

GemPCSC

||||| Android Integration Guide



www.gemalto.com

gemalto 
security to be free

Table of Contents

1. Introduction	3
1.1. Android USB host mode	3
1.2. CCID standard	3
1.3. GemPCSC Service: CCID smart card reader service	4
2. GemPCSC Service.....	4
2.1. Overview	4
2.2. Architecture	4
2.3. Connecting to the GemPCSC Service	5
2.4. GemPCSC_Reader data.....	6
2.5. GemPCSC_StatusInfo data	6
2.6. Limitation.....	6
3. GemPCSC Demo	7
4. Annex	11
4.1. Bind to the GemPCSC Service.....	11
4.2. Interface IGemPCSC.....	11
4.2.1. GemPCSC_Initialize	11
4.2.2. GemPCSC_Finalize	12
4.2.3. GemPCSC_GetLastError	12
4.2.4. GemPCSC_BeginTransaction	12
4.2.5. GemPCSC_EndTransaction.....	13
4.2.6. GemPCSC_ListReaders.....	13
4.2.7. GemPCSC_IsCardPresent	14
4.2.8. GemPCSC_Connect.....	14
4.2.9. GemPCSC_Disconnect	15
4.2.10. GemPCSC_IsConnected	15
4.2.11. GemPCSC_Transmit.....	16
4.2.12. GemPCSC_Status.....	16
4.2.13. License.....	17

1. Introduction

1.1. Android USB host mode

Android can support a variety of USB peripherals through USB host mode. In USB host mode, the Android-powered phone or tablet acts as the host. Examples of devices include digital cameras, keyboards, mice, and game controllers. USB devices that are designed for a wide range of applications and environments can still interact with Android applications that can correctly communicate with the device.

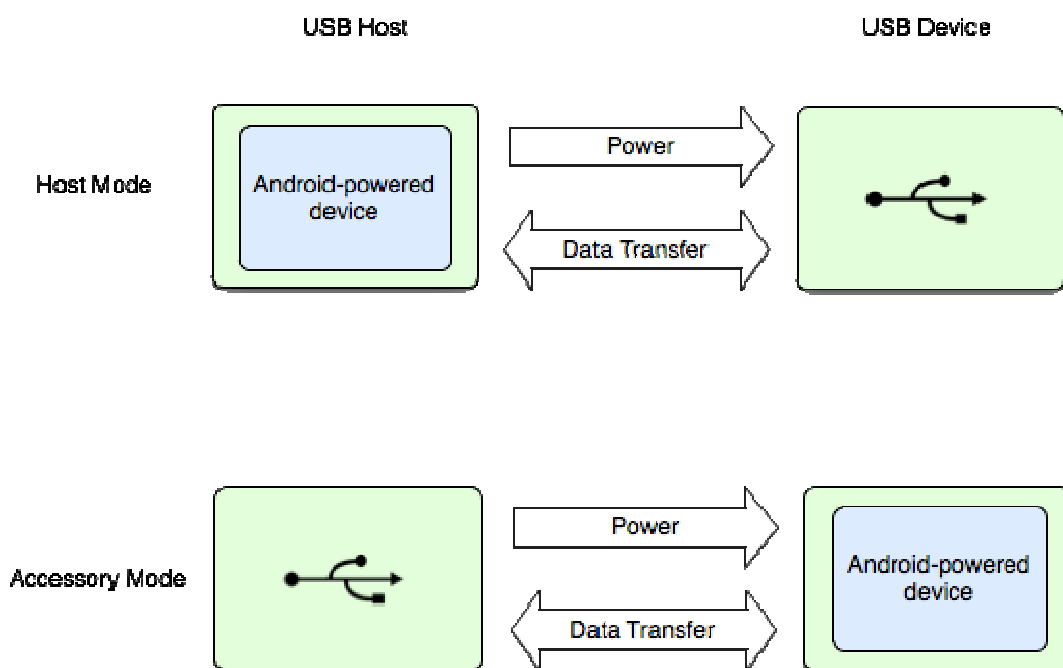


Figure 1: USB Host and Accessory Modes

Figure 1 shows the differences between the two modes. When the Android-powered device is in host mode, it acts as the USB host and powers the bus. When the Android-powered device is in USB accessory mode, the connected USB hardware (an Android USB accessory in this case) acts as the host and powers the bus.

1.2. CCID standard

Integrated circuit(s) card interface devices are Universal Serial Bus (USB) devices that interface with integrated circuit cards (with contacts, as specified in ISO/IEC 7816). These devices have been assigned the USB class number 0x0B.

The acronym CCID most commonly refers to the standardized protocol by which a USB host interacts with such a USB smart card reader.

You can find the CCID specification here:

http://www.usb.org/developers/devclass_docs/DWG_Smart-Card_CCID_Rev110.pdf

1.3. GemPCSC Service: CCID smart card reader service

The GemPCSC Service is the CCID implementation based on android USB APIs to enable the smart card application on android phone and tablet. Basically it should

- Support any CCID smart card reader.
- Support ISO7816 smart card with T=1 or T=0.
- Support Exchange levels: short APDU, TPDU and extended APDU.

2. GemPCSC Service

2.1. Overview

GemPCSC is an Android Service which enables your Android application to control the CCID smart card reader and send APDU commands to T=0 or T=1 ISO7816 smart card. It exposes a set of Java interfaces, covering CCID specification. The entire code is pure Java based on the standard Android 3.1 (or later) USB host API.

GemPCSC Service is an Android Package (`GemPCSC_Service.apk`) that must be installed on device in order to communicate with USB CCID smart card readers.

2.2. Architecture

It is possible to communicate with the GemPCSC Service over IPC (Inter Process Communication) by using the AIDL (Android Interface Definition Language) API which gives full access to all the functions supported by the GemPCSC Service. The AIDL file is the description of the methods implemented by the service.

The GemPCSC Service implements an Android service which is in charge of processing commands sent by a third party app that connects to it.

It can be regarded in the same way as using a DLL on Windows OS, the GemPCSC Service implements the methods, and the third party app can call them with the definition included in the AIDL interface description file.

Note that GemPCSC Service, like other application objects, runs in the main thread of its hosting process. As the transmissions with smart card are synchronous GemPCSC Service has no more thread to execute communication request. That means that if you need to do some intensive requests or blocking operations, application should spawn its own thread in which to do that work (for example you can use `AsyncTask` to interact with UI of your application: <http://developer.android.com/reference/android/os/AsyncTask.html>).

2.3. Connecting to the GemPCSC Service

Before being able to use methods defined in the AIDL file, the third party app must connect to the GemPCSC Service. This must be done by sending a bind request to the service by implementing a ServiceConnection callback that will receive the service instance (for more information: <http://developer.android.com/guide/components/bound-services.html#Binding>)

Third party app should bind to GemPCSC Service using this intent action:

```
final Intent intentService = new Intent("com.gemalto.gempcsc.service.action");
```

You can see a sample in Annex.

GemPCSC Service is an Android Service which exposes some methods to communicate with smart card via CCID USB smart card reader.

The following AIDL file `IGemPCSC.aidl`, `GemPCSC_StatusInfo.aidl` and `GemPCSC_Reader.aidl` must be included in the third party app to use the GemPCSC Service. This is the interface of all supported methods as describe into `IGemPCSC.aidl`:

```
interface IGemPCSC
{
    // Service management
    int GemPCSC_Initialize();
    void GemPCSC_Finalize(int sessionId);

    // Error management
    int GemPCSC_GetLastError();

    // GemPCSC transaction management
    boolean GemPCSC_BeginTransaction(int sessionId);
    boolean GemPCSC_EndTransaction(int sessionId);

    // GemPCSC service methods
    GemPCSC_Reader[] GemPCSC_ListReaders(int sessionId);
    boolean GemPCSC_IsCardPresent(int sessionId);
    byte[] GemPCSC_Connect(int sessionId, in GemPCSC_Reader reader, byte
protocol);
    boolean GemPCSC_Disconnect(int sessionId);
    boolean GemPCSC_IsConnected(int sessionId);
    byte[] GemPCSC_Transmit(int sessionId, in byte[] apduCommand);
    byte[] GemPCSC_Control(int sessionId, int controlCode, in
byte[] escapeCommand);
    GemPCSC_StatusInfo GemPCSC_Status(int sessionId);
}
```

Methods exposed from GemPCSC Service are described in Annex.

2.4. GemPCSC_Reader data

GemPCSC_Reader object is a representation of CCID USB smart card reader connected to the device. It exposes basics information like: product ID, product name, reader name, vendor ID, vendor name and slot count.

You don't need to create any instance of GemPCSC_Reader object, just call `GemPCSC_ListReaders()` to get list of connected reader and use them to communicate with the GemPCSC Service.

2.5. GemPCSC_StatusInfo data

You get GemPCSC_StatusInfo object when you call method `GemPCSC_Status()`. This object contains information about card and reader like ATR and slot status.

2.6. Limitation

Implementation of the GemPCSC Service wants to associate one application with one session ID, so one reader / one slot. So, you cannot communicate with two or more different readers inside the same application.

3. GemPCSC Demo

A complete Android application project (GemPCSC_Demo project) is provided with the GemPCSC Service in order to demonstrate use of the GemPCSC interface.

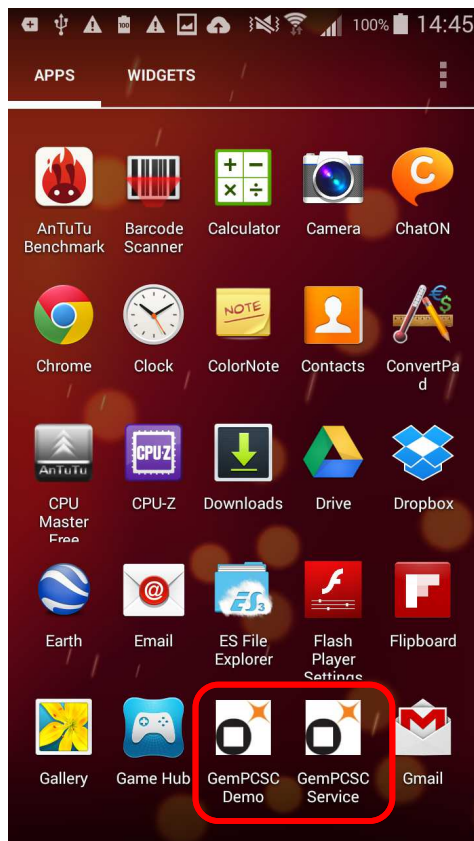


Figure 2: GemPCSC Service and GemPCSC Demo installed on device.

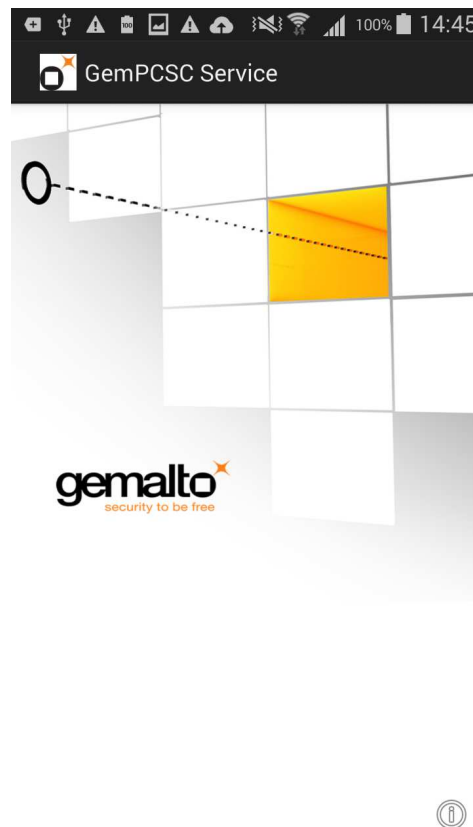


Figure 3: GemPCSC Service activity.

Figure 2 shows installed application: GemPCSC Demo and GemPCSC Service, which is an Android application containing a single activity (Figure 3) and GemPCSC Service needed to communicate with USB CCID smart card reader.

To display the license, just touch the information button at bottom right (Figure 3).

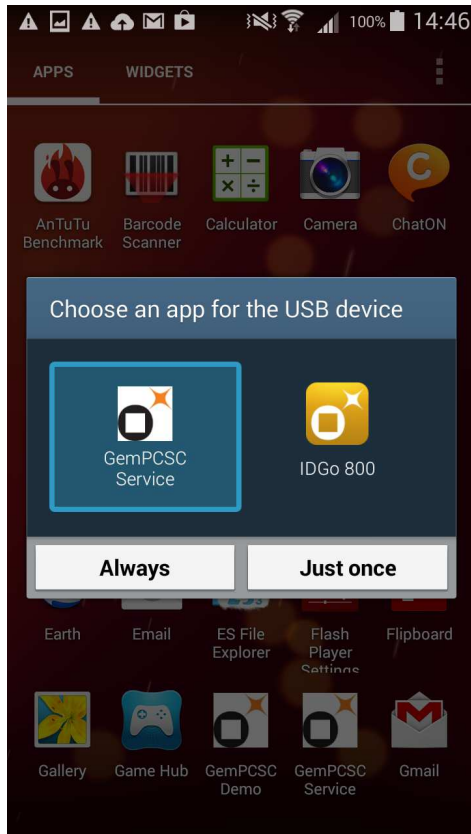


Figure 4: Select application to use with CCID USB reader which is just plugged.

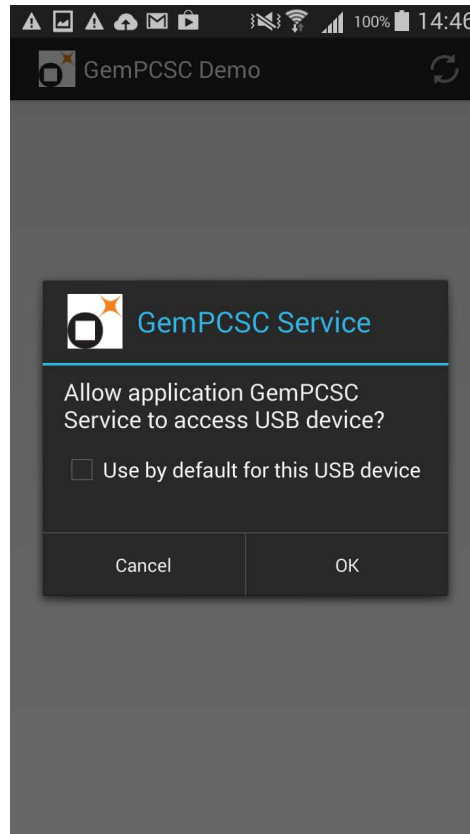


Figure 5: Permission required to use GemPCSC Service if not already selected as shown in Figure 4.

Figure 4 shows popup which appear when a CCID USB reader is plugged to the device. If you select GemPCSC Service, you give permission to communicate with smart card reader. If not you will get permission popup shown in figure 5 when you will try to communicate with CCID USB smart card reader.

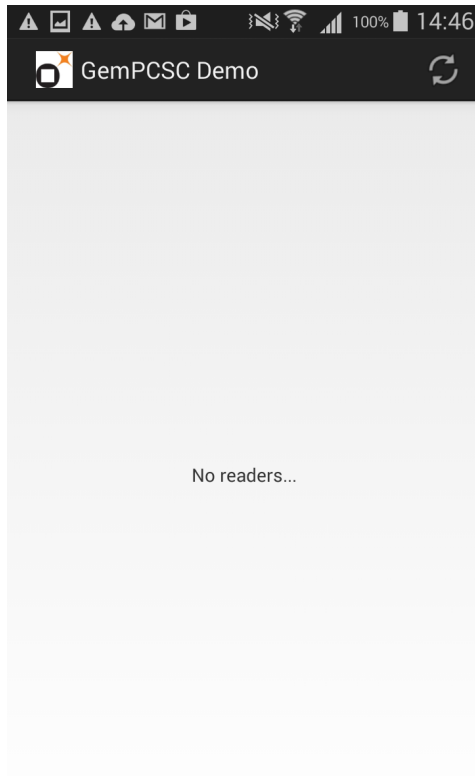


Figure 6: *GemPCSC_ListReaders()*
return an empty reader list.

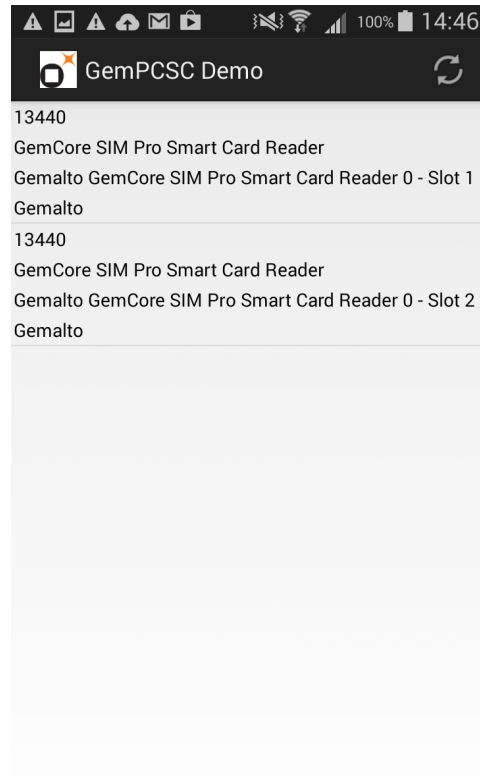


Figure 7: *GemPCSC_ListReaders()*
return a *GemPCSC_Reader* object for
each smart card reader detected.

Figure 6 & 7 show different call's results to *GemPCSC_ListReaders()*. Figure 7 details that you get a *GemPCSC_Reader* object for each smart card reader slot, so multi slot smart card reader could be used with GemPCSC Service.

On Figure 7, click on selected item if you want to transmit APDU on the selected slot (Figure 8 & 9).

In the sample below, you can see how to call *GemPCSC_ListReaders()*.

```
IGemPCSC gemPCSC_Service;
// bind to service
try {
    final int sessionID = gemPCSC_Service.GemPCSC_Initialize();
    final GemPCSC_Reader [] usbReaders =
gemPCSC_Service.GemPCSC_ListReaders(sessionID);
    if (gemPCSC_Service.GemPCSC_GetLastError() == GemErrors.GE_OK) {
        // update reader list
    }
} catch (RemoteException e) {}
```

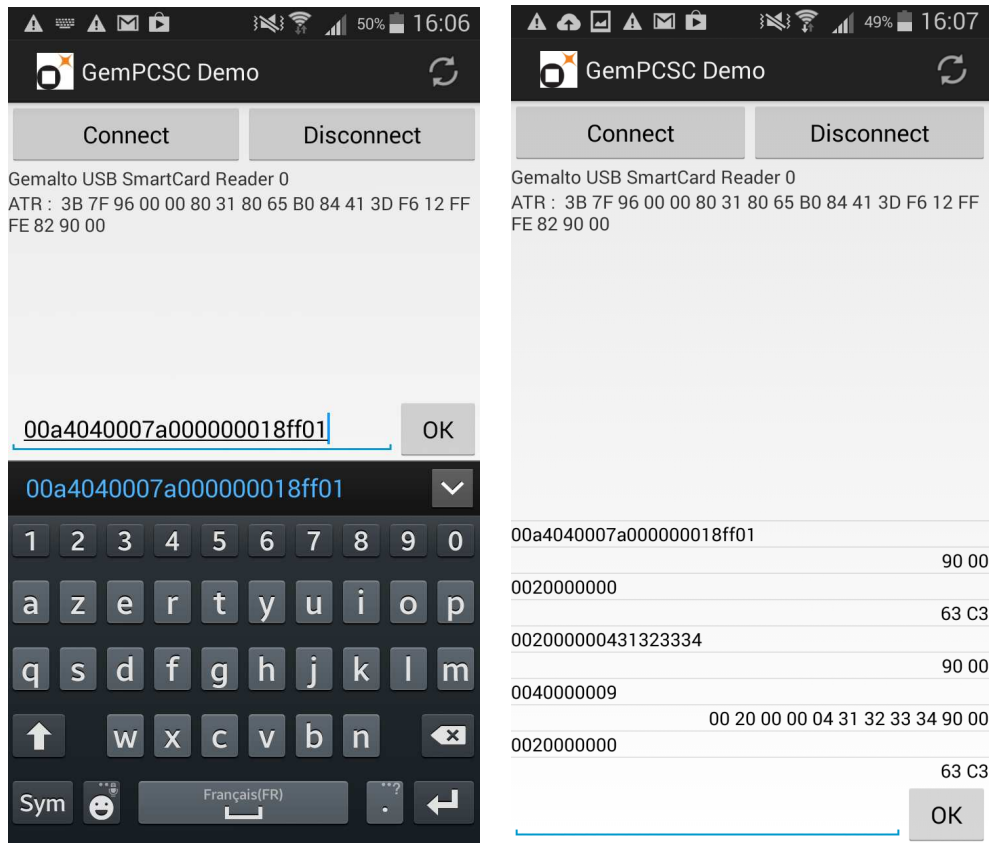


Figure 8 & 9: Communication with smart card using `GemPCSC_Transmit()`.

When you connect to the smart card reader using `GemPCSC_Connect()` with selected `GemPCSC_Reader` object, you get the ATR from smart card and then you can exchange APDU with `GemPCSC_Transmit()`.

You can see below the associated sample code which shows how to connect to smart card reader (`GemPCSC_Connect()`) and how to communicate with smart card (`GemPCSC_Transmit()`).

```
IGemPCSC gemPCSC_Service;
GemPCSC_Reader usbReader;
EditText editTextMessage;
// bind to service
try {
    final int sessionID = gemPCSC_Service.GemPCSC_Initialize();
    final byte[] ATR = gemPCSC_Service.GemPCSC_Connect(sessionID, usbReader,
(byte) 0x00);
    if (GemErrors.GE_OK == gemPCSC_Service.GemPCSC_GetLastError()) {
        // display ATR...
        final String msg = editTextMessage.getText().toString();
        byte[] apdu = gemPCSC_Service.GemPCSC_Transmit(sessionID,
stringToBytes(msg));
        if (GemErrors.GE_OK == gemPCSC_Service.GemPCSC_GetLastError()) {
            // manage response...
        }
    }
} catch (RemoteException e) {}
```

4. Annex

4.1. Bind to the GemPCSC Service

First step to communicate with smart card is to bind application to the GemPCSC Service.

Mandatory point: use this action `"com.gemalto.gempcsc.service.action"` as intent's parameter.

Here is a sample to show how to bind an activity to the GemPCSC Service:

```
final ServiceConnection serviceConnection = new ServiceConnection() {

    @Override
    public void onServiceDisconnected(ComponentName name) {
        // on service disconnected
    }

    @Override
    public void onServiceConnected(ComponentName name, IBinder service) {
        // on service connected
        IGemPCSC gemPCSC_Service = IGemPCSC.Stub.asInterface(service);
        // ready to use IGemPCSC interface
    }
};

final Intent intentService = new Intent("com.gemalto.gempcsc.service.action");
this.bindService(intentService, serviceConnection, Context.BIND_AUTO_CREATE);
```

4.2. Interface IGemPCSC

When you call a method from Interface IGemPCSC, a result value is updated according to execution status. You can access this value by calling `GemPCSC_GetLastError()`.

Error values are defined into `GemErrors.java`.

It's recommended to check result after each call like this:

```
if (GemPCSC_GetLastError() == GemErrors.GE_OK) // OK
```

For each method described below, you can find which value could be used as result.

4.2.1. GemPCSC_Initialize

```
int GemPCSC_Initialize();
```

Initializes the GemPCSC Service and opens a session with the service.

Returns

The session ID of the opened session or -1 if there is no session available.

Result can take different values:

`GemErrors.GE_OK` if OK

`GemErrors.GE_NO_SESSION_AVAILABLE` if no session available.

4.2.2. GemPCSC_Finalize

```
void GemPCSC_Finalize(int sessionId);
```

Finalize a session with the GemPCSC Service previously established with `GemPCSC_Initialize()` method.

Parameters

`sessionId` : The session ID previously returned by `GemPCSC_Initialize()`.

Result can take different values:

`GemErrors.GE_OK` if OK

`GemErrors.GE_INVALID_SESSION_ID` if session is invalid

`GemErrors.GE_TRANSACTION_LOCKED` if transaction is already locked by a previous call to `GemPCSC_BeginTransaction()` with a different `sessionId`.

4.2.3. GemPCSC_GetLastError

```
int GemPCSC_GetLastError();
```

Returns

The execution status code of the last executed method.

As described into `GemErrors.java`, the following execution status code can be returned :

```
public class GemErrors
{
    public static final int GE_OK = 0;
    public static final int GE_TRANSACTION_LOCKED = 200;
    public static final int GE_INVALID_SESSION_ID = 301;
    public static final int GE_NO_SESSION_AVAILABLE = 302;
    public static final int GE_CARD_NOT_PRESENT = 500;
    public static final int GE_READER_NOT_CONNECTED = 501;
    public static final int GE_UNKNOWN = 1000;
}
```

4.2.4. GemPCSC_BeginTransaction

```
boolean GemPCSC_BeginTransaction(int sessionId);
```

Begins a transaction with the specified session. Other sessions will be not able to access to the card until the session is unlocked by calling `GemPCSC_EndTransaction()`.

Parameters

`sessionId` : the session ID previously returned by `GemPCSC_Initialize()`.

Returns

`true` if transaction can be established, else `false`.

Result can take different values:

GemErrors.GE_OK if OK

GemErrors.GE_INVALID_SESSION_ID if session is invalid

GemErrors.GE_TRANSACTION_LOCKED if transaction is already locked by a previous call to GemPCSC_BeginTransaction() with a different sessionId.

4.2.5. GemPCSC_EndTransaction

```
boolean GemPCSC_EndTransaction(int sessionId);
```

Ends a transaction with the specified session. Other sessions will be able to access the card after this call.

Parameters

sessionId: The session ID previously return by GemPCSC_Initialize().

Returns

true if transaction can be ended, else false.

Result can take different values:

GemErrors.GE_OK if OK

GemErrors.GE_INVALID_SESSION_ID if session is invalid

GemErrors.GE_TRANSACTION_LOCKED if transaction is already locked by a previous call to GemPCSC_BeginTransaction() with a different sessionId.

4.2.6. GemPCSC_ListReaders

```
GemPCSC_Reader[] GemPCSC_ListReaders(int sessionId);
```

Returns all CCID USB Smart Card Readers plugged to the device.

Parameters

sessionId: The session ID previously returned by GemPCSC_Initialize().

Returns

An array of GemPCSC_Reader object that represent each CCID USB reader available from GemPCSC Service.

Result can take different values:

GemErrors.GE_OK if OK

GemErrors.GE_INVALID_SESSION_ID if session is invalid

GemErrors.GE_TRANSACTION_LOCKED if transaction is already locked by a previous call to GemPCSC_BeginTransaction() with a different sessionId.

4.2.7. GemPCSC_IsCardPresent

```
boolean GemPCSC_IsCardPresent(int sessionId);
```

Checks if a smart card is present into the CCID USB Smart Card Reader.

Parameters

`sessionId` : The session ID previously returned by `GemPCSC_Initialize()`.

Returns

true if the Smart Card is present into the CCID USB Smart Card Reader, else false.

Result can take different values:

`GemErrors.GE_OK` if OK

`GemErrors.GE_INVALID_SESSION_ID` if session is invalid

`GemErrors.GE_TRANSACTION_LOCKED` if transaction is already locked by a previous call to `GemPCSC_BeginTransaction()` with a different `sessionId`

`GemErrors.GE_READER_NOT_CONNECTED` if reader is not connected

`GemErrors.GE_UNKNOWN` if Service cannot retrieve value from smart card reader.

4.2.8. GemPCSC_Connect

```
byte[] GemPCSC_Connect(int sessionId, in GemPCSC_Reader reader);
```

Establishes a connection with the Smart Card present into the given CCID USB Smart Card Reader. If no card detected in the specified reader, an error is returned.

Parameters

`sessionId` : The session ID previously returned by `GemPCSC_Initialize()`.

`reader` : CCID USB Smart Card Reader to connect with.

Returns

Byte array that represent the ATR returned by the smart Card.

Result can take different values:

`GemErrors.GE_OK` if OK

`GemErrors.GE_INVALID_SESSION_ID` if session is invalid

`GemErrors.GE_TRANSACTION_LOCKED` if transaction is already locked by a previous call to `GemPCSC_BeginTransaction()` with a different `sessionId`

`GemErrors.GE_READER_NOT_CONNECTED` if reader is not connected

`GemErrors.GE_CARD_NOT_PRESENT` if card is not present.

4.2.9. GemPCSC_Disconnect

```
boolean GemPCSC_Disconnect(int sessionId);
```

Disconnects from the smart Card connected with the selected session.

Parameters

`sessionId`: The session ID previously returned by `GemPCSC_Initialize()`.

Returns

true if the smart card is correctly disconnected, else false.

Result can take different values:

`GemErrors.GE_OK` if OK

`GemErrors.GE_INVALID_SESSION_ID` if session is invalid

`GemErrors.GE_TRANSACTION_LOCKED` if transaction is already locked by a previous call to `GemPCSC_BeginTransaction()` with a different `sessionId`.

4.2.10. GemPCSC_IsConnected

```
boolean GemPCSC_IsConnected(int sessionId);
```

Indicates if the selected session is already connected to a smart card.

Parameters

`sessionId`: The session ID previously returned by `GemPCSC_Initialize()`.

Returns

true if the session is connected to a smart card, else false.

Result can take different values:

`GemErrors.GE_OK` if OK

`GemErrors.GE_INVALID_SESSION_ID` if session is invalid

`GemErrors.GE_TRANSACTION_LOCKED` if transaction is already locked by a previous call to `GemPCSC_BeginTransaction()` with a different `sessionId`.

4.2.11. GemPCSC_Transmit

```
byte[] GemPCSC_Transmit(int sessionId, in byte[] apduCommand);
```

Transmits an APDU command to the smart card connected to the indicated session.

Parameters

`sessionId`: The session ID previously returned by `GemPCSC_Initialize()`.

`apduCommand`: The APDU command to transmit to the smart card.

Returns

The smart card response to the APDU command. The response includes the SW two bytes status code.

Result can take different values:

`GemErrors.GE_OK` if OK

`GemErrors.GE_INVALID_SESSION_ID` if session is invalid

`GemErrors.GE_TRANSACTION_LOCKED` if transaction is already locked by a previous call to

`GemPCSC_BeginTransaction()` with a different `sessionId`

`GemErrors.GE_CARD_NOT_PRESENT` if card is not present

`GemErrors.GE_UNKNOWN` if service cannot retrieve value from smart card reader.

4.2.12. GemPCSC_Status

```
GemPCSC_StatusInfo GemPCSC_Status(int sessionId);
```

Provides the current status of a smart card in the reader.

Parameters

`sessionId`: The session ID previously returned by `GemPCSC_Initialize()`.

Returns

The `GemPCSC_StatusInfo` object which contains information about smart card in the reader as described in section 2.5. `GemPCSC_StatusInfo` data.

Result can take different values:

`GemErrors.GE_OK` if OK

`GemErrors.GE_INVALID_SESSION_ID` if session is invalid

`GemErrors.GE_TRANSACTION_LOCKED` if transaction is already locked by a previous call to

`GemPCSC_BeginTransaction()` with a different `sessionId`

`GemErrors.GE_READER_NOT_CONNECTED` if reader is not connected

`GemErrors.GE_UNKNOWN` if service cannot retrieve value from smart card reader.

4.2.13. License

END USER LICENSE AGREEMENT

PLEASE READ CAREFULLY: THE USE OF THE SOFTWARE IS SUBJECT TO THE TERMS AND CONDITIONS THAT FOLLOW (“AGREEMENT”). BY DOWNLOADING, INSTALLING, COPYING, ACCESSING, OR USING THE SOFTWARE, OR BY CHOOSING THE “I ACCEPT” OPTION LOCATED ON OR ADJACENT TO THE SCREEN WHERE THIS AGREEMENT MAY BE DISPLAYED, YOU AGREE TO THE TERMS OF THIS AGREEMENT, ANY APPLICABLE WARRANTY STATEMENT AND THE TERMS AND CONDITIONS CONTAINED IN THE “ANCILLARY SOFTWARE” (as defined below). IF YOU ARE ACCEPTING THESE TERMS ON BEHALF OF ANOTHER PERSON OR A COMPANY OR OTHER LEGAL ENTITY, YOU REPRESENT AND WARRANT THAT YOU HAVE FULL AUTHORITY TO BIND THAT PERSON, COMPANY, OR LEGAL ENTITY TO THESE TERMS. IF YOU DO NOT AGREE TO THESE TERMS, DO NOT DOWNLOAD, INSTALL, COPY, ACCESS, OR USE THE SOFTWARE.

QUANTITY OF DEVICES:

1. GENERAL TERMS

- a. *You* and *Your* refer either to an individual person or to a single legal entity.
- b. *Gemalto* means Gemalto S.A. a company organized under the law of France with its principal place of business located at 6, Rue de la Verrerie, 92190 Meudon (France).
- c. *Software* means that certain USB Driver and related updates and upgrades You may be separately authorized to receive, licensed materials, user documentation, user manuals, and operating procedures. “Ancillary Software” means all or any portion of Software provided under public, open source, or third party license terms.
- e. *Specification* means technical information about Software products published in Gemalto product manuals, user documentation, and technical data sheets in effect on the date Gemalto delivers the Software to You.

2. LICENSE TERMS AND RESTRICTIONS

- a. Subject to the terms and conditions of this Agreement, Gemalto grants You a non-exclusive, non-transferable, royalty-free, fully paid-up worldwide license to Use (as defined below) in object code form one copy of the Software on a device running on an Android operating system at a time for Your internal business purposes (i.e., means Use of the Software for your customary personal or internal business purposes and does not mean any distribution whatsoever of the Software or any component. “Internal Business purposes” does not include any Use of the Software by persons that are not Your authorized employees, or Your authorized agents. All such employees and agents shall be notified by you as to the terms and conditions of the Agreement and shall agree to be bound by it before they can have

Use of the Software) unless otherwise indicated above. “Use” means to install, store, load, execute and display the Software in accordance with the Specifications. Your Use of the Software is subject to these license terms and to the other restrictions made available to You with the Software, including license terms, warranty statements, Specifications, and “readme” or other informational files included in the Software itself. Such restrictions are hereby incorporated in this Agreement by reference.

- b. Time limit. Once Gemalto authorizes You access to the Software, the first twenty (20) calendar days from the date You install the Software You benefit of a free trial version to evaluate the fitness of the Software for operation in your conditions (the “Free Trial Period”). This right automatically terminates at the end of the Free Trial Period, which is programmed into the Software. Following this Free Trial Period, if you wish to continue to use the Software You must pay, unless otherwise agree with Gemalto, the license fee charged by Gemalto. Once you have paid the applicable licensee fee Gemalto will provide you with a license key to activate the Software in order to continue using it during its term. The term during which You can use the Software after the expiration of the Trial Period to be agreed with Gemalto may vary from three (3) calendar months up to a maximum of twenty four (24) calendar months. The right to use the Software automatically terminates at the end of the agreed upon term, a time limit with automatic expiry is programmed into the Software.
- c. This Agreement confers no title or ownership and is not a sale of any rights in the Software. Third-party suppliers are intended beneficiaries under this Agreement and independently may protect their rights in the Software in the event of any infringement. All rights not expressly granted to You are reserved solely to Gemalto or its suppliers. Nothing herein should be construed as granting You, by implication, estoppel or otherwise, a license relating to Software other than as expressly stated above in this section 2.
- d. Unless otherwise expressly permitted by Gemalto, You (a) may only make copies or adaptations of the Software for archival purposes or when copying or adaptation is an essential step in the authorized Use of the Software on a backup device, provided that copies and adaptations are used in no other manner and provided further that the Use on the backup device is discontinued when the original or replacement device becomes operable, and (b) may not copy the Software onto or otherwise Use or make it available on, to, or through any public or external distributed network.
- e. To Use Software identified as an update or upgrade, You must first be licensed for the original Software identified by Gemalto as eligible for the update or upgrade. If the update or upgrade is intended to substantially replace the original Software, after updating or upgrading, You may no longer Use the original Software that formed the basis for Your update or upgrade eligibility unless otherwise provided

by Gemalto in writing. Nothing in this Agreement grants You any right to purchase or receive Software updates, upgrades, or support, and Gemalto is under no obligation to make such support available to you. Updates, upgrades, enhancements, or other Support may only be available under separate Gemalto support agreements. Gemalto reserves the right to require additional licenses and fees for Software upgrades or other enhancements, or for Use of the Software on upgraded devices.

- f. You must reproduce all copyright notices that appear in or on the Software (including documentation) on all permitted copies or adaptations. Copies of documentation are limited to Your internal business purposes (hereabove defined).
- g. Software is not specifically designed, manufactured, or intended for use as parts, components, or assemblies for the planning, construction, maintenance, or direct operation of a nuclear facility. You are solely liable if Software is Used for these applications and will indemnify and hold Gemalto harmless from all loss, damage, expense, or liability in connection with such Use.
- h. You will not modify, reverse engineer, disassemble, decrypt, decompile, or make derivative works of the Software. Where You have other rights mandated under statute, You will provide Gemalto with reasonably detailed information regarding any intended modifications, reverse engineering, disassembly, decryption, or decompilation and the purposes therefore.
- i. Extending the Use of Software to any legal entity other than You as a function of providing services, (i.e.; making the Software available through a commercial timesharing or service bureau) must be authorized in writing by Gemalto prior to such Use and may require additional licenses terms. You may not distribute, resell, or sublicense the Software.
- j. Notwithstanding anything in this Agreement to the contrary, all or any portion of the Software which constitutes Ancillary Software is licensed to You subject to the terms and conditions of the Software license agreement accompanying such Ancillary Software, whether in the form of a separate agreement, shrink wrap license or electronic license terms accepted at time of download. Use of the Ancillary Software by You shall be governed entirely by the terms and conditions of such license and, with respect to Gemalto, by the limitations and disclaimers of sections 3 and 5 hereof. Gemalto has identified any Ancillary Software by either noting the Ancillary Software provider's ownership within each Ancillary Software program file and/or by providing information in the **annex 1** below, or "ancillary.txt" or "readme" file that is provided as part of the installation of the Software. By accepting the terms and conditions of this Agreement, You are also accepting the terms and conditions of each Ancillary Software license. Your right to use the Software (or portion thereof) subject to such Ancillary Software license are defined and restricted in such Ancillary Software license. If the Software includes Ancillary Software licensed under the GNU General Public License

and/or under the GNU Lesser General Public License ("GPL Software"), a complete machine-readable copy of the source code ("GPL Source Code") is either: (i) included with the Software that is delivered to You; or (ii) upon your written request, Gemalto will provide to You, for a fee covering the cost of distribution, a complete machine-readable copy of the GPL Source Code, by mail.

3. WARRANTY

DISCLAIMER OF WARRANTIES:

The Software is provided to You at minimal charge, consequently TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, GEMALTO AND ITS SUPPLIERS PROVIDE THE SOFTWARE "AS IS" AND WITH ALL FAULTS, AND HEREBY DISCLAIM ALL INDEMNITIES, WARRANTIES AND CONDITIONS, EITHER EXPRESS, IMPLIED, WHETHER BY STATUE, COMMON LAW, CUSTOM OR OTHERWISE, INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF TITLE AND NON-INFRINGEMENT, ANY IMPLIED WARRANTIES, DUTIES OR CONDITIONS OF MERCHANTABILITY, OF FITNESS FOR A PARTICULAR PURPOSE, AND OF LACK OF VIRUSES. Gemalto does not warrant that the operation of Software will be uninterrupted or error free or that the Software will meet Your requirements. Some jurisdictions do not allow exclusion of implied warranties or limitations on the duration of implied warranties, so the above disclaimer may not apply to You in its entirety.

4. LIMITATION OF LIABILITY AND REMEDIES

The Software is provided to You at minimal charge, consequently notwithstanding any damages that You might incur, and except for damages for bodily injury (including death) and for the amounts in section 4.a, the entire aggregate liability of Gemalto and any of its suppliers relating to the Software or this Agreement, and Your exclusive remedy for all of the foregoing, shall be limited to one hundred (100) Euro. TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, IN NO EVENT SHALL GEMALTO OR ITS SUPPLIERS BE LIABLE FOR ANY SPECIAL, INCIDENTAL, INDIRECT, OR CONSEQUENTIAL DAMAGES WHATSOEVER INCLUDING, BUT NOT LIMITED TO, DAMAGES FOR LOSS OF PROFITS OR REVENUES, BUSINESS INTERRUPTION, DOWNTIME COSTS, FAILURE TO REALIZE EXPECTED SAVINGS, LOSS, DISCLOSURE, UNAVAILABILITY OF OR DAMAGE TO DATA, SOFTWARE RESTORATION, OR LOSS OF PRIVACY ARISING OUT OF OR IN ANY WAY RELATED TO THE USE OF OR INABILITY TO USE THE SOFTWARE, OR OTHERWISE IN CONNECTION WITH ANY PROVISION OF THIS AGREEMENT, EVEN IF GEMALTO OR ANY SUPPLIER HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES AND EVEN IF THE REMEDY FAILS OF ITS ESSENTIAL PURPOSE. Some jurisdictions do not allow the exclusion or limitation of incidental or consequential damages, so the above limitation or exclusion may not apply to you.

5. TERMINATION

This Agreement is effective unless terminated or rejected. Notwithstanding the foregoing, this Agreement will also terminate upon conditions set forth elsewhere in this Agreement or if You fail to comply with any term or condition hereof. Immediately upon termination You will destroy the Software and all copies of the Software or return them to Gemalto. You may retain one copy of the Software subsequent to termination solely for archival purposes only. At Gemalto's request, You will certify in writing to Gemalto that You have complied with these requirements. Sections 3, 4, 5 and 6 of this Agreement will survive termination of this Agreement.

6. GENERAL

- a. You may not assign, sublicense, delegate or otherwise transfer ("Assign") all or any part of this Agreement without prior written consent from Gemalto, and compliance with Gemalto's Software license transfer policies and any applicable third party license terms. Any such attempted Assignment will be null and void. Where an authorized Assignment occurs in accordance with this section, Your rights under this Agreement will terminate, and You will immediately deliver the Software and all copies to the Assignee. The Assignee must agree in writing to the terms of this Agreement, and the transferee thereafter will be considered "You" for purposes of this Agreement. You may transfer firmware only upon transfer of the associated hardware.
- b. If the Software is licensed for use in the performance of a U.S. Government prime contract or subcontract, You agree that, consistent with FAR 12.211 and 12.212, commercial computer Software, computer Software documentation and technical data for commercial items are licensed under Gemalto's standard commercial license.
- c. To the extent You export, re-export, or import Software, technology, or technical data licensed or provided hereunder, You assume sole responsibility for complying with applicable laws and regulations and for obtaining required export and import authorizations. Gemalto may suspend performance if You are in violation of any applicable laws or regulations.
- d. You agree that Gemalto may audit Your compliance with this Agreement. Any such audit would be at Gemalto's expense, require reasonable notice, and would be performed during normal business hours.
- e. This Agreement is governed by the laws of France excluding rules as to choice and conflict of law. You and Gemalto agree that the United Nations Convention on Contracts for the International Sale of Goods will not apply to this Agreement.
- f. Subject to the other terms and conditions of this Agreement, this Agreement is the entire agreement between Gemalto and You regarding Your Use of the Software, and supersedes and replaces any previous communications, representations, or agreements, or Your additional or inconsistent terms, whether oral or written. In the event any provision of this Agreement is held invalid or unenforceable the remainder of the Agreement will remain enforceable and unaffected thereby.

- g. Gemalto's failure to exercise or delay in exercising any of its rights under this Agreement will not constitute or be deemed a waiver or forfeiture of those rights.

ANNEX 1

Third Party Software

The following Third Party Software are included in the Licensed Materials. Unless otherwise provided, the Third Party Software identified are licensed under the licenses indicated below. The following acknowledgements are made, as variously required:

Code from OpenCard Framework (Package opencard.opt.terminal.protocol)

Copyright 1997 - 1999 IBM Corporation


Redistribution and use in source (source code) and binary (object code) forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributed source code must retain the above copyright notice, this list of conditions and the disclaimer below.
2. Redistributed object code must reproduce the above copyright notice, this list of conditions and the disclaimer below in the documentation and/or other materials provided with the distribution.
3. The name of IBM may not be used to endorse or promote products derived from this software or in any other form without specific prior written permission from IBM.
4. Redistribution of any modified code must be labeled "Code derived from the original OpenCard Framework".

THIS SOFTWARE IS PROVIDED BY IBM "AS IS" FREE OF CHARGE. IBM SHALL NOT BE LIABLE FOR INFRINGEMENTS OF THIRD PARTIES RIGHTS BASED ON THIS SOFTWARE. ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IBM DOES NOT WARRANT THAT THE FUNCTIONS CONTAINED IN THIS SOFTWARE WILL MEET THE USER'S REQUIREMENTS OR THAT THE OPERATION OF IT WILL BE UNINTERRUPTED OR ERROR-FREE. IN NO EVENT, UNLESS REQUIRED BY APPLICABLE LAW, SHALL IBM BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE. ALSO, IBM IS UNDER NO OBLIGATION TO MAINTAIN, CORRECT, UPDATE, CHANGE, MODIFY, OR OTHERWISE SUPPORT THIS SOFTWARE.

Code from CCID free software driver

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.



This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA



APPENDIX 1

Licensed Materials

GemPCSCService.apk. Version 1.5.

GemPCSCDemo.apk. Version 1.5.

||||| The world leader in digital security

www.gemalto.com

gemalto 
security to be free