SDK API User's Guide (ETC)

Version 0.6.0

Display Audio

Solution Team

Release information

The following changes have been make to this document.

Change History

Date	Change
04 Dec. 2017	First release for v1.0.0

Proprietary Notice

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

이 문서는 Display Audio 개발을 위하여 제공되는 기타 SDK 들에 대해서 설명한 문서이다.

1.2 Feature

제공되는 SDK 들은 다음과 같다.

-. libnxconfig	: private configuration (Android SharedPrefrence Modeling)
-. libnxsqlutils	: SQLite database access. (support reading operation)
-. libnxdaudioipc	: IPC Server / Client Engine, IPC engine wrapper

Chap 2. Private Configuration

2.1 Overview

Application 들의 개별 Configuration 을 관리하여 줄 수 있는 libnxconfig 를 제공하여 준다. 해당 library 는 Thread-safe 하지 못함으로 application 간 configuration 을 공유하는 목적으로는 사용이 부적합하다.

2.2 APIs

2.2.1 Open()

```
virtual int32_t Open(
        const char *pFile
) = 0;
```

Description

Open XML configuration file.

Parameter

-. pFile	: XML Configuration file.

Parameter

0 is successful. -1 is failed.

2.2.2 Close()

```
virtual void Open(
        void
) = 0;
```

Description

Close XML configuration file.

Parameter

None.

Parameter

None.

2.2.3 Write()

```
virtual int32_t Write(
         const char *pKey,
         char *pValue
) = 0;
```

Description

Write Configuration.

Parameter

-. pKey	: configuration key.
-. pValue	: configuration value.

Parameter

0 is successful. -1 is failed.

2.2.4 Read()

```
virtual int32_t Read(
         const char *pKey,
         char **ppValue
) = 0;
```

Description

Read Configuration.

Parameter

-. pKey	: configuration key.
-. ppValue	: configuration value.

Parameter

0 is successful. -1 is failed.

2.2.5 Remove()

```
virtual int32_t Read(
         const char *pKey
) = 0;
```

Description

Remove Configuration.

Parameter

-. pKey	: configuration key.

Parameter

0 is successful. -1 is failed.

2.2.6 Dump()

```
virtual void Dump(
         void
) = 0;
```

Description

Dump configuration for debugging.
Parameter None.
Parameter None.

Chap 3. SQL

3.1 Overview

Application 내에서 Database 를 읽기 위한 용도로 사용할 수 있는 SQLite Wrapper 를 제공하여 준다. 이는 해당 database 내의 table 전체에 대해서 query 하여 읽도록 설계되어있다. SQLite 는 File 기반의 database 로 하나의 process 가 database 에 접근하는 동안 data 의 무결성을 보장한다.

3.2 APIs

3.2.1 NX_SQLiteGetData()

```
int32_t NX_SQLiteGetData(
        const char *pDatabase,
        const char *pTable,
        int32_t (*cbFunc)(void*, int32_t, char**, char**),
        void *pObj = NULL,
) = 0;
```

Description

Access database using SQLite.

Parameter

-. pDatabase	: database name.
-. pTable	: table name.
-. cbFunc	: result data callback

int32_t cbFunc(void *pObj, int32_t iColumnNum, char **ppColumnValue, char **ppColumnName)

-. pObj	: private handle
-. iColumnNum	: column number of table.
-. ppColumnValue	: column value of table.
-. ppColumnName	: column name of table.

-. pObj	: private handle.

Return Value

0 is successful, -1 is failed.

Chap 4. IPC

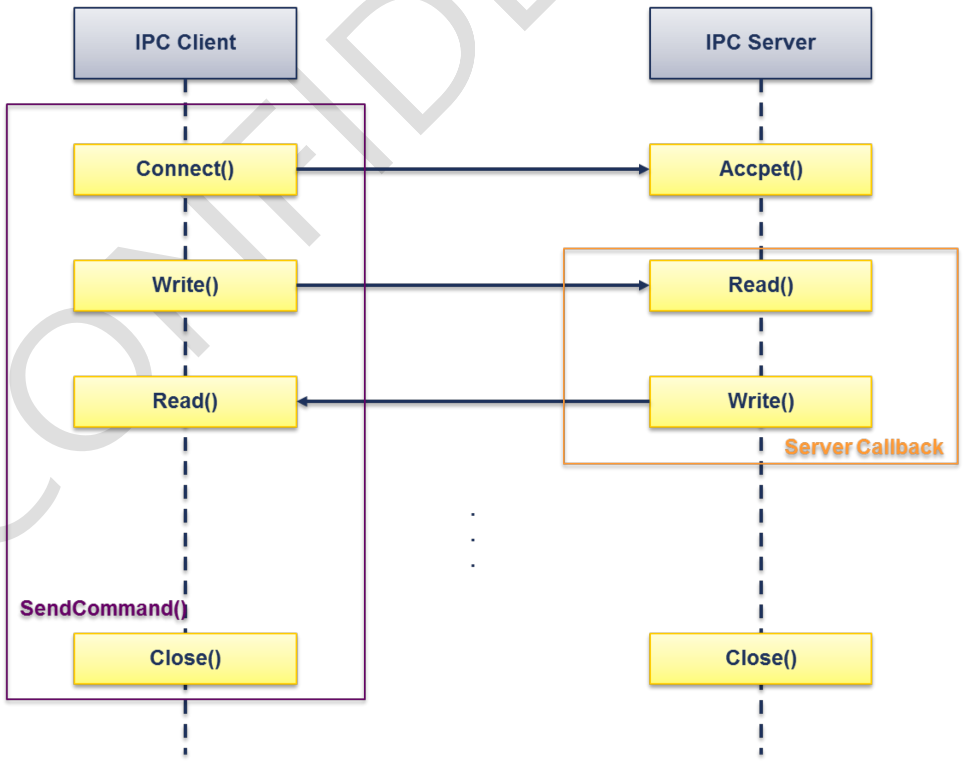
4.1 Overview

Display Audio Manager 와 개별 Application 간의 통신을 위한 IPC Engine 을 제공한다. Display Audio 내에서는 이를 통하여 Display Audio Manager 와 통신을 하며 이를 통하여 화면 전환 및 기타 Command 들을 전달하는 용도로 사용한다.

제공되는 IPC Engine 은 Unix Domain Socket 기반이며, 동인한 Socket Name 에 대한 단일 connection 만을 지원하도록 설계되어있다.

4.2 Block Diagram

IPC 는 Server 와 Client 관련 API 를 제공하며, 아래와 같은 구조로 되어있다.



4.3 APIs

4.3.1 IPC Manager

IPC 통신을 위한 기본적인 Library 이다.

4.3.1.1 GetIpcManagerHandle()

```
extern NX_IIpcManager* GetIpcManagerHandle(
        void
);
```

Description

Get IPC Manager handle.

Parameter

None.

Return Value

NX_IIPcManager NULL is failed

4.3.1.2 Write()

```
virtual int32_t Write(
        int32_t iSock,
        int32_t *pBuf,
        int32_t iSize
) = 0;
```

Description

Write data.

Parameter

-. iSock	: socket file descriptor.
-. pBuf	: buffer for writing.
-. iSize	: writing size.

Return Value

If it is successful, return value is written size.

4.3.1.3 Read()

```
virtual int32_t Read(
        int32_t iSock,
        int32_t *pBuf,
        int32_t iSize
) = 0;
```

Description

Read data.

Parameter

-. iSocket	: socket file descriptor.
-. pBuf	: buffer for reading.
-. iSize	: buffer size.
Return Value	
If it is successful, return value is read size.	

4.3.1.4 StartServer()

```
virtual int32_t StartServer(
        const char *pSock
) = 0;
```

Description

Start IPC server.

Parameter

-. pSock : socket name.

Return Value

If it is successful, return value is zero.

4.3.1.5 StopServer()

```
virtual int32_t StopServer(
        void
) = 0;
```

Description

Stop IPC server.

Parameter

None.

Return Value

If it is successful, return value is zero.

4.3.1.6 RegServerCallbackFunc()

```
virtual void RegServerCallbackFunc(
        int32_t (*cbFunc)( int32_t, uint8_t *, uint8_t *, int32_t, void * ),
        void *pObj;
) = 0;
```

Description

Register callback for Processing server data.

Parameter

-. cbFunc : processing server data function.

int32_t (*cbFunc)(int32_t iSock, uint8_t *pSendBuf, uint8_t *pRecvBuf, int32_t iMaxBufSize, void *pObj)

-. iSock	: connected client socket.
-. pSendBuf	: buffer for sending.

-. pRecvBuf	: buffer for receving.
-. iMaxBufSize	: Max buffer size for receving / sending
-. pObj	: private handle.
-. pObj	: private handle.

Return Value

None.

4.3.1.7 SendCommand()

```
virtual void SendCommand(
        const char *pSock,
        uint8_t *pSendBuf,
        int32_t iSendSize,
        uint8_t *pRecvBuf,
        int32_t iRecvMaxSize
) = 0;
```

Description

Send IPC Command for single transaction. (single write and single read)

Parameter

-. pSock	: socket name of server.
-. pSendBuf	: buffer for sending.
-. iSendSize	: buffer size of sending.
-. pRecvBuf	: buffer for receving.
-. iRecvMaxBufSize	: max buffer size of receiving.

Return Value

If it is successful, return value is actual reading size.

```
virtual void SendCommand(
        const char *pSock,
        int32_t (*cbFunc)( int32_t, uint8_t*, uint8_t*, int32_t, void *),
        void *pObj
) = 0;
```

Description

Send IPC Command for multiple transaction. (multiple write and single read)

Parameter

-. pSock	: socket name of server.
-. cbFunc	: callback for multiple writing and reading.

int32_t (*cbFunc)(int32_t iSock, uint8_t *pSendBuf, uint8_t *pRecvBuf, int32_t iMaxBufSize, void *pObj)

-. iSock	: connected client scoekt.
-. pSendBuf	: buffer for sending.
-. pRecvBuf	: buffer for receving.
-. iMaxBufSize	: Max buffer size for receving / sending

-. pObjf	: private handle.
-. pObj	: private handle
Return Value	
If it is successful, return value is return value of callback.	

4.3.2 IPC Packet

IPC 통신을 위한 Packet 을 만들거나 읽기 위한 Library 로서, Packet 의 구조는 다음과 같다.

-. Key(4 Bytes)	: Packet 을 구분하기 위한 Key Value.
-. Payload Size(2Bytes)	: Payload Size. (2 Byte 값으로 최대 65535 개)
-. Payload Buffer	: Payload Buffer. (Big Endian)

KEY (4Bytes)	Payload Size (2Bytes)	Payload Buffer (n Bytes) : buffer[0] .. buffer[n]

4.3.2.1 NX_IpcMakePacket()

```
int32_t NX_IpcMakePacket(
         uint32_t iKey,
         void *pPayload,
         int32_t iPayloadSize,
         void *pOutBuf,
         int32_t iOutMaxSize
);
```

Description

Make packet for IPC.

Parameter

-. iKey	: key value.
-. pPayload	: buffer for payload.
-. iPayloadSize	: payload size.
-. pOutBuf	: output buffer for making packet.
-. iOutMaxSize	: max output buffer size.

Return Value

On Success, the number of byte is returned. Otherwise, -1 is returned.

4.3.2.2 NX_IpcParsePacket()

```
int32_t NX_IpcParsePacket (
         void *pInBuf,
         int32_t iInBufSize,
         uint32_t *iKey,
         void *ppPayload,
         uint32_t *iPayloadSize
);
```

Description

Parse packet for IPC.

Parameter

-. pInBuf	: buffer
-. iInBufSize	: buffer size.
-. iKey	: key value of received buffer.
-. ppPayload	: payload value for received buffer.
-. iPayloadSize	: payload size for received buffer.

Return Value

On Success, zero is returned. Otherwise, -1 is returned.

4.3.2.3 NX_IpcDumpPacket()

```
void NX_IpcDumpPacket (
        void *pInBuf,
        int32_t iInBufSize,
        const char *pFunc = NULL,
        int32_t iLien = 0
);
```

Description

Dump Packet for debugging.

Parameter

-. pInBuf	: buffer
-. iInBufSize	: buffer size.
-. pFunc	: function name by calling this function. (__FUNCTION__)
-. iLine	: line by calling this function. (__LINE__)

Return Value

None.

4.3.3 IPC Utils

Display Audio 에서 쓰기 쉽게 IPC 통신과 관련된 기능을 쓰기 쉽게 Wrapping 한 library 이다.

4.3.3.1 NX_GetProcessInfo()

```
int32_t NX_GetProcessInfo(
        NX_PROCESS_INFO *pInfo
);
```

Description

Make process information for IPC.
Parameter -. pInfo : Process Information for IPC.
Return Value On Success, zero is returned. Otherwise, -1 is returned.

4.3.3.2 NX_RequestCommand()

```
int32_t NX_RequestCommand(
        NX_IIpcManager* pIpcManager,
        NX_PROCESS_INFO* pInfo,
        int32_t iCommand
);
```

Description

Send command to Display Audio Manager.

Parameter

-. pIpcManager	: IPC manager handle by making GetIpcManagerHandle().
-. pInfo	: Process Information by making NX_GetProcessInfo()..
-. iCommand	: Reserved command for communication.

Return Value

On Success, received key value is returned. Otherwise, -1 is returned.

```
int32_t NX_RequestCommand(
        NX_IIpcManager* pIpcManager,
        void *pPayload,
        int32_t iPayloadSize,
        int32_t iCommand
);
```

Description

Send payload to Display Audio Manager.

Parameter

-. pIpcManager	: IPC manager handle by making GetIpcManagerHandle().
-. pPayload	: payload for sending to Display Audio Manager.
-. iPayloadSize	: payload size for sending.
-. iCommand	: Reserved command for communication.

Return Value

On Success, received key value is returned. Otherwise, -1 is returned.

4.3.3.3 NX_DumpHex()

```
int32_t NX_DumpHex(
        const void *pData,
        int32_t iSize
```

);
Description Dump data for debugging.
Parameter -. pData : buffer for dump. -. iSize : buffer size for dump.
Return Value None.

Chap 5. History

5.1 Known Issue.

Not Yet.

5.2 To Do List

Not Yet.