Manual de Integração Android Moderninha Smart

Sumário

Versionamento

Versão doc.	Data	Autor	Descrição	Versão PlugPag
1.0.0	28/01/2019	Carlos Vaccari	Criação do documento.	1.0
1.0.1	06/03/2019	Cássio Corrêa	Ajuste layout da capa	1.0
1.0.2	06/03/2019	Cássio Corrêa	Movido seção observações	1.0

Introdução

Este documento destina-se a integradores que utilizarão o terminal **Moderninha Smart** do PagSeguro como solução de pagamento integrada através da biblioteca **PlugPagService PagSeguro.**

A biblioteca **PlugPagService** permite aos integradores implementar aplicativos que podem ser utilizados na **Moderninha Smart.**

Observações Importantes

A biblioteca PlugPagService para o sistema operacional Android possui algumas restrições para seu uso.

- A biblioteca PlugPag possui suporte da API level 24 (7.0 Nougat) à 28 (9.0 Pie), devido a versão de android presente no terminal **Moderninha Smart**.
- Apenas uma única instância do PlugPag deve existir durante o uso do aplicativo. A existência de múltiplas instâncias pode fazer com que o comportamento seja indeterminado.
- As chamadas dos métodos da classe `PlugPag` devem ser feitas em uma `Thread` que execute em background pois podem demorar para finalizar a execução. Caso a execução seja feita na `Thread` principal (`UI Thread`), o aplicativo pode apresentar um ANR (Application Not Responding). Além disso, alguns métodos executam transações utilizando chamadas remotas pela internet, o que impossibilita suas chamadas na `Thread` principal.

Importando a biblioteca PlugPag

Para importar a biblioteca PlugPagService na sua aplicação nativa Android basta seguir os passos descritos abaixo:

1- Inserir no arquivo “build.gradle” do projeto a URL do repositório **Maven** do PlugPag:

```
allprojects {
 repositories {
  ...
  maven {
   url 'https://github.com/pagseguro/PlugPagServiceWrapper/raw/master'
  }
  ...
 }
}
```

2- Inserir as dependências no arquivo “build.gradle” da aplicação:

```
dependencies {
    ...
    implementation 'com.android.support:design:28.0.0'

    implementation
'br.com.uol.pagseguro.plugpagservice.wrapper:wrapper:1.0'
    ...
}
```

A versão da dependência `com.android.support:design` deve ser a mesma utilizada para as demais dependências `com.android.support`. A versão `28.0.0` é a mais recente no momento da edição desse documento.

AndroidManifest.xml

Permissões

Para integrar a biblioteca a biblioteca PlugPagService em aplicativos para Android é necessário adicionar a seguinte permissão ao *AndroidManifest.xml*.

```
<permission android:name="br.com.uol.pagseguro.permission.MANAGE_PAYMENTS"/>
```

Essa permissão permite à biblioteca realizar o bind ao PlugPagService, serviço embarcado da **Moderninha Smart**, que gerencia todas as transações de pagamento.

Intent-filter

Para que seu aplicativo possa ser escolhido como aplicativo default de pagamento e receber intents de inserção de cartão, é necessário adicionar o seguinte código em seu Manifest.xml dentro da sua activity principal.

```
<intent-filter>
      <action android:name="br.com.uol.pagseguro.PAYMENT"/>
      <category android:name="android.intent.category.DEFAULT"/>
</intent-filter>
```

Classes

A biblioteca PlugPagService é composta de um conjunto de classes.

A classe principal chama-se `PlugPag`, mas é necessário utilizar classes auxiliares para configurações e trocas de informações.

Segue abaixo uma lista com classes que compõem a biblioteca.

Classe	**Descrição**
PlugPag	Classe principal da biblioteca. Essa classe é responsável pelas transações.
PlugPagAbortResult	Resultado obtido ao solicitar um cancelamento de operação, enquanto a operação está em andamento.
PlugPagAppIdentification	Identificação do aplicativo.
PlugPagEventData	Dados de eventos gerados durante transações para atualização de eventos no aplicativo.
PlugPagPaymentData	Informações de um pagamento a ser realizado.
PlugPagTransactionResult	Resultado de uma transação.
PlugPagVoidData	Informações de um estorno a ser realizado.
PlugPagException	Tipo principal de exceções geradas pelo PlugPag.
PlugPagVoidTransactionException	Exceção lançada quando ocorrer um erro durante a configuração de um estorno.

Interfaces

As interfaces visam facilitar e padronizar algumas chamadas de métodos de forma assíncrona.

Interface	Descrição
PlugPagEventListener	Interface com método chamado quando um evento é enviado durante uma transação.
PlugPagAuthenticationListener	Interface com métodos chamados quando é gerado um retorno de uma solicitação de autenticação.

API

Abaixo segue a descrição da interface pública da biblioteca PlugPagService.

PlugPag

Essa é a classe principal da biblioteca.

É por meio dessa classe que é possível realizar transações nos terminais e leitores.

Constantes

int	RET_OK Código utilizado para indicar sucesso nas operações. Valor: 0
int	REQUEST_CODE_AUTHENTICATION Código utilizado para iniciar a Activity de autenticação. Valor: 46981
int	TYPE_CREDITO Tipo de pagamento: crédito. Valor: 1
int	TYPE_DEBITO Tipo de pagamento: débito Valor: 2
int	TYPE_VOUCHER Tipo de pagamento: voucher (vale refeição) Valor: 3

int	INSTALLMENT_TYPE_A_VISTA Forma de parcelamento: à vista Valor: 1
int	INSTALLMENT_TYPE_PARC_VENDEDOR Forma de parcelamento: parcelamento vendedor Valor: 2
int	ERROR_REQUIREMENTS_MISSING_PERMISSIONS Código de retorno para indicar erro de falta de permissões do aplicativo. Valor: -3000
int	ERROR_REQUIREMENTS_ROOT_PERMISSION Código de retorno para indicar que o aparelho possui permissões de root. Valor: -3001

Construtores

PlugPag(Context context, PlugPagAppIdentification appIdentification)

Cria uma instância do PlugPag utilizando `context` para acessar dados e recursos do dispositivo e identificando as transações com os dados do `appIdentification`.

Gera uma exceção se `context` ou `appIdentification` forem nulos.

Métodos

Tipo de retorno	Método e descrição
PlugPagAbortResult	abort() Solicita o cancelamento da operação atual. O cancelamento da transação **não** ocorre instantaneamente, pois depende do fluxo da transação. Só é possível solicitar o cancelamento de operação quando estiver fazendo transação com leitores. Retorna o resultado da solicitação de cancelamento.
int	checkRequirements(int deviceType) Verifica os requisitos para que os métodos da biblioteca possam ser executados para o `deviceType` desejado. O parâmetro `deviceType` pode ser `PlugPagDevice.TYPE_PINPAD` ou `PlugPagDevice.TYPE_TERMINAL`. Retorna `PlugPag.RET_OK` se os requisitos forem contemplados ou um código de erro caso algum requisito esteja ausente.
PlugPagTransactionResult	doPayment(PlugPagPaymentData paymentData) Efetua um pagamento. Retorna o resultado da transação.
PlugPagAppIdentification	getAppIdentification() Retorna a identificação do aparelho definido no construtor da classe.

String	getApplicationCode() Retorna o código da aplicação. Esse código é uma constante da biblioteca.
String	getLibVersion() Retorna a versão da biblioteca PlugPagService.
PlugPagTransactionResult	getLastApprovedTransaction() Consulta a última transação aprovada no terminal. Esse método não funciona para leitores. Se o terminal estiver ocupado, essa chamada aguarda sua liberação, bloqueando a `Thread`. Retorna o resultado da última transação aprovada. Retorna `null` se não houver transação disponível ou se for feita uma consulta em um leitor.
int	initBTConnection(PlugPagDevice deviceInformation) Configura a conexão bluetooth utilizando os dados de `deviceInformation`. Retorna `PlugPag.RET_OK` em caso de sucesso.
void	invalidateAuthentication() Invalida uma autenticação. Equivalente a realizar um logout.
boolean	isAuthenticated() Verifica se há um usuário autenticado. Retorna `true` se houver um usuário autenticado, `false` caso contrário.

void	requestAuthentication(PlugPagAuthenticationListener listener) Solicita autenticação. O resultado da autenticação é notificado ao listener que é passado no parâmetro `listener`.
void	setEventListener(PlugPagEventListener listener) Armazena a referência de uma instância de interface que receberá os eventos gerados durante as transações. Os eventos são gerados apenas para transações feitas utilizando um leitor.
int	setVersionName(String appName, String appVersion) Define o nome e a versão do aplicativo que está integrando com o PlugPag. `appName` pode ter no máximo 25 caracteres. `appVersion` pode ter no máximo 10 caracteres. Retorna um código de erro se um dos parâmetros for nulo ou vazio.
PlugPagTransactionResult	voidPayment(PlugPagVoidData voidData) Efetua um estorno de um pagamento em um leitor. Se `voidData` for nulo, é equivalente a fazer uma chamada `voidPayment()` para solicitar um estorno em um terminal. Retorna o resultado da transação.

PlugPagTransactionResult	voidPayment() Efetua um estorno de um pagamento em um terminal. Esse método só deve ser utilizado para fazer estornos em terminais. Chamar este método para realizar um estorno em um leitor resultará em um erro. Retorna o resultado da transação.

PlugPagAbortResult

Essa classe contém dados resultantes de uma solicitação de cancelamento de operação.

Construtores

PlugPagAbortResult(int result) Cria um container de dados resultantes de um cancelamento de operação com o código result.

Métodos

int	getResult() Retorna o código de resultado da solicitação de cancelamento de operação.

PlugPagAppIdentification

Essa classe representa a identificação de um aplicativo.

Construtores

PlugPagAppIdentification(String name, String version) Cria uma identificação do aplicativo, definindo seu nome e sua versão com os valores de `name` e `version`, respectivamente. Gera uma exceção se `name` ou `version` forem nulos ou vazios.

Métodos

String	getName() Retorna o nome do aplicativo.
String	getVersion() Retorna a versão do aplicativo.

PlugPagEventData

Essa classe representa um evento gerado pela biblioteca PlugPag para o aplicativo de integração.

Constantes

int	EVENT_CODE_DEFAULT Código padrão de evento. Utilizado quando nenhum evento foi enviado. Valor: -1
int	EVENT_CODE_WAITING_CARD Código de evento indicando que o leitor está aguardando o usuário inserir o cartão. Valor: 0
int	EVENT_CODE_INSERTED_CARD Código de evento indicando que o cartão foi inserido. Valor: 1
int	EVENT_CODE_PIN_REQUESTED Código de evento indicando que o leitor está aguardando o usuário digitar a senha. Valor: 2
int	EVENT_CODE_PIN_OK Código de evento indicando que a senha digitada foi validada com sucesso. Valor: 3

int	EVENT_CODE_SALE_END Código de evento indicando o fim da transação. Valor: 4
int	EVENT_CODE_AUTHORIZING Código de evento indicando que o pinpad está aguardando autorização da senha digitada para prosseguir com a transação. Valor: 5
int	EVENT_CODE_INSERTED_KEY Código de evento indicando que a senha foi digitada. Valor: 6
int	EVENT_CODE_WAITING_REMOVE_CARD Código de evento indicando que o leitor está aguardando o usuário remover o cartão. Valor: 7
int	EVENT_CODE_REMOVED_CARD Codigo de evento indicando que o cartão foi removido do leitor. Valor: 8

Construtores

PlugPagEventData(int eventCode) Cria um identificador de evento gerado pela biblioteca para o aplicativo de integração, com o código `eventCode`.

Métodos

int	getEventCode() Retorna o código do evento gerado.

Métodos estáticos

String	getDefaultMessage(int eventCode) Retorna uma mensagem padrão para um dado código de evento. Se o código de evento for inválido, retorna uma mensagem padrão.

PlugPagPaymentData

Essa classe representa os dados de um pagamento.

É nessa classe que são definidas informações de tipo de pagamento, valor a ser pago e parcelas, além e outras informações gerenciais.

Construtores

PlugPagPaymentData(int paymentType, int amount, int installmentType, int installments, String userReference)

PlugPagPaymentData(int paymentType, int amount, int installmentType, int installments, String userReference, Boolean printReceipt)

Cria um conjunto de informações necessários para iniciar um pagamento. O pagamento configurado será do tipo `paymentType`, com o valor `amount`, com parcelamento do tipo `installmentType`, com `installments` número de parcelas, identificado por `userReference`. O parametro `printReceipt` é opcional e indicará se deverá ser impresso os comprovantes da transação.
O `amount` definido é o valor em centavos a ser pago. Para um pagamento de R$ 1,50, o `amount` deverá ser de `150`.

Gera uma exceção se o `userReference` for nulo ou vazio.

Métodos

int	getAmount() Retorna o valor a ser pago.
int	getInstallments() Retorna o número de parcelas do pagamento.

int	getInstallmentType() Retorna o tipo de parcelamento. Valores: `PlugPag.INSTALLMENT_TYPE_A_VISTA` ou `PlugPag.INSTALLMENT_TYPE_PARC_VENDEDOR`
int	getType() Retorna o tipo de pagamento. Valores: `PlugPag.TYPE_CREDITO`, `PlugPag.TYPE_DEBITO` ou `PlugPag.TYPE_VOUCHER`.
String	getUserReference() Retorna o código de venda.

PlugPagPaymentData.Builder

Construtor de objetos `PlugPagPaymentData`.

Construtores

Builder() Cria um construtor de objetos `PlugPagPaymentData`.

Métodos

PlugPagPaymentData	build() Cria um `PlugPagPaymentData` com os dados armazenados no `Builder`.
Builder	setAmount(int amount) Define o valor a ser pago. Retorna a referência do próprio Builder para chamadas encadeadas. Se `amount` for igual a 1, o tipo de parcelamento é automaticamente definido para `PlugPag.INSTALLMENT_TYPE_A_VISTA`. Gera uma exceção se `amount` não for maior do que zero.
Builder	setInstallments(int installments) Retorna a quantidade de parcelas do pagamento. Retorna a referência do próprio Builder para chamadas encadeadas. Gera uma exceção se `installments` não for maior do que zero.

Builder	setInstallmentType(int installmentType) Define o tipo de parcelamento. Valores válidos para `installmentType` são `PlugPag.INSTALLMENT_TYPE_A_VISTA` e `PlugPag.INSTALLMENT_TYPE_PARC_VENDEDOR`. Retorna a referência do próprio Builder para chamadas encadeadas. Gera uma exceção se `installmentType` for inválido.
Builder	setType(int type) Define o tipo de pagamento. Valores válidos para `type` são `PlugPag.TYPE_CREDITO`, `PlugPag.TYPE_DEBITO` e `PlugPag.TYPE_VOUCHER`. Retorna a referência do próprio Builder para chamadas encadeadas. Gera uma exceção se `type` for inválido.
Builder	setUserReference(String userReference) Define o código de venda. Retorna a referência do próprio Builder para chamadas encadeadas. Gera uma exceção se `userReference` for nulo ou vazio.

PlugPagTransactionResult

Essa classe representa o resultado de uma transação.

Construtores

PlugPagTransactionResult(String message, String transactionCode, String transactionId, String date, String time, String hostNsu, String cardBrand, String bin, String holder, String userReference, String terminalSerialNumber, String amount, String availableBalance, String cardApplication, String cardCryptogram, String label, String holderName, String extendedHolderName) Cria um objeto para armazenar um conjunto de informações resultantes de uma transação.
PlugPagTransactionResult(String message, String errorCode, String transactionCode, String transactionId, String date, String time, String hostNsu, String cardBrand, String bin, String holder, String userReference, String terminalSerialNumber, String amount, String availableBalance, String cardApplication, String cardCryptogram, String label, String holderName, String extendedHolderName, int result) Cria um objeto para armazenar um conjunto de informações resultantes de uma transação, adicionando o código de resultado `result`.

Métodos

String	getAmount() Retorna o valor transacionado.
String	getAvailableBalance() Retorna o saldo da conta, caso o método de pagamento seja `PlugPag.TYPE_VOUCHER`.
String	getBin() Retorna os 4 (quatro) últimos dígitos do cartão utilizado.

String	getCardApplication() Retorna a aplicação do cartão.
String	getCardBrand() Retorna a bandeira do cartão utilizado.
String	getCardCryptogram() Retorna o criptograma do cartão.
String	getDate() Retorna a data da transação.
String	getErrorCode() Se um erro ocorreu durante a transação, retorna o código de erro.
String	getExtendedHolderName() Retorna o nome completo do titular do cartão utilizado.
String	getHolder() Retorna os 4 últimos dígitos do cartão utilizado.
String	getHolderName() Retorna o nome do titular do cartão utilizado.
String	getHostNsu() Retorna um identificador único do host (servidor).
String	getLabel() Retorna o label do cartão utilizado.
String	getMessage() Retorna uma mensagem do resultado da transação, definida pela biblioteca.

int	getResult() Retorna o código do resultado.
String	getTerminalSerialNumber() Retorna o número de série do terminal ou leitor utilizado para efetuar o pagamento.
String	getTime() Retorna o horário da transação.
String	getTransactionCode() Retorna o código da transação.
String	getTransactionId() Retorna o ID da transação.
String	getUserReference() Retorna o código de venda o pagamento efetuado.

PlugPagTransactionResult.Builder

Construtor de objetos `PlugPagTransactionResult`.

Construtores

Builder() Cria um construtor de objetos `PlugPagTransactionResult`.	

Métodos

PlugPagTransactionResult	build() Constrói uma instância da classe `PlugPagTransactionResult` utilizando os dados armazenados.
Builder	setAmount(String amount) Define o valor da transação. Retorna a referência do próprio Builder para chamadas encadeadas.
Builder	setAvailableBalance(String availableBalance) Define o saldo disponível. Retorna a referência do próprio Builder para chamadas encadeadas.
Builder	setBin(String bin) Define o BIN do cartão utilizado na transação. Retorna a referência do próprio Builder para chamadas encadeadas.

Builder	setCardApplication(String cardApplication) Define a aplicação do cartão utilizado na transação. Retorna a referência do próprio Builder para chamadas encadeadas.
Builder	setCardBrand(String cardBrand) Define a bandeira do cartão utilizado na transação. Retorna a referência do próprio Builder para chamadas encadeadas.
Builder	setCardCryptogram(String cardCryptogram) Define o criptograma do cartão utilizado na transação. Retorna a referência do próprio Builder para chamadas encadeadas.
Builder	setDate(String date) Define a data da transação. Retorna a referência do próprio Builder para chamadas encadeadas.
Builder	setExtendedHolderName(String extendedHolderName) Define o nome completo do titular do cartão utilizado na transação. Retorna a referência do próprio Builder para chamadas encadeadas.
Builder	setHolder(String holder) Define o nome do titular do cartão. Retorna a referência do próprio Builder para chamadas encadeadas.

Builder	setHolderName(String holderName) Define o nome do titular do cartão. Retorna a referência do próprio Builder para chamadas encadeadas.
Builder	setHostNsu(String hostNsu) Define o NSU do host que executou a transação. Retorna a referência do próprio Builder para chamadas encadeadas.
Builder	setLabe(String label) Define o label do cartão utilizado. Retorna a referência do próprio Builder para chamadas encadeadas.
Builder	setMessage(String message) Define a mensagem do resultado da transação que será construído. Retorna a referência do próprio Builder para chamadas encadeadas.
Builder	setTerminalSerialNumber(String terminalSerialNumber) Define o número de série do terminal ou leitor utilizado na transação. Retorna a referência do próprio Builder para chamadas encadeadas.

Builder	setTime(String time) Define o horário da transação. Retorna a referência do próprio Builder para chamadas encadeadas.
Builder	setTransactionCode(String transactionCode) Define o código da transação. Retorna a referência do próprio Builder para chamadas encadeadas.
Builder	setTransactionId(String transactionId) Define o ID da transação. Retorna a referência do próprio Builder para chamadas encadeadas.
Builder	setUserReference(String userReference) Define o código de venda da transação. Retorna a referência do próprio Builder para chamadas encadeadas.

PlugPagVoidData

Essa classe representa os dados de um estorno.

É nessa classe que são definidos dados necessários para solicitar o estorno de um pagamento.

Construtores

PlugPagVoidData(String transactionCode, String transactionId)

PlugPagVoidData(String transactionCode, String transactionId, Boolean printReceipt)

Cria um conjunto de informações para solicitar o estorno de um pagamento identificado pelo `transactioCode` e `transactionId` fornecidos. O parametro `printReceipt` é opcional e indicará se deverá ser impresso os comprovantes da transação.

Gera uma exceção se `transactionCode` ou `transactionId` forem nulos ou vazios.

Métodos

String	getTransactionCode() Retorna o código da transação que será estornada.
String	getTransactionId() Retorna o ID da transação que será estornada.

PlugPagVoidData.Builder

Construtor de objetos `PlugPagVoidData`.

Construtores

Builder() Cria um construtor de objetos `PlugPagVoidData`.

Métodos

PlugPagVoidData	build() Constrói uma instância da classe `PlugPagVoidData` utilizando os dados armazenados.
Builder	setTransactionCode(String transactionCode) Define o código da transação. Retorna a referência do próprio Builder para chamadas encadadas. Gera uma exceção se `transactionCode` for nulo ou vazio.
Builder	setTransactionId(String transactionid) Define o ID da transação. Retorna a referência do próprio Builder para chamadas encadadas. Gera uma exceção se `transactionId` for nulo ou vazio.

Exemplos

Seguem abaixo alguns exemplos de camadas dos métodos do PlugPag para realizar transações.

As formas de fazer as chamadas e de tratar os valores retornados vão depender da implementação do seu aplicativo.

Nos exemplos, quando houver diferença entre chamadas para terminais e leitores, ambas chamadas serão apresentadas. Caso as chamadas sejam iguais para leitores e terminais, apenas uma chamada será apresentada.

Pagamentos

Pagamento de R$250,00, no crédito, à vista:

```
public void startPayment(Context context) {
    // Define o terminal ou leitor que será utilizado para transação
    String deviceIdentification = "Nome ou MAC address do leitor/pinpad";
    PlugPagDevice device = new PlugPagDevice(deviceIdentification);

    // Define os dados do pagamento
    PlugPagPaymentData paymentData =
            new PlugPagPaymentData(
                    PlugPag.TYPE_CREDITO,
                    25000,
                    PlugPag.INSTALLMENT_TYPE_A_VISTA,
                    1,
                    "CODVENDA");

    // Cria a identificação do aplicativo
    PlugPagAppIdentification appIdentification =
            new PlugPagAppIdentification("MeuApp", "3.1.1");

    // Cria a referência do PlugPag
    PlugPag plugpag = new PlugPag(context, appIdentification);

    // Ativa terminal e faz o pagamento
    int initResult = plugpag.initializeAndActivatePinpad(new
    PlugPagActivationData("SEU_CODIGO_DE_ATIVAÇÃO")));

    if (initResult == PlugPag.RET_OK) {
        PlugPagTransactionResult result = plugpag.doPayment(paymentData);

        // Trata o resultado da transação
        ...
    }
}
```

Pagamento de R$300, no crédito, parcelado em 3 parcelas:

```
public void startPayment(Context context) {
    // Define os dados do pagamento
    PlugPagPaymentData paymentData =
            new PlugPagPaymentData(
                    PlugPag.TYPE_CREDITO,
                    30000,
                    PlugPag.INSTALLMENT_TYPE_PARC_VENDEDOR,
                    3,
                    "CODVENDA");

    // Cria a identificação do aplicativo
    PlugPagAppIdentification appIdentification =
            new PlugPagAppIdentification("MeuApp", "3.1.1");

    // Cria a referência do PlugPag
    PlugPag plugpag = new PlugPag(context, appIdentification);

    // Ativa terminal e faz o pagamento
    int initResult = plugpag.initializeAndActivatePinpad(new
    PlugPagActivationData("SEU_CODIGO_DE_ATIVAÇÃO")));

    if (initResult == PlugPag.RET_OK) {
        PlugPagTransactionResult result = plugpag.doPayment(paymentData);

        // Trata o resultado da transação
        ...
    }
}
```

Pagamento de R$150,00, no débito:

```
public void startPayment(Context context) {
    // Define o terminal ou leitor que será utilizado para transação
    String deviceIdentification = "Nome ou MAC address do leitor/pinpad";
    PlugPagDevice device = new PlugPagDevice(deviceIdentification);

    // Define os dados do pagamento
    PlugPagPaymentData paymentData =
            new PlugPagPaymentData(
                    PlugPag.TYPE_DEBITO,
                    15000,
                    PlugPag.INSTALLMENT_TYPE_A_VISTA,
                    1,
                    "CODVENDA");

    // Cria a identificação do aplicativo
    PlugPagAppIdentification appIdentification =
            new PlugPagAppIdentification("MeuApp", "3.1.1");

    // Cria a referência do PlugPag
    PlugPag plugpag = new PlugPag(context, appIdentification);

    // Ativa terminal e faz o pagamento
    int initResult = plugpag.initializeAndActivatePinpad(new
    PlugPagActivationData("SEU_CODIGO_DE_ATIVAÇÃO")));

    if (initResult == PlugPag.RET_OK) {
        PlugPagTransactionResult result = plugpag.doPayment(paymentData);

        // Trata o resultado da transação
        ...
    }
}
```

Pagamento de R$50, no voucher:

```
public void startPayment(Context context) {
    // Define o terminal ou leitor que será utilizado para transação
    String deviceIdentification = "Nome ou MAC address do leitor/pinpad";
    PlugPagDevice device = new PlugPagDevice(deviceIdentification);

    // Define os dados do pagamento
    PlugPagPaymentData paymentData =
            new PlugPagPaymentData(
                    PlugPag.TYPE_VOUCHER,
                    5000,
                    PlugPag.INSTALLMENT_TYPE_A_VISTA,
                    1,
                    "CODVENDA");

    // Cria a identificação do aplicativo
    PlugPagAppIdentification appIdentification =
            new PlugPagAppIdentification("MeuApp", "3.1.1");

    // Cria a referência do PlugPag
    PlugPag plugpag = new PlugPag(context, appIdentification);

    // Ativa terminal e faz o pagamento
    int initResult = plugpag.initializeAndActivatePinpad(new
    PlugPagActivationData("SEU_CODIGO_DE_ATIVAÇÃO")));

    if (initResult == PlugPag.RET_OK) {
        PlugPagTransactionResult result = plugpag.doPayment(paymentData);

        // Trata o resultado da transação
        ...
    }
}
```

Estornar um pagamento

```
public void voidPayment(Context context) {

    // Cria a identificação do aplicativo
    PlugPagAppIdentification appIdentification =
            new PlugPagAppIdentification("MeuApp", "3.1.1");

    // Cria a referência do PlugPag
    PlugPag plugpag = new PlugPag(context, appIdentification);

    // Ativa terminal e faz o pagamento
    int initResult = plugpag.initializeAndActivatePinpad(new
    PlugPagActivationData("SEU_CODIGO_DE_ATIVAÇÃO")));

    if (initResult == PlugPag.RET_OK) {
        PlugPagTransactionResult result = plugpag.voidPayment(
                                "transactionCode",
                                "transactionId");

        // Trata o resultado do estorno
        ...
    }
}
```

Verificar autenticação

```
public void checkAuthentication(Context context) {
    // Cria a identificação do aplicativo
    PlugPagAppIdentification appIdentification =
            new PlugPagAppIdentification("MeuApp", "3.1.1");

    // Cria a referência do PlugPag
    PlugPag plugpag = new PlugPag(context, appIdentification);

    // Verifica autenticação
    boolean authenticated = plugpag.isAuthenticated();

    if (authenticated) {
        // Usuário autenticado
        ...
    } else {
        // Usúario não autenticado
        ...
    }
}
```

Invalidar autenticação

```
public void logout(Context context) {
    // Cria a identificação do aplicativo
    PlugPagAppIdentification appIdentification =
            new PlugPagAppIdentification("MeuApp", "3.1.1");

    // Cria a referência do PlugPag
    PlugPag plugpag = new PlugPag(context, appIdentification);

    // Invalida a autenticação existente
    plugpag.invalidateAuthentication();
    ...
}
```

Solicitar autenticação

```
public void showAuthenticationActivity(Activity activity) {
    // Cria a identificação do aplicativo
    PlugPagAppIdentification appIdentification =
            new PlugPagAppIdentification("MeuApp", "3.1.1");

    // Cria a referência do PlugPag
    PlugPag plugpag = new PlugPag(activity, appIdentification);

    // Implementa a interface PlugPagAuthenticationListener
    PlugPagAuthenticationListener authListener =
        new PlugPagAuthenticationListener() {
            @Override
            void onSuccess() { … }

            @Override
            void onError() { … }
        };

    // Solicita autenticação
    plugpag.requestAuthentication(myAuthenticationListener);
    ...
}
```

O resultado da autenticação será retornado através da classe PlugPagInitializationResult.

Se a autenticação for efetuada com sucesso, o método getResult irá retornar valor igual a PlugPag.RET_OK. Caso contrário, mais informações do erro podem ser encontradas nos métodos `getErrorCode`() e `getErrorMessage`().

Obter versão da biblioteca

```
public void getLibVersion(Context context) {
    // Cria a identificação do aplicativo
    PlugPagAppIdentification appIdentification =
            new PlugPagAppIdentification("MeuApp", "3.1.1");

    // Cria a referência do PlugPag
    PlugPag plugpag = new PlugPag(context, appIdentification);

    // Obtém a versão da biblioteca
    String version = plugpag.getLibVersion();
}
```

Reimpressão da via do estabelecimento

```
public void reprintStablishmentReceipt() {
    // Cria a identificação do aplicativo
    PlugPagAppIdentification appIdentification =
            new PlugPagAppIdentification("MeuApp", "3.1.1");

    // Cria a referência do PlugPag
    PlugPag plugpag = new PlugPag(context, appIdentification);

    // Obtém a versão da biblioteca
    PlugPagPrintResutlt result = plugpag.reprintStablishmentReceipt();
}
```

Reimpressão da via do cliente

```
public void reprintCustomerReceipt() {
    // Cria a identificação do aplicativo
    PlugPagAppIdentification appIdentification =
            new PlugPagAppIdentification("MeuApp", "3.1.1");

    // Cria a referência do PlugPag
    PlugPag plugpag = new PlugPag(context, appIdentification);

    // Obtém a versão da biblioteca
    PlugPagPrintResutlt result = plugpag.reprintCustomerReceipt();
}
```

Calcular parcelas

```
public void calculateInstallments() {
    // Cria a identificação do aplicativo
    PlugPagAppIdentification appIdentification =
            new PlugPagAppIdentification("MeuApp", "3.1.1");

    // Cria a referência do PlugPag
    PlugPag plugpag = new PlugPag(context, appIdentification);

    // Obtém a versão da biblioteca
    Array<String> installments = plugpag.calculateInstallments(saleValue);
}
```

O resultado retornado será uma String contendo a quantidade de parcelas permitidas mais o valor de cada parcela, seguindo o padrão “quantidadeDeParcelasx R$ valorDaParcela”.

Códigos de retorno

Os códigos de retorno descritos abaixo são obtidos ao chamar o método `getResult()` de um `PlugPagTransactionResult` retornado por um dos métodos de transação de um objeto `PlugPag`: `doPayment(PlugPagPaymentData)`, `voidPayment(PlugPagVoidData)`, `voidPayment()` e `getLastApprovedTransaction()`.

Valor	Descrição	Ação
0	Transação concluída com sucesso.	
-1001	Mensagem gerada maior que buffer dimensionado.	Coletar log (se existir) e enviar para o suporte.
-1002	Parâmetro de aplicação inválido.	Coletar log (se existir) e enviar para o suporte.
-1003	Terminal não está pronto para transacionar.	Tente novamente.
-1004	Transação não realizada.	Verificar mensagem retornada.
-1005	Buffer de resposta da transação inválido ao obter as informações de resultado da transação.	Realizar consulta de última transação.
-1006	Parâmetro de valor da transação não pode ser nulo.	Verificar implementação da chamada da biblioteca.
-1007	Parâmetro de valor total da transação não pode ser nulo.	Verificar implementação da chamada da biblioteca.
-1008	Parâmetro de código de venda não pode ser nulo.	Verificar implementação da chamada da biblioteca.

-1009	Parâmetro de resultado da transação não pode ser nulo.	Verificar implementação da chamada da biblioteca.
-1010	Driver de conexão não encontrado.	Verificar se todos os arquivos estão no diretório correto.
-1011	Erro ao utilizar driver de conexão.	Reinstalar os arquivos do driver de conexão.
-1012	Formato do valor da venda inválido.	Valor deve ser um número inteiro sem vírgula.
-1013	Comprimento do código de venda superior a 10 dígitos.	Truncar código de venda para no máximo 10 dígitos.
-1014	Buffer de recepção corrompido.	Refaça a transação.
-1015	Nome da aplicação maior que 25 caracteres.	Limitar nome da aplicação a 25 caracteres.
-1016	Versão da aplicação maior que 10 caracteres.	Limitar versão da aplicação em 10 caracteres.
-1017	Necessário definir nome da aplicação.	Definir nome e versão da aplicação com `setVersionName(String, String)`
-1018	Não existem dados da última transação.	Refaça a transação.
-1019	Erro de comunicação com terminal (resposta inesperada).	Realizar consulta de última transação.
-1024	Erro na carga de tabelas.	Refazer inicialização (carga de tabelas).
-1030	Token não encontrado	Refazer autenticação.

-1031	Valor inválido	Verificar o valor configurado para pagamento e tentar novamente. Valor mínimo: R$ 1,00
-1032	Parcelamento inválido	Verificar o número de parcelas e tentar novamente.
-2001	Porta COM informada não encontrada.	Informar uma porta COM válida.
-2002	Não foi possível obter configurações da porta COM informada.	Informar uma porta COM válida.
-2003	Não foi possível configurar a porta COM informada.	Informar uma porta COM válida.
-2005	Não foi possível enviar dados pela porta COM informada.	Informar uma porta COM válida.
-2022	Java – Adaptador Null.	Verificar implementação.
-2023	Java – erro em DeviceToUse.	Coletar log (se existir) e enviar para o suporte.
-2024	Java – erro no serviço RfcommSocket.	Coletar log (se existir) e enviar para o suporte.
-2026	Java – Close exception.	Coletar log (se existir) e enviar para o suporte.
-3001	Permissão de root	Remover permissão de root do aparelho.
-4046	Não existe dados de autenticação	Efetuar a autenticação.