GOODCOM NFC SDK

Version No.	Date	Description
V2.1.1	2021/6/16	Add the API of Mifare DESFire card (See Chapter 7)

Table

1. SDK Environmental Configurations

Please refer to the file ***Configuration for GoodCom SDK.docx***

2. Introduction

SDK uses Handler to transfer the information, and do the corresponding operation according to the received messages. e.g. reading and writing card.

3. Example for Reading mifare card

```
import com.goodcom.gcsdk.GCAndroidNFC;
GCAndroidNFC mGcAndroidNFC = new GCAndroidNFC();//SDK NFC Construtor
    After instantiating an object, it will complete the initialization of the NFC module, including
powering up and starting the detection of the NFC card.
GcMsgHandler mGcMsgHandler = new GcMsgHandler(this.getMainLooper());
mGcAndroidNFC.RegisterNFCHandler(mGcMsgHandler);
Declare a Handler object and pass the Handler to GCAndroidNFC.
/////////////Example in the file Mifare_Read.java ////////////////////
public class GcMsgHandler extends Handler{
        public GcMsgHandler(Looper L)
        {
            super(L);
        }

         @Override
         public void handleMessage(Message msg)
         {
             switch(msg.what)
             {
                // The NFC module enters into the status of testing card.
                case   GCAndroidNFC.GC_MSG_NFC_WAIT:
                {
//                      Toast.makeText(Mifare_Read.this, "NFC wait!" ,   2000).show();
                }
                break;
                // The NFC card is detected and the message is sent with the serial number of
the card.
                case   GCAndroidNFC.GC_MSG_NFC_READY:
                {
```

```
//                              Toast.makeText(NFC_Write.this, "NFC ready!" ,    500).show();
                                char [] serial_no = (char[]) msg.obj;
                                String displayString = "Serial Number:";

                                displayString += String.valueOf(serial_no);
                                nfc_text.setText(displayString);
                                mGcAndroidNFC.GCDefaultAuth();
                            }
                            break;
                            /// The card password is successfully authenticated and the next read and write
operation can be performed. The current example is to read the Mifare card.
                            case    GCAndroidNFC.GC_MSG_NFC_AUTH:
                            {
//                              Toast.makeText(Mifare_Read.this, "NFC auth!" ,    500).show();
                                mGcAndroidNFC.GCReadMF1();// GC API Read NFC
                            }
                            break;
                            //Reading card successfully and return card data.
                            case    GCAndroidNFC.GC_MSG_READ_MF1:
                            {
                                char [] test = (char[]) msg.obj;
                                nfc_edit.setText(String.valueOf(test));
                            }
                            break;
                            default:
                                break;
                        }
                    }
            }
```

4. Introduction for Communication message

Name of Communication Message	Functions
GCAndroidNFC.GC_MSG_NFC_WAIT	NFC module enters into the status of detecting card.
GCAndroidNFC.GC_MSG_NFC_READY	The NFC card is detected and the message is sent with the serial number of the card.
GCAndroidNFC.GC_MSG_NFC_AUTH	Card password authentication is successful, and the next read and write operations can be performed.
GCAndroidNFC.GC_MSG_READ_MF1	Read Mifare card successfully, return

	card data
GCAndroidNFC.GC_MSG_WRITE_MF1	Write Mifare card successfully answered
GC_MSG_READ_MF0	Read the Ultralight card successfully, and return card data
GC_MSG_WRITE_MF0	Write an Ultralight card successfully answered
GC_MSG_SET_MF1_VALUE	Set value of mifare card
GC_MSG_GET_MF1_VALUE	Get value of mifare card
GC_MSG_MF1_VALUE_ADD	Add value of mifare card
GC_MSG_MF1_VALUE_REDUCE	Reduce value of mifare card
GC_MSG_SECTOR_FINISH	Write mifare card sector finish

5. The API of card control

```
//Read ultralight card

public void GCReadMF0();

//Write ultralight card

public void GCWriteMF02(char[] data);

//Read mifare card

public void GCReadMF1();


//Write Mifare card

public void GCWriteMF1(char[] data);

//Set the value of Block in mifare card,the block is default

public void GCSetValue(int nValue)

//Get the value of Block in mifare card,the block is default

public void GCGetValue()

//Add the value of Block in mifare card,the block is default
```

```
public void GCAddValue(int nValue)

//Reduce the value of Block in mifare card,the block is default

public void GCReduceValue (int nValue)

//The mifare card needs password to unlock before read/write.

//Use default password to auth.

public void GCDefaultAuth();

//Close the card reader.

public void free();

protected void onDestroy() {

        mGcAndroidNFC.free();

        super.onDestroy();

};
```

6. New API of mifare card

6.1 Use custom password to authenticate

```
/*Use default password to authenticate a sector,
the block is belong this sector*/
public void GCDefaultAuth(char block);
public void GCDefaultAuth(char auth_type, char block);
//Use custom password to authenticate
public void GCUserAuth(char auth_type, char block, char[] userPwd);
```

6.2. Custom write and read

```
/*
* char nStartBlock: Write from this block
* char nBlockNum: Number of blocks written. Value range 1 to 4.
* char[] data: Data length must be nBlockNum * 16
* Can only write several blocks in the same sector.
* Please pay attention to the scope of writing, do not write to the password area easily
*/
    public void GCWriteMF1(char nStartBlock, char nBlockNum,
char[] data);

/* Custom read mifrae1, can set start block and block numbers
* char nStartBlock: Read from this block
* char nBlockNum: Number of blocks to read. Value range 1 to 4.
*/
public void GCReadMF1(char nStartBlock, char nBlockNum);

/* 2018/10/29 new add, the JAR need to upgrade 1.0.8
* char nStartBlock: Write from this block
* char nBlockNum: Number of blocks written. Value range 1 to 3.
* char[] data: if bUseGcFormat = true, the max length of data is
12+(nBlockNum-1)*16, 1<= nBlockNum<=3
else the length of data must be 16 * nBlockNum
* boolean bUseGcFormat: Whether use GoodCom Custom Format to write NFC
*/
public void GCWriteMF1(char nStartBlock, char nBlockNum, char[] data, boolean
bUseGcFormat)
/*Set value of block*/
public void GCSetValue(int Block, int nValue)
/*Get value of block*/
public void GCGetValue(int Block)

/*Add value of block*/
public void GCAddValue(int Block, int nValue)
/*Reduce value of block*/
public void GCReduceValue(int Block, int nValue)
/*Reset the NFC reader, will re-detect the card*/
public void GcNfcReset()
/* 2018/12/13 new add, the JAR need to upgrade 1.0.9
* The current sector is write finish, can authenticate the next sector to write*/
public void GCFinishSector()
```

7. API of Mifare DESFire Card

7.1 MF3_AuthPsw

public void MF3_AuthPsw(**char**[] aid,**int** pswNo,**char**[] psw)		
Description		Authentication password
Parameters	aid	Select the AID that needs to be authenticated. Length must be 3. (ex:0x00,0x00,0x00)
	pswNo	Specify one of the 14 sets of passwords in the application (0-0x0D)
	psw	Length must be 16
Return		void

7.2 MF3_Create_Application

public void MF3_Create_Application(**char**[] aid,**int** set,**int** keyNum)		
Description		Create an application
Parameters	aid	Set Application ID. Length must be 3. (ex:0x00,0x00,0x01)
	set	Apply master key settings (Default: 0x0F)
	keyNum	Number Of Keys (0-0x0D)
Return		void

7.3 MF3_Create_AndWriteStdFile

public void MF3_Create_AndWriteStdFile(**int** fileNo, **char**[] data, **byte** type, **int** accessRig, **int** fileSize)		
Description		Create and write files
Parameters	fileNo	File Number (0x00~0x0F)
	data	Data to be written. Length of the data must be less than fileSize
	type	*DES_COMMIT_TYPE_NO* *// No encryption* *DES_COMMIT_TYPE_DES_MAC* *// Mac encryption* *DES_COMMIT_TYPE_DES* *// DES encryption*
	accessRig	Set access permissions (ex:0x0000)
	fileSize	Set the file size
Return		void

7.4 MF3_ReadFile

public void MF3_ReadFile(**int** fileNo,**int** offSet,**int** readLen)		
Description		Read File
Parameters	fileNo	File Number (0x00~0x0F)

	offSet	Start reading position
	readLen	Read length
Return		void

7.5 MF3_Format

public void MF3_Format()	
Description	Format the card First, pass the password authentication of the PICC. PICC (AID :0x00,0x00,0x00), pswNo=0.
Parameters	void
Return	void