

SmartCard Framework v1 開発者ガイド

SDK/J Authentication Package Version:1.0



重要

Copyright © 2008-2010 Ricoh Co., Ltd.

ご注意

1. 本書の内容に関しては、将来予告無しに変更することがあります。
2. 本書の一部または全部を無断で複写、複製、改変、引用、転載、配布することはできません。
3. 本書および本書の対象となるサンプルコードについて、当社は、何らの保証もいたしません。
本書および本書の対象となるサンプルコードを使用したことにより生じるお客様の損害、逸失利益、または第三者からのいかなる請求につきましても、当社は一切その責任を負いかねますので、予めご了承下さい。

4. 商標について

Ethernetは、富士ゼロックス株式会社の登録商標です。

PostScript、Acrobatは、アドビシステムズ社の各国での登録商標または商標です。

Windowsは、米国Microsoft Corporationの米国及びその他の国における登録商標です。

UNIXは、X/Openカンパニーリミテッドがライセンスしている米国ならびに他の国における登録商標です。

Red Hatは、Red Hat, Inc. の米国およびその他の国における商標または登録商標です。

Linuxは、Linus Torvalds氏の米国およびその他の国における商標または登録商標です。

Java、JVM (CVM) 、CDCは、すべてSUNの米国およびその他の国における商標または登録商標です。

Eclipseは、eclipse.orgの米国およびその他の国における商標または登録商標です。

OSGi(TM)はThe Open Services Gateway Initiativeの米国およびその他の国における商標または登録商標です。

Apacheは、The Apache Software Foundationの米国およびその他の国における商標または登録商標です。

その他の製品名、名称は、各社の商標または登録商標です。

目次

1. はじめに	2
1.1. 対象読者	2
1.2. 動作環境	2
1.3. 制限事項	2
2. アーキテクチャ概要.....	3
2.1. カードマネージャ (CardManagerクラス)	3
2.2. カードサービス (CardServiceクラス)	4
2.3. カードサービスレジストリ (CardServiceRegistryクラス)	4
2.4. カードサービスバンドル (RegisterBundleクラス)	4
3. アプリケーションの開発.....	5
3.1. アプリケーションの作成	5
3.2. カードサービスの作成	7
3.3. カードサービスの登録	9
4. アプリケーションのインストール.....	11
4.1. DALP ファイルの設定	11
4.2. インストール	11
5. 付録	12
5.1. エミュレータによる SmartCard Framework アプリケーションの動作確認方法.....	12
変更履歴	14

1. はじめに

本書は、SmartCard Framework v1（以下、SCF）の使用方法について説明します。SCF は、Device SDK Type-J（以下、SDK/J）で、スマートカードを利用するためのソフトウェアフレームワークです。

1.1. 対象読者

本書は、スマートカードを利用する SDK/J アプリケーションの開発者を対象に記述されております。本書を読み進めるにあたり、以下の知識が必要になります。

- SDK/J アプリケーションを開発するための十分な知識
- スマートカードに関する基礎知識

1.2. 動作環境

SCF を使用するには、以下の環境が必要となります。

- PC/SC daemon が有効に設定されている

（PC/SC daemon の設定方法は、「SDK/J Authentication Package 設定ガイド」を参照ください。）

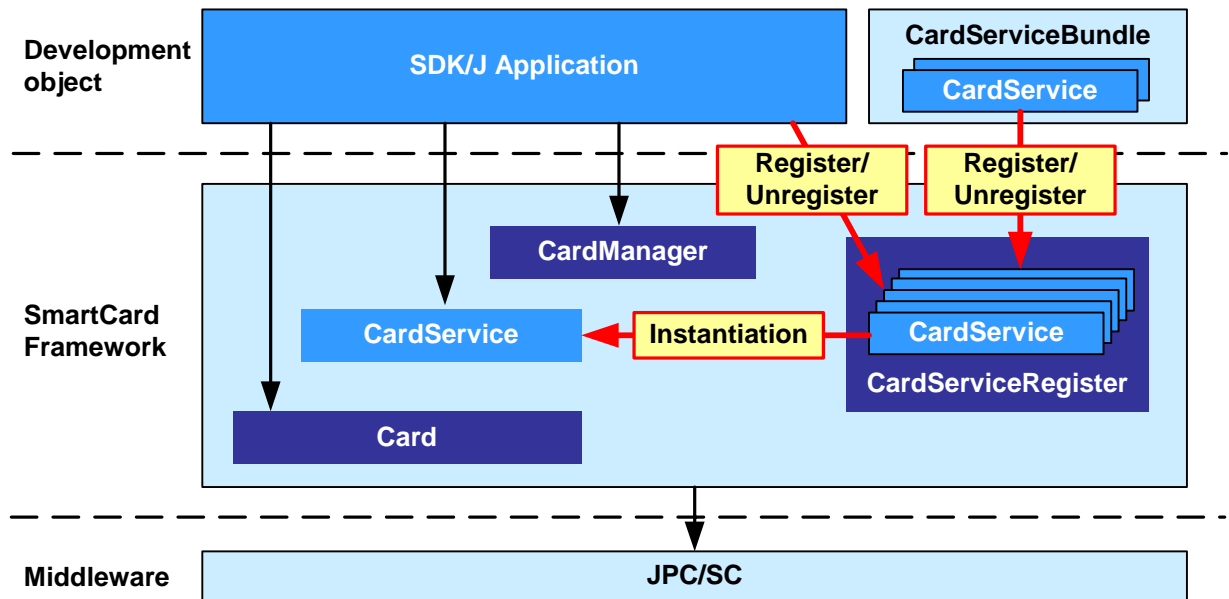
1.3. 制限事項

SCF には、以下の制限事項があります。

- 同一実行環境上で、SCF と OpenCard Framework を同時に使用した場合の動作は、保証されません
- 同時に複数のカードリーダーおよびその他の USB デバイスを使用した場合の動作は、保証されません

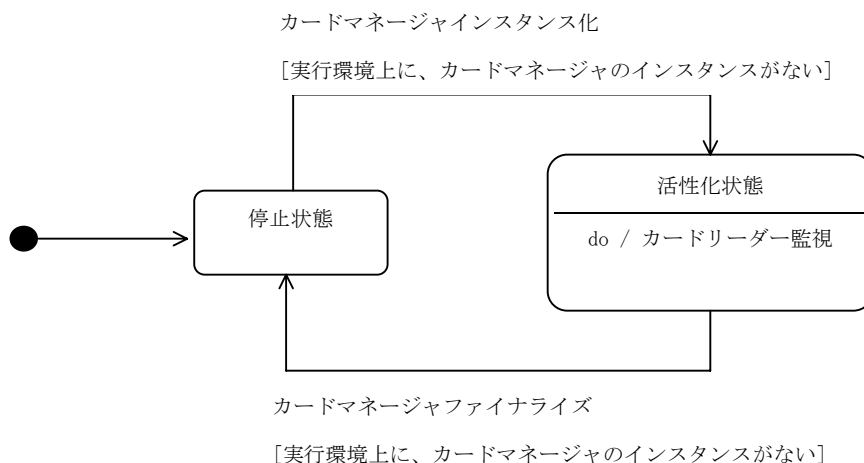
2. アーキテクチャ概要

SCF の構成要素およびそれに関連するソフトウェアの配置を簡略化した概念図を、下図に示します。SCF では、アプリケーションロジックとカードの仕様に依存するロジックを分けて実装することが可能です。これは、アプリケーション開発者にカードの詳細な仕様を公開出来ない場合や、運用性を考慮した場合、重要になります。本節では、SCF の構成要素と、それに対応する Java のクラスに関して説明します。



2.1. カードマネージャ（CardManagerクラス）

アプリケーションに対するフロントエンドとなります。SCF は、実行環境上でひとつ目のカードマネージャがインスタンス化される際、デーモンスレッドを生成し、最後のカードマネージャがファイナライズされる際、デーモンスレッドを破棄します。デーモンスレッドは、機器に接続されているカードリーダーおよびその状態を監視します。



2.2. カードサービス (CardServiceクラス)

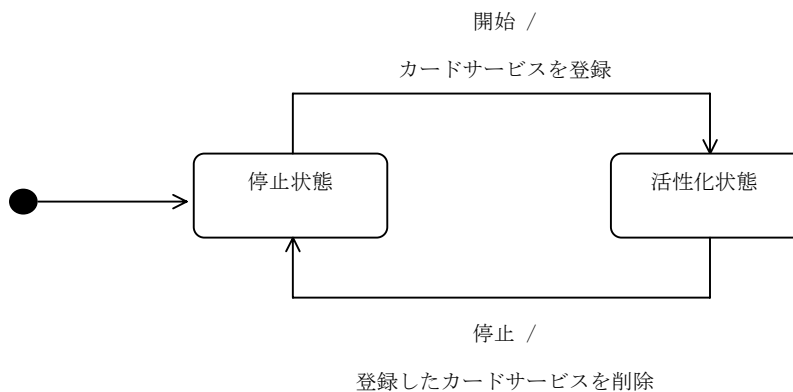
カードの仕様を抽象化します。通常、カードの詳細な仕様を知っているカードベンダやカード発行者によって実装されます。カードサービスを利用するには、そのカードサービスがカードサービスレジストリに登録されている必要があります。カードサービスのレジストリへの登録/削除は、通常、アプリケーションまたはカードサービスバンドルで実装されます。

2.3. カードサービスレジストリ (CardServiceRegistryクラス)

利用可能なカードサービスが格納された領域で、実行環境に唯一のインスタンスが存在します。SCF は、カードサービスを要求されると、カードサービスレジストリに登録されているカードサービスの中から、適切なものを識別し、そのインスタンスを返します。

2.4. カードサービスバンドル (RegisterBundleクラス)

カードサービスの配布媒体としての役割を果たします。カードサービス作成者は、使用するカードサービスをカードサービスレジストリに登録/削除するカードサービスバンドルを実装して、ユーザ（アプリケーション開発者も含む）に提供します。カードサービスは、カードサービスバンドルが、活性化状態の間だけ、利用可能となります。



3. アプリケーションの開発

ここでは、簡単なサンプルコードを用いて、SCF を用いたスマートカードアプリケーションの実装方法を説明します。SCF アプリケーションの開発には、**scard.jar** 内のクラスを使用します。

なお、SDK/J アプリケーションの作成に関する内容や、厳密な例外処理については省略します。

3.1. アプリケーションの作成

3.1.1. スマートカードの制御

スマートカードを制御するには、Card クラスのインスタンスを取得する必要があります。Card クラスのインスタンスを取得する方法として、ここでは、代表的なふたつの方法について説明します。

ひとつ目は、CardManager クラスの waitForCard メソッドを利用する方法です。このメソッドは、カードリーダーにカードが挿入されるか、引数で指定したタイムアウト時間を超過するまで待機します。

waitForCard サンプルコード

```

・
/** カードマネージャをインスタンス化する */
CardManager cm = new CardManager();

/** waitForCard メソッドを呼び出す */
Card card = cm.waitForCard(CardManager.INFINITY);
・

```

ふたつ目は、CardEvent クラスを利用する方法です。スマートカードが挿入された時の処理と、取り出された時の処理を記述した CardEventListener の実装を作成し、CardManager クラスのインスタンスに登録します。Card クラスのインスタンスは、CardEvent クラスの getSlot メソッドで取得した Slot オブジェクトの getCard メソッドで取得することができます。

CardEvent サンプルコード

```

・
/* カードマネージャをインスタンス化する */
CardManager cm = new CardManager();

Card card = null;
/* カードマネージャにリスナに登録する */
cm.addListener(new CardEventListener() {

    public void inserted (CardEvent event) {
        /** カード認識時の処理を実装 */
        card = event.getSlot().getCard();
    }

    public void removed (CardEvent event) {
        /** カード除去時の処理を実装 */
        card = null;
    }

});
・

```

Card クラスのインスタンスを取得したら、Card クラスが提供するメソッドを利用して、スマートカードと通信することができます。Card インスタンスの使用例を以下に示します。詳細は、Javadoc を参照してください。

Card インスタンス サンプルコード

```

・
・
/* Card.Info オブジェクトでカードに関する情報を取得する */
Slot slot    = card.getInfo().getSlot();           // スロットを取得
int protocol = card.getInfo().getProtocol();       // プロトコルを取得
byte[] atr    = card.getInfo().getAtr();           // ATR を取得

/* Card.IO オブジェクトでカードと通信する */
byte[] response = card.getIO().transmit(
    new byte[] {(byte)0x80, (byte)0x20, (byte)0x00, (byte)0x00}); // カードにコマンドを送信
・
・

```

3.1.2. カードサービスの利用

SCF では、アプリケーションでカードとの通信を全て実装することも出来ませんが、カードサービスを利用することにより、カードの詳細な仕様を意識せずにアプリケーションを作成することが可能となります（ただし、SCF のカードサービスレジストリに、カードサービスが登録されている必要があります。レジストリへのカードサービス登録方法は後述します）。カードサービスは、Card クラスの `getCardService` メソッドで取得することができます。`getCardService` メソッドのパラメータには、カードサービスのインタフェースを指定します。

FileAccessService サンプルコード

```

・
・
/** FileAccessServiceをCardインスタンスから取得する */
FileAccessService service =
    (FileAccessService) card.getCardService(FileAccessService.class);

/** FileAccessServiceの提供するサービスを利用する */
・
・

```

AppletService サンプルコード

```

・
・
/** AppletServiceをCardインスタンスから取得する */
AppletService service =
    (AppletService) card.getCardService(AppletService.class);

/** AppletServiceの提供するサービスを利用する */
・
・

```


3.2. カードサービスの作成

ここでは、カードサービスの実装方法を説明します。

カードサービスは、通常、以下の手順に従って作成します。

STEP-1 カードサービスのインタフェースを定義する。

カードサービスが提供するサービスを定義したインタフェースを作成します。または、SCF であらかじめ用意しているカードサービスインタフェースを使用する場合、このSTEPは省略します。

```
public interface AuthenticationService {

    public byte[] getUserName() throws CardServiceException;
    public byte[] getPassword() throws CardServiceException;

}
```

STEP-2 CardService クラスを継承したクラスを作成する。

全てのカードサービスは、CardService クラスを継承する必要があります。

```
/** CardService を継承する */
public MyAuthenticationService extends CardService {

}
```

STEP-2.1 initialize メソッドオーバーライドする。

initialize メソッドは、SCF によって、カードサービスがインスタンス化された後、最初に一度だけ呼び出されます。このメソッドのパラメータは、このカードサービスに関連付けるCardオブジェクトです。この Card オブジェクトが、カードサービスで、サポートしていないスマートカードの場合、UnsupportedCardException 例外をスローする必要があります。

```
public MyAuthenticationService extends CardService {

    public synchronized void initialize(Card card)
    throws UnsupportedCardException, CardServiceException {
        super.initialize(card);

        /**
         * cardがサポート外の場合、
         * UnsupportedCardExceptionをthrowする
         */
    }

}
```

STEP-2.2 カードサービスのインタフェースを実装する。

このカードサービスが、提供すべきサービスを定義したインタフェース（STEP-1 参照）を実装して完成です。カードサービスの利用者は、ここで実装したインタフェースをキーにして、カードサービスを取得することになります。

```
/** サービスを定義した interface を実装する */
public MyAuthenticationService extends CardService implements AuthenticationService {

    public synchronized void initialize(Card card)
    throws UnsupportedOperationException, CardServiceException {
        .
        .
        .
    }

    public byte[] getUser_name() throws CardServiceException {
        /**
         * getUser_nameメソッドの実装
         */
    }

    public byte[] getPassword() throws CardServiceException {
        /**
         * getPasswordメソッドの実装
         */
    }

}
```

3.3. カードサービスの登録

ここでは、カードサービスのカードサービスレジストリへの登録/削除の方法を説明します。

3.3.1. カードサービスバンドルからの登録

この方法では、カードサービスバンドルを作成し、バンドルからカードサービスのカードサービスレジストリへの登録/削除を行います。特に、カードサービスが複数のアプリケーションから使用される場合には、この方法を用いるようにしてください。

RegisterBundle クラスを継承したクラス（このクラスがカードサービスバンドルの BundleActivator になります）を作成し、getCardServices メソッドをオーバーライドします。

getCardServices メソッドでは、カードサービスレジストリに登録するカードサービスの Class を返します。カードサービスバンドルは、起動時に、getCardServices メソッドが返すカードサービスをカードサービスレジストリに登録し、停止時に、自身が登録したカードサービスをカードサービスレジストリから削除します。

```
public class MyRegister extends RegisterBundle {
    /**
     * getCardServicesメソッドをオーバーライドして、カードサービスを返す
     */
    protected Class[] getCardServices() throws Exception {
        return new Class[] {
            MyAuthenticationService.class
        };
    }
}
```

カードサービスバンドルのjarファイルは、上記 RegisterBundle 継承クラスと、各種カードサービスおよびカードサービスインタフェースから構成されます。

カードサービスバンドルを用いる場合、アプリケーションのインストール時にはカードサービスインタフェースのみをインストールします。カードサービスインタフェースは、4章で述べるように option-jar タグを使用してインストールしてください。

3.3.2. アプリケーションからの登録

この方法では、アプリケーションからカードサービスの登録を行います。カードサービスが1つのアプリケーションからしか使用されない場合には、この方法を用いても構いません。

アプリケーションでカードサービスを利用する前に、カードサービスレジストリにカードサービスを登録しておきます。

```
.
.
/** CardServiceRegister にカードサービスを登録 */
new CardServiceRegister().add(MyAuthenticationService.class);
.
.
```

カードサービスの利用が終了したら、カードサービスレジストリからカードサービスを削除します。

```
.  
.br/>/** CardServiceRegister からカードサービスを削除 */  
new CardServiceRegister().remove(MyAuthenticationService.class);  
.br/>.
```

この方法を用いる際には、アプリケーションのインストール時にはカードサービスの実装とカードサービスインタフェースをインストールします。

4. アプリケーションのインストール

SCF を利用した SDK/J アプリケーションのインストール方法は、以下になります。

4.1. DALP ファイルの設定

DALPファイルの書式については、SDK/J 開発者ガイドを参照してください。

SCF の jar ファイル (scard.jar) は、SDK/J の SD カードに同梱されているので、scard.jar を DALP ファイル内で指定する必要はありません。

以下に、SCF を利用するアプリケーションの DALP ファイル設定例を示します。

```

    ⋮
    <resources>
        <dsdk version = "2.0"/>
        <jar href="./123456789.jar" basepath="current" main="true"/>
        <option-jar href="./cardservice.jar" basepath="current" main="false"/> (1)
        <encode-file>123456789</encode-file>
    </resources>
    ⋮

```

(1) アプリケーションがカードサービスバンドルによって提供されるカードサービスインタフェースを使用する場合、そのインタフェースはoption-jar タグを使用してインストールする必要があります。

※ option-jar タグを使用した場合、インストール後に機器を再起動する必要があります。

4.2. インストール

通常の SDK/J アプリケーションと同様にインストールしてください。

詳細は、SDK/J のユーザーズガイドを参照ください。

5. 付録

5.1. エミュレータによる SmartCard Framework アプリケーションの動作確認方法

<<注意>>

本ドキュメントに記載された内容は、記載段階でのエミュレータの最新バージョンである Embedded Software Architecture Emulator 4.13d に関してのものになります。その他のバージョンのエミュレータを使用した場合の動作の保証はいたしかねますので、ご了承ください。

エミュレータへの SmartCard Framework v1.0（以下 SCF）アプリケーションのインストールを以下の手順で行うことで、エミュレータ上で SCF アプリケーションの動作確認を行うことができます。

STEP-1 SCF 利用環境を構築する

SCF を使用するための環境を構築します。

1. カードリーダーをインストールする

カードリーダーをインストールしていない場合、PC/SC 対応のカードリーダーをインストールしてください。

2. jpcsc.dll を cdc-dsdk4 に配置する

Java から PC/SC を利用するための JNI である jpcsc.dll を、<エミュレータインストールパス>%cdc-dsdk4 ディレクトリに配置してください。jpcsc.dll は、MUSCLE project の Web サイトよりダウンロードすることができます。

MUSCLE project : <http://www.linuxnet.com/>

3. jpcsc.jar, scard.jar をエミュレータのクラスパスに追加する

Java から PC/SC を利用するための API である jpcsc.jar および SCF の jar ファイルを、エミュレータのクラスパスに追加します。jpcsc.jar は、MUSCLE project の Web サイトよりダウンロードすることができます。以下の例（<エミュレータインストールパス>%cdc-dsdk4 に jpcsc.jar を配置した場合）を参考に、エミュレータの起動バッチファイルの-classpath を編集してください。

```

:

SET MYCLASSPATH=-classpath

.%resource%sdk%emulator%common%emulatorCommon.jar;.%mnt%sd2%sdk%common%jars%dsdk%dsdkCommon.jar;.%lib
%jh.jar;.%lib%xml2-2.3.0.jar;.%mnt%sd2%sdk%common%framework.jar;.%mnt%sd2%sdk%common%jars%smartcard%
scard.jar;.%jpcsc.jar

:

```

STEP-2 option-jar でインストールされる jar ファイルをエミュレータのクラスパスに追加する

DALP ファイルの option-jar タグはエミュレータではサポートされません。option-jar で指定される jar ファイルを、STEP-1.3 と同様の方法でエミュレータのクラスパスに追加します。

STEP-3 カードサービスバンドルの jar ファイルをエミュレータのクラスパスに追加する

ServerType アプリケーションはエミュレータではサポートされません。カードサービスバンドルからのカードサービスのインストールを行っている場合には、jar ファイルを、STEP-1.3 と同様の方法方法でエミュレータのクラスパスに追加します。

STEP-4 カードサービスの登録を行う

アプリケーション内で動的にカードサービス登録を行っていない場合(カードサービスバンドルを使用している場合も含む) は、SmartCard.properties ファイルからのカードサービスの登録※を行う必要があります。アプリケーションで使用しているカードサービスを SmartCard.properties ファイルに記述し、<エミュレータインストールパス>%cdc-dsdk4%cdc-toolkit%CDCTK10%lib ディレクトリに配置してください。

STEP-5 インストールする

エミュレータを、<エミュレータインストールパス>%startemulator-jvm.bat で起動します。
通常の SDK/J アプリケーションと同様の方法でインストールを行ってください。

※ SmartCard.properties ファイルからのカードサービスの登録

SCF では、初めてカードマネージャクラスがロードされる際、[*java.home*]/lib/ に SmartCard.properties というファイルが存在するかどうかのチェックが行われます。SmartCard.properties が上記パスに存在する場合、プロパティ名「jp.co.ricoh.dsdk.scard.service」に記述されたカードサービスが、カードサービスレジストリに登録されます。

以下に、SmartCard.properties ファイルの記述例を示します。カードサービスを「;」区切りで記述します。記述した SmartCard.properties は、[*java.home*]/lib/ に配置してください。

```
jp.co.ricoh.dsdk.scard.service=my.package.CardServiceA;my.package.CardServiceB
```

変更履歴

Ver. 1.0	SDK/J v2 オプションパッケージ版 v1.0 をベースに SDK/J v4 以降版を作成 動作環境 を更新 Implementation-Version: 1.0-1.0
Ver. 1.0.4	「5.1. エミュレータによる SmartCard Framework アプリケーションの動作 確認方法」の記載を更新。 Implementation-Version: 1.0-1.0