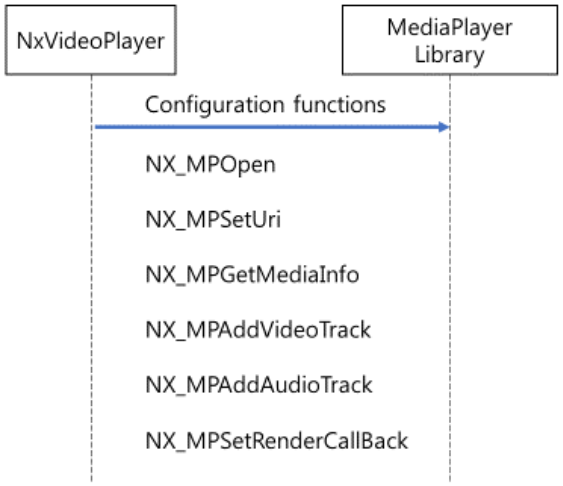
Media Player Initialization is done by CNX_MoviePlayer->InitMediaPlayer

InitMediaPlayer function uses Media Player library's Configuration functions in following order, NX_MPOpen, NX_MPSetUri, NX_MPGetMediaInfo, NX_MPAddVideoTrack, NX_MPAddAudioTrack.

All Media Player library functions must return MP_ERR_NONE.

NX_MPAddVideoTrack is used for video and NX_MPAddAudioTrack is used for audio.

2.2.1.1 Flow

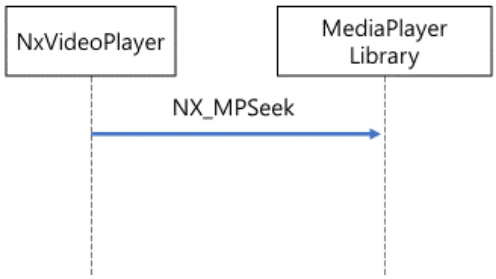


2.2.2 Progress Bar

When Progress Bar is clicked, Bar UI and video position are changed to corresponding value.

NX_MPSeek in Media Player library is used.

2.2.2.1 Flow



2.2.3 Prev Button

When prev button is clicked, NxVideoPlayer stops playing and plays previous video file.

Index of video file list is set to previous video file.

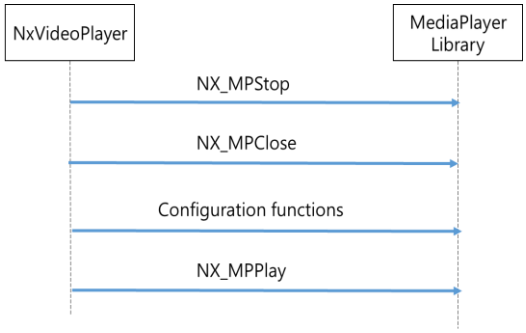
NX_MPStop and NX_MPClose in Media Player library are used.

Initialization is done with previous video file like above explanation.

NX_MPPlay in Media Player library is used after Initialization is succeed.

Media Player library callback rendering will be explained in Play Button.

2.2.3.1 Flow



2.2.4 Play Button

When play button is clicked, NxVideoPlayer plays video file.

In NxVideoPlayer application, play button sequence contains Initialization of Media Player library.

Just after Initialization of Media Player lib is done, NxVideoPlayer updates UI information such as, progress bar, title, and so on.

If Initialization is failed, NX_MPClose in Media Player library is called and Initialization is tried again with next video file in list.

When NxVideoPlayer is already playing video file, play button does nothing.

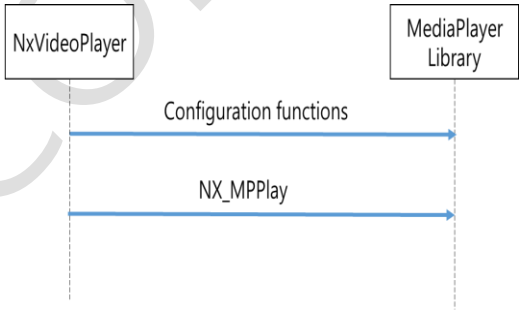
When NxVideoPlayer is in pause, play button just call NX_MPPlay without Initialization.

Media Player Library calls callback function for rendering after NX_MPPlay is called, and NxVideoPlayer draws screen.

Media Player Library calls callback function for rendering is pended until NxVideoPlayer completes drawing screen.

Calling callback function for rendering is repeated while NxVideoPlayer is playing video.

2.2.4.1 Flow

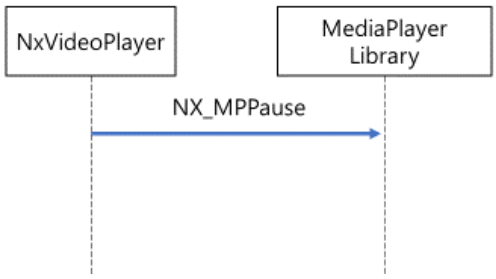


2.2.5 Pause Button

When pause button is clicked, NxVideoPlayer pauses video playing.

NX_MPPause in Media Player library is used.

2.2.5.1 Flow



2.2.6 Next Button

When next button is clicked, NxVideoPlayer stops playing and plays next video file.

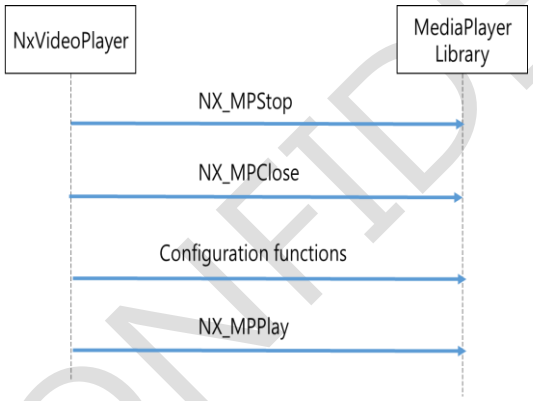
Index of video file list is set to next video file.

NX_MPStop and NX_MPClose in Media Player library are used.

Initialization is done with next video file like above explanation.

NX_MPPlay in Media Player library is used.

2.2.6.1 Flow

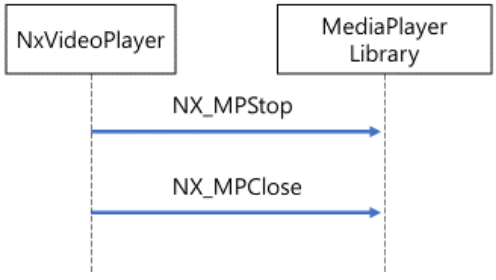


2.2.7 Stop Button

When stop button is clicked, NxVideoPlayer stops playing video.

NX_MPStop and NX_MPClose in Media Player library are used.

2.2.7.1 Flow



2.2.8 Playlist Button

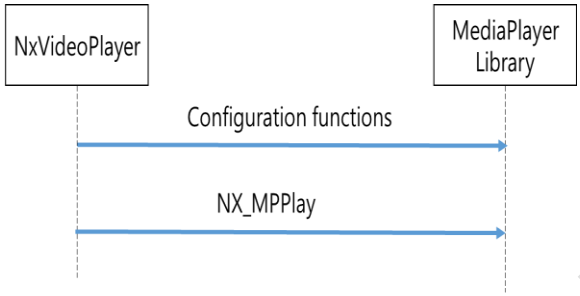
When playlist button is clicked, NxVideoPlayer shows playlist UI.

If NxVideoPlayer was playing video, it continues until some video file is selected.

If file is selected, NxVideoPlayer stops playing video and plays selected video file.

2.2.8.1 Flow

The case that file is selected.



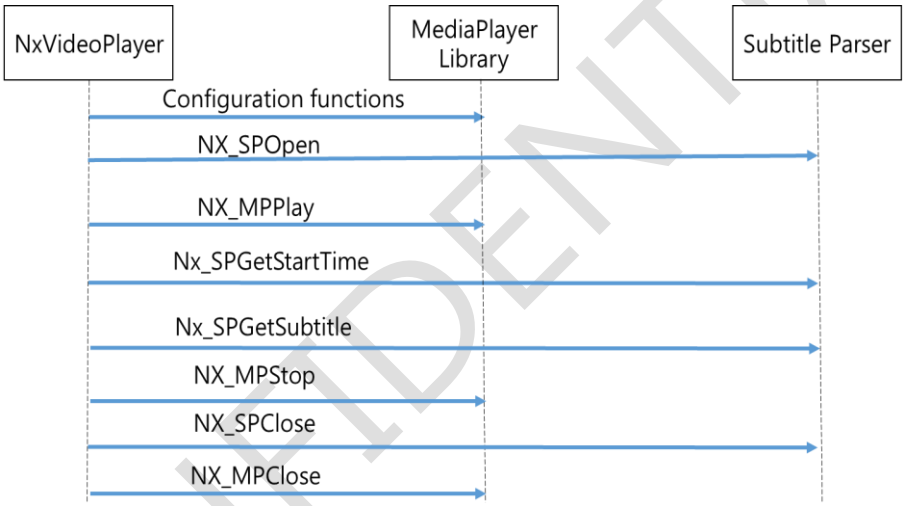
Chap 3. Subtitle

3.1 Overview

In NxVideoPlayer application, Subtitle Parser library is wrapped by CNX_MoviePlayer class.

NX_SPGetStartTIme and NX_SPGetSubtitle are called repeatedly while NxVideoPlayer is playing video.

3.2 Application Overall Flow



Chap 4. Storage Event

4.1 OverView

NxVideoPlayer detects storage events.

Used storage events are removing external storage(USB, SD-card) and media scan done.

Media scan done event is received, whenever storage status is changed such as inserting and removing.

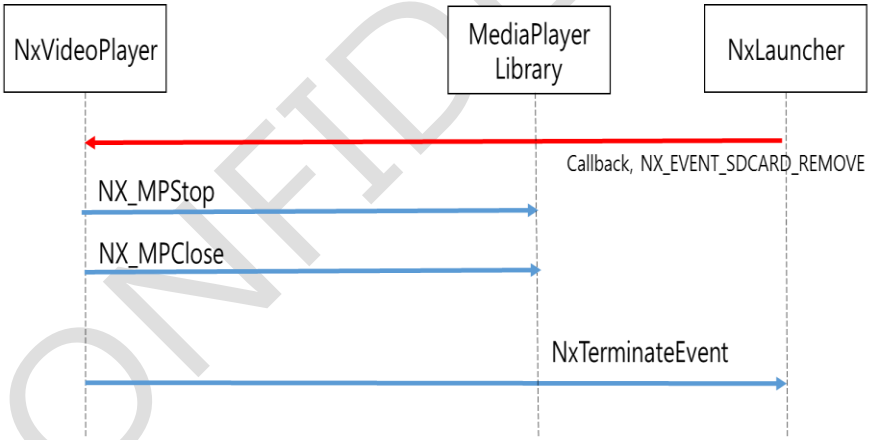
If removing storage event is received, NxVideoPlayer is closed.

If media scan done event is received, NxVideoPlayer refreshes media file list.

4.2 Flow

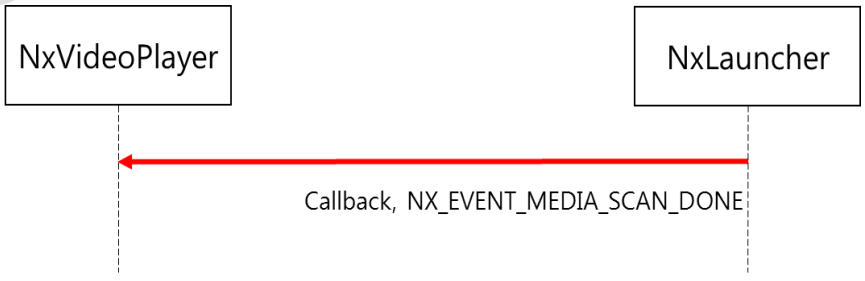
4.2.1 Removing Storage Event

NX_EVENT_SDCARD_REMOVE and NX_EVENT_USB_REMOVE are used.



4.2.2 Inserting Storage Event

When NX_EVENT_MEDIA_SCAN_DONE is received, media file list is refreshed.



Chap 5. ADD Function

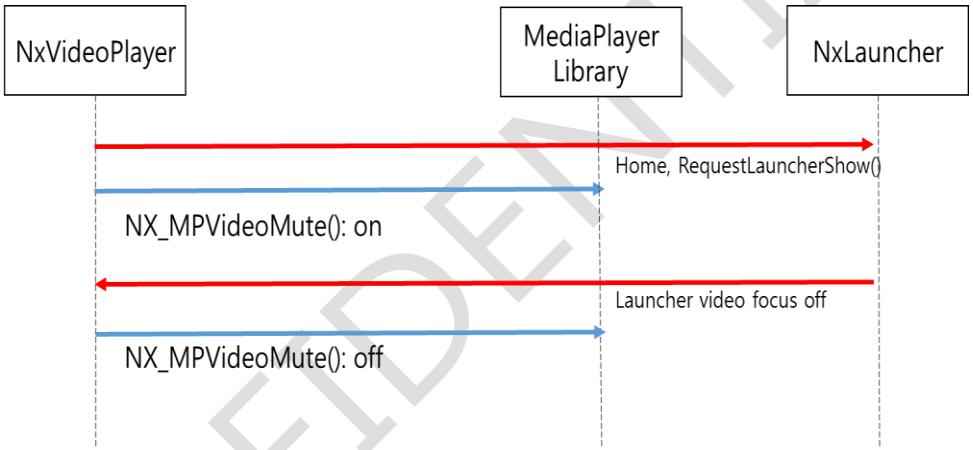
5.1 Add Function

5.1.1 Video Mute

Click the home button on the status bar to run video mute.

When video mute, only audio is played.

5.1.1.1 Flow



.

5.1.2 CheckThumbnailInVideoFile

Check if the thumbnail exists in the videofile..

Prototype:

CheckThumbnailInVideoFile(const char *pInFile, const char *pOutFile, int maxWidth,
int *pThumbnailWidth, int *pThumbnailHeight)

Parameters:

const char *pInFile: In File (input).

int *_t pThumbnailWidth: thumbnailWidth (output).

int *_t pThumbnailHeight: thumbnailHeight (output).

Return value:

return 1: have thumbnail.

return 0: no thumbnail

5.1.3 GetThumbnail

Get Thumbnail in the videofile..

Prototype:

GetThumbnail(const char *pInFile, const char *pOutFile)

Parameters:

const char *pInFile: In File (input).

const char *pOutFile: Out File (output).

Return value:

If success return 0, otherwise return error.

5.1.4 Make Thumbnail

This function creates a thumbnail(jpeg file).

Prototype:

Int MakeThumbnail(const char *pInFile, const char *pOutFile, int maxWidth,
int maxHeight, int timeRatio)

Parameters:

const char *pInFile: In File (input).

const char *pOutFile: Out File(input).

int32_t maxWidth: Max Width (input). (64 – 1280)

int32_t maxHeight: Max Height (input). (64 – 1280)

int32_t timeRatio: Time Ratio (input). (1 – 99)

Return value:

If success return 0, otherwise return error.

5.1.5 SetAudioSync

This function controls the AudioSync.

Prototype:

Int SetAudioSync(int64_t syncTimeMs)

Parameters:

Int64_t syncTimeMS: +,- ms (input).

+: audio is faster.

-: audio is slower.

Return value:

If success return 0, otherwise return error.

5.1.6 GetVideoSpeedSupport

This function checks if video speed is available.

- Support file: .avi, .mkv, .mp4
- Support codec: h264, mpeg4

Prototype:

Int GetVideoSpeedSupport()

Parameters:

None

Return value:

If support return 0, otherwise return error.

5.1.7 SetVideoSpeed

This function controls the video speed.

Prototype:

Int SetVideoSpeed(float Speed)

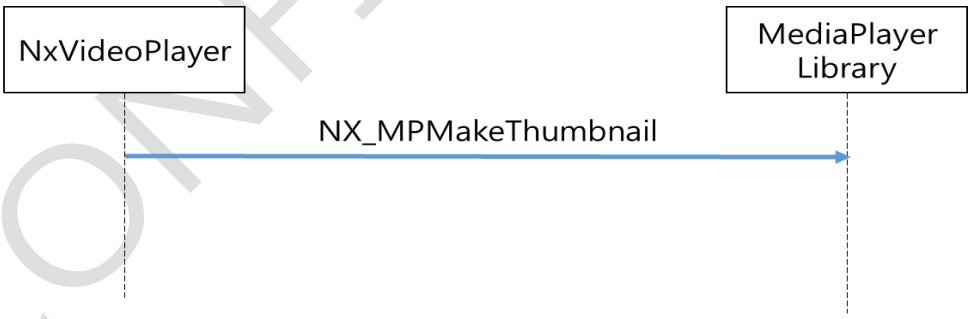
Parameters:

float Speed: 2,3,4,5,6,8….(input).

Return value:

If success return 0, otherwise return error.

5.1.7.1 Flow



.

Chap 6. Known Issues

6.1 Known Issues

End Of Stream occasionally occurs twice during video speed control.

6.2 To Do List

Supporting Multi Language for subtitle and playlist UI.