Manual de Integração
Android

Sumário

OBJ

OBJ

OBJ

OBJ

OBJ

OBJ

OBJ

OBJ

OBJ

OBJ

OBJ

OBJ

OBJ

OBJ

OBJ

OBJ

OBJ

OBJ

OBJ

OBJ

OBJ

OBJ

OBJ

OBJ

OBJ

OBJ

OBJ

OBJ

OBJ

OBJ

OBJ

OBJ

Versionamento

Versão doc.	Data	Autor	Descrição	Versão PlugPag
1.0.0	03/04/2018	Hugo Yamashita	Criação do documento.	3.0.0
1.0.1	04/04/2018	Hugo Yamashita	Correção de dependências para criação de aplicativos.	3.0.0
1.0.2	25/04/2018	Hugo Yamashita	Novos códigos de erros. Mais informações no resultado de transações.	3.0.1
1.0.3	24/10/2018	Hugo Yamashita	PlugPag versão 3.0.3, com suporte a Minizinha Chip.	3.0.3
1.0.4	09/11/2018	Hugo Yamashita	PlugPag versão 3.1.0 com suporte a transações por tarja.	3.1.0
1.0.5	10/12/2018	Hugo Yamashita	PlugPag versão 3.1.1 com correções de bugs.	3.1.1
1.0.6	10/05/2019	Hugo Yamashita	PlugPag versão 3.1.2 com correção de bugs.	3.1.2
1.0.7	03/11/2021	Igor Rotondo Bagliotti	Atualização documentação PlugPag	4.2.0
1.08	29/11/2021	José Pereira		4.5.1

Introdução

Este documento destina-se a integradores que utilizarão os terminais **Moderninha PRO, Moderninha Wifi**, **Moderninha Wifi Plus**, **Moderninha Plus** e **Minizinha Chip*** ou os leitores **Minizinha, Mini** e **Mob Pin 10** da PagSeguro como solução de pagamento integrada através da biblioteca **PlugPag PagSeguro** com dispositivos **Android.**

A biblioteca **PlugPag** permite aos integradores implementar aplicativos que consigam comunicar via bluetooth com os terminais e leitores da **PagSeguro**.

* A Minizinha Chip deve estar com versão igual ou superior a 1.0.11 para aceitar conexões bluetooth

Pareamento bluetooth

O pareamento bluetooth deve ser realizado pelo menu de pareamento do seu dispositivo Android.

As moderninhas são identificadas pelo padrão "**Modelo-nº de série"**. Os leitores **Mini** e **Minizinha** são identificados pelo padrão "**PAX-nº de série".** O leitor **Mob Pin 10** é identificado pelo padrão "**MOBI-nº de série"** ou **"MOBIPIN-nº de série"**.

Para tornar visível o bluetooth dos dispositivos, basta apertar a tecla ‘0’.

Exemplos de identificação

Moderninha Pro: PRO-12345678

Moderninha Wifi: W-12345678

Moderinha Wifi Plus: W+-123456789

Moderninha Plus: PLUS-12345678

Minizinha Chip: MCHIP-123456789

Mini / Minizinha: PAX-12345678

Mobi Pin 10: MOBI-12345678 / MOBIPIN-12345678

Importando a biblioteca PlugPag

Para importar a biblioteca PlugPag 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/pagseguromaster/plugpag/raw/master/3.x/android'
  }
  ...
 }
}
```

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

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

    implementation 'br.com.uol.pagseguro:plugpag:3.1.2'
    ...
}
```

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.

3- Para integrações utilizando os modelos de smartphone M30 (PagPhone) – smartphone com leitor de pagamento integrado da PagSeguro – é necessário a inclusão de dependência da biblioteca Neptune para uso do pinpad.

```
    ...
    ...
}
```

Para realizar o download do arquivo .JAR, o mesmo poderá ser localizado no repositório

AndroidManifest.xml

Permissões

Para integrar a biblioteca a biblioteca PlugPag em aplicativos para Android é necessário adicionar algumas permissões no arquivo `AndroidManifest.xml`.

Permissões obrigatórias

```
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.BLUETOOTH"/>
<uses-permission android:name="android.permission.READ_PHONE_STATE"/>
```

Sem essas permissões, não será possível fazer transações utilizando os terminais e leitores da PagSeguro.

Permissões opcionais

```
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
```

Essas permissões permitem à biblioteca obter coordenadas no momento da transação. Essas coordenadas são enviadas aos servidores do PagSeguro e ajudam a melhorar nossos serviços.

Activity

Algumas funcionalidades da biblioteca PlugPag necessitam que uma `Activity` seja iniciada.

Para isso, é necessário incluir o trecho abaixo:

```
<application ...>
    ...
    <activity
        android:name="br.com.uol.pagseguro.plugpag.PlugPagActivity" />
    ...
</application>
```

Manifest Merger

Caso o Android Studio e o SDK do Android estejam atualizados, não há necessidade de incluir as linhas acima no *AndroidManifest.xml* do seu aplicativo pois o Manifest Merger fará isso.

Para garantir que o Manifest Merger funcionou corretamente, abra o `AndroidManifest.xml` no Android Studio, clique em "Merged Manifest", na parte inferior do editor e certifique-se de que as linhas acima (permissões e Activity) estão presentes.

Classes

A biblioteca PlugPag 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
DeviceInfo	Informações sobre o aparelho (smartphone/tablet) utilizado.
PlugPag	Classe principal da biblioteca. Essa classe é responsável pela configuração de comunicação com os dispositivos bluetooth e pelas transações.
PlugPagAbortResult	Resultado obtido ao solicitar um cancelamento de operação, enquanto a operação está em andamento.
PlugPagAppIdentification	Identificação do aplicativo.
PlugPagDevice	Identificação do terminal ou leitor que será utilizado para as transações.
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.

PlugPagBluetoothException	Exceção lançada quando ocorrer algum problema durante a configuração do bluetooth ou durante a comunicação via bluetooth.
sPlugPagDeviceInfoException	Exceção lançada quando ocorrer algum problema durante a configuração do terminal ou leitor.
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.

Observações

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

- A biblioteca PlugPag possui suporte da API level 16 (4.1 Jelly Bean) à 28 (9.0 Pie), devido a restrições de protocolos de comunicação.
- 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.
- Por utilizar bluetooth para comunicação, o dispositivo Android não deve estar muito distante do terminal ou leitor.
- 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.
- Não é possível fazer chamadas da biblioteca caso o usuário tenha permissões de `root` no aparelho por motivos de segurança.

API

Abaixo segue a descrição da interface pública da biblioteca PlugPag para aparelho com sistema operacional Android.

DeviceInfo

Essa classe possui métodos para obtenção de informações do aparelho (smartphone/tablet) utilizado.

Alguns dados obtidos por essa classe são carregados quando solicitados em vez de fazer cache de informações, fazendo com que algumas operações tenham um tempo de execução longo.

Construtores

DeviceInfo(Context context)

Cria um DeviceInfo utilizando o `context` fornecido para buscar informações do aparelho quando necessário.

Gera uma exceção se `context` for nulo.

Métodos

Tipo de retorno	Método e descrição
String	getImei() Obtém o IMEI do aparelho. Se a permissão `android.permission.READ_PHONE_STATE` não for concedida, returna `null`.
String	getVersion() Retorna a versão do sistema operacional.

Pair<Double, Double>	getCoordinates() Obtém as coordenadas do aparelho. A disponibilidade e precisão das coordenadas vão depender das configurações do aparelho e disponibilidade de sinais de GPS, Wi-Fi e telefonia. Se as permissões `android.permission.ACCESS_COARSE_LOCATION` e `android.permission.ACCESS_FINE_LOCATION` não forem concedidas, a coordenada retornada será sempre `(0, 0)`.
String	getDeviceModel() Retorna o nome da fabricante e o modelo do aparelho.
String	getDeviceId() Calcula um ID único para o aparelho.
String	getOs() Retorna a constante `"ANDROID"`.
String	getOsVersion() Retorna a versão do sistema operacional.

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: 43981
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 PlugPag.
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.
PlugPagTransactionResult	VoidQrCodePayment(PlugPagvoidData voidData) Irá gerar um QrCode de estorno de uma transação. Retornará 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.

PlugPagDevice

Essa classe representa um terminal ou um leitor.

Constantes

int	TYPE_UNDEFINED Tipo de terminal ou leitor indefinido. Valor: -1
int	TYPE_PINPAD Tipo: leitor Valor: 0
int	TYPE_TERMINAL Tipo: terminal Valor: 1

Construtores

PlugPagDevice(String identification)

Cria uma identificação de terminal ou leitor utilizando `identification` como nome ou MAC address do dispositivo.

Gera uma exceção se `identification` for nulo ou vazio.

Métodos

String	getIdentification() Retorna a identificação do leitor ou terminal.

int getType()

Retorna o tipo do leitor ou terminal, de acordo com as constantes dessa classe.

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)

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 `amount` definido é o valor em centavos a ser pago. Para um pagamento de R$ 1,50, o `amount` deverá ser de `150`.

O `userReference` só é editável se o pagamento for feito integrado com as Moderninhas PRO ou WIFI. **Atualmente não é aplicável para leitores**.

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)

Cria um conjunto de informações para solicitar o estorno de um pagamento identificado pelo `transactioCode` e `transactionId` fornecidos.

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", "3.1.2");

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

    // Prepara conexão bluetooth e faz o pagamento
    int initResult = plugpag.initBTConnection(device);

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

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

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

```
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,
                    30000,
                    PlugPag.INSTALLMENT_TYPE_PARC_VENDEDOR,
                    3,
                    "CODVENDA");

    // Cria a identificação do aplicativo
    PlugPagAppIdentification appIdentification =
            new PlugPagAppIdentification("MeuApp", "3.1.2");

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

    // Prepara conexão bluetooth e faz o pagamento
    int initResult = plugpag.initBTConnection(device);

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

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

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

```
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,
                    30000,
                    PlugPag.INSTALLMENT_TYPE_PARC_COMPRADOR,
                    3,
                    "CODVENDA");

    // Cria a identificação do aplicativo
    PlugPagAppIdentification appIdentification =
            new PlugPagAppIdentification("MeuApp", "3.1.2");

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

    // Prepara conexão bluetooth e faz o pagamento
    int initResult = plugpag.initBTConnection(device);

    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", "3.1.2");

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

    // Prepara conexão bluetooth e faz o pagamento int
    initResult = plugpag.initBTConnection(device);

    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", "3.1.2");

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

    // Prepara conexão bluetooth e faz o pagamento
    int initResult = plugpag.initBTConnection(device);

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

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

Pagamento de R$50, no QrCode Elo 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_QRCODE_ELO_DEBITO,
                    5000,
                    PlugPag.INSTALLMENT_TYPE_A_VISTA,
                    1,
                    "CODVENDA");

    // Cria a identificação do aplicativo
    PlugPagAppIdentification appIdentification =
            new PlugPagAppIdentification("MeuApp", "3.1.2");

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

    // Prepara conexão bluetooth e faz o pagamento
    int initResult = plugpag.initBTConnection(device);

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

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

Pagamento de R$50, no Pix:

```
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_PIX,
                    5000,
                    PlugPag.INSTALLMENT_TYPE_A_VISTA,
                    1,
                    "CODVENDA");

    // Cria a identificação do aplicativo
    PlugPagAppIdentification appIdentification =
            new PlugPagAppIdentification("MeuApp", "3.1.2");

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

    // Prepara conexão bluetooth e faz o pagamento
    int initResult = plugpag.initBTConnection(device);

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

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

Pagamento de R$50, no QrCode Elo Crédito:

```
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_QRCODE_ELO_CREDITO,
                    5000,
                    PlugPag.INSTALLMENT_TYPE_A_VISTA,
                    1,
                    "CODVENDA");

    // Cria a identificação do aplicativo
    PlugPagAppIdentification appIdentification =
            new PlugPagAppIdentification("MeuApp", "3.1.2");

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

    // Prepara conexão bluetooth e faz o pagamento
    int initResult = plugpag.initBTConnection(device);

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

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

Estornar um pagamento

Terminais

```
public void voidPayment(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);

    // Cria a identificação do aplicativo
    PlugPagAppIdentification appIdentification =
            new PlugPagAppIdentification("MeuApp", "3.1.2");

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

    // Prepara conexão bluetooth e faz o pagamento
    int initResult = plugpag.initBTConnection(device);

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

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

Leitores

```
public void voidPayment(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 estorno
    PlugPagVoidData voidData =
            new PlugPagVoidData
                    .Builder()
                    .setTransactionCode("transactionCode")
                    .setTransactionId("transactionId")
                    .build();

    // Cria a identificação do aplicativo
    PlugPagAppIdentification appIdentification =
            new PlugPagAppIdentification("MeuApp", "3.1.2");

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

    // Prepara conexão bluetooth e faz o pagamento
    int initResult = plugpag.initBTConnection(device);

    if (initResult == PlugPag.RET_OK) {
        PlugPagTransactionResult result = plugpag.voidPayment(voidData);

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

Consultar última transação aprovada

Terminais

```
public void getLastApprovedTransaction(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);

    // Cria a identificação do aplicativo
    PlugPagAppIdentification appIdentification =
            new PlugPagAppIdentification("MeuApp", "3.1.2");

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

    // Prepara conexão bluetooth e faz o pagamento
    int initResult = plugpag.initBTConnection(device);

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

        // Trata os dados da transação
        retornada ...
    }
}
```

Leitores

Dados das transações feitas com leitores devem ser armazenadas pela aplicação.

Se for feita uma chamada do método `getLastApprovedTransaction()` com uma conexão configurada para um leitor, será retornado `null`.

Verificar autenticação

```
public void checkAuthentication(Context context)
    { // Cria a identificação do aplicativo
    PlugPagAppIdentification appIdentification =
            new PlugPagAppIdentification("MeuApp", "3.1.2");

    // 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", "3.1.2");

    // 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", "3.1.2");

    // 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 método `PlugPag.requestAuthentication(PlugPagAuthenticatinoListener)` inicia uma nova `Activity` para efetuar a autenticação de forma segura.

O resultado da autenticação será passado para `myAuthenticationListener`, uma implementação da interface `PlugPagAuthenticationListener`.

Se a autenticação for efetuada com sucesso, o método `onSuccess()` será invocado. Caso contrário, o método invocado é o `onError()`.

Abortar transação

```
public void abortTransaction(Context context) throws InterruptedException
    { // 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
    final 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", "3.1.2");

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

    // Prepara conexão bluetooth e faz o pagamento
    int initResult = plugpag.initBTConnection(device);

    if (initResult == PlugPag.RET_OK) {
        // Cria uma Thread para fazer um
        pagamento new Thread(new Runnable() {
            @Override
            public void run() {
                PlugPagTransactionResult result =
                        plugpag.doPayment(paymentData);
            }
        }).start();

        // Aguarda 2 segundos e aborta a
        transação Thread.sleep(2000);
        PlugPagAbortResult abortResult = plugpag.abort();
    }
}
```

Obter versão da biblioteca

```
public void getPlugPagVersion(Context context) {
    // Cria a identificação do aplicativo
    PlugPagAppIdentification appIdentification =
            new PlugPagAppIdentification("MeuApp", "3.1.2");

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

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

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.
-1020	Transação por Bluetooth não permitida quando o terminal está em modo compartilhado.	Desativar modo compartilhado.

-1024	Erro na carga de tabelas.	Refazer inicialização (carga de tabelas).
	Erro de comunicação bluetooth. Verificar se o terminal/leitor está ligado, verificar se o bluetooth do aparelho está ligado e tentar novamente.	
-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.
-2004	Timeout de comunicação Bluetooth.	Refaça a transação.
-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.
-2027	Java - Bluetooth socket	Reiniciar o aplicativo ou o terminal e tentar novamente.
-2033	Bluetooth socket	Reiniciar o aplicativo ou o terminal e tentar novamente.
-3000	Permissões não concedidas	Incluir permissões no `AndroidManifest.xml` e solicitá-las ao usuário em tempo de execução.
-3001	Permissão de root	Remover permissão de root do aparelho.
-4046	Não existe dados de autenticação	Efetuar a autenticação.

Constantes

```
public class PlugPag {

   public static final int RET_OK;
   public static final int BUFF_SIZE;
   public static final int NULL_PTR;
   public static final int POS_NOT_READY;
   public static final int TRANS_DENIED;
   public static final int DATA_INV_RESULT_MESSAGE;
   public static final int INV_AMOUNT_PARAM;
   public static final int INV_TOT_AMOUNT_PARAM;
   public static final int INV_USER_REF_PARAM;
   public static final int INV_TRS_RESULT_PARAM;
   public static final int DRIVER_NOT_FOUND;
   public static final int DRIVER_FUNCTION_ERROR;
   public static final int JNI_EXIT_EXCEPTION;

   public static final int CREDIT = 1;
   public static final int DEBIT = 2;
   public static final int VOUCHER = 3;
   public static final int  QRCODE_ELO_DEBITO = 4;
   public static final int  PIX = 5;
   public static final int  TYPE_QRCODE_ELO_CREDITO = 7;
   public static final int A_VISTA = 1;
   public static final int PARC_VENDEDOR = 2;
   public static final int PARC_COMPRADOR = 3;
}
```