Manual de Integração
iOS

Sumário

Versionamento

Versão Doc.	Data	Autor	Descrição	Versão PlugPag
1.0.1	15/02/2018	Hildequias Junior	Criação do documento	3.0.0
1.0.2	21/06/2018	Hildequias Junior	Adicionado suporte para D180	3.0.1
1.0.3	23/10/2018	Hildequias Junior	Adicionado suporte para Minizinha Chip	3.0.2
1.0.4	03/10/2019	Hildequias Junior	Correções de bugs	3.2.0

Introdução

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

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

Programa MFI

Algumas máquinas não possuem suporte a BLE 4.0, sendo assim é necessário se inscrever no programa Make for iPhone (MFI) dos fabricantes para integrar com aplicativos iOS. Os modelos que fazem parte são: (Mobi Pin 10, D200 e Mini), esse programa é uma exigência da Apple para que seu aplicativo tenha permissão de uso desses hardwares externos.

Se esse for o seu caso é necessário entrar em contato com o PagSeguro e solicitar a inscrição do seu aplicativo antes da publicação na Apple Store.

Pareamento Bluetooth

O pareamento bluetooth deve ser realizado pela biblioteca PlugPag com seu dispositivo iOS.

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** padrão e a **Minizinha Chip** pelo **MCHIP-nº de série.**

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

Exemplo de implementação do pareamento pode ser visto na seção **Exemplo de utilização para pareamento**.

Requisitos Mínimos

- Bluetooth Low Energy (BLE, SMART) 4.0.
- IOS 8.0.

Importando a biblioteca PlugPag

Para importar a biblioteca PlugPag na sua aplicação nativa iOS basta seguir os passos descritos abaixo:

- Inserir o arquivo **PlugPag.framework** no Build Phases -> Link Binary with Libraries do seu projeto.
- Inserir o arquivo **PlugPag.bundle** no seu projeto, caso esteja usando Swift é preciso criar um arquivo Bridging-Header e importar o arquivo PlugPag.h.

Ex:

```
#import <PlugPag/PlugPag.h>
```

- Em Build Settings do seu projeto alterar Enable Bitcode para No.
- Inserir as seguintes Cocoa Keys no arquivo Info.plist para descrever o uso do Bluetooth e Location da PlugPag. **NSBluetoothPeripheralUsageDescription** - Privacy - Bluetooth Peripheral Usage Description

  NSLocationWhenInUseUsageDescription - Privacy - Location When In Use Usage Description

- **UI**SupportedExternalAccessoryProtocols - **(**Array) -

  com.paxsz.iPOS

  br.com.gertec.protocoloGertec

  Obs: só deve ser informado esses bundle quando seu aplicativo for fazer uso dos modelos Mini, Mobi Pin 10 e D200, para mais informações acesse a sessão MFI.

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
PlugPag	Classe principal da biblioteca. Essa classe é responsável pela configuração e comunicação com os dispositivos bluetooth e pelas transações.
PlugPagAbortResult	Resultado obtido ao solicitar um cancelamento de operação, enquanto a operação está em andamento.
PlugPagDevice	Identificação do terminal ou leitor que será utilizado para as transações.
PlugPagPaymentData	Informações de um pagamento a ser realizado.
PlugPagTransactionResult	Resultado de uma transação.
PlugPagVoidData	Informações de um estorno a ser realizado.
PlugPagInitializationResult	Resultado de uma inicialização da plugpag.
PlugPagInstallmentResult	Resultado do cálculo de parcelamento vendedor ou comprador.
PlugPagInstallment	Informações do parcelamento.

PlugPag

Delegates:	-(void) peripheralDiscover:(PlugPagDevice*) plugPagDevice; -(void) userEventsInterface:(int) event; -(void) pairPeripheralStatus:(int) status;
Métodos:	-(void) setDelegate:(id<PP_PlugPagDelegate>) delegate; -(void) pairPeripheral: (PlugPagDevice *) plugPagDevice; -(void) cancelPairPeripheral; -(void) startScanForPeripherals; -(void) stopScanForPeripherals; -(BOOL) isPlugPagDeviceConnected:(PlugPagDevice *) device; -(const char *) getVersionLib; -(int) setInitBTConnection:(PlugPagDevice *) peripheral; -(PlugPagInitializationResult *) initializeAndActivate: (PlugPagActivationData *) activationData; -(PlugPagTransactionResult *) doPayment:(PlugPagPaymentData *) paymentData; -(PlugPagTransactionResult *) voidPayment:(PlugPagVoidData *) voidData; -(PlugPagAbortResult *) abort; -(PlugPagTransactionResult *) getLastApprovedTransaction; -(PlugPagInstallmentResult *) calculateInstallments: (NSString *) saleValue type:(InstallmentType) installmentType; -(int) plugPagAppIdentification:(NSString *) appName withVersion:(NSString *) appVersion;

-(BOOL) isAuthenticated;

-(void) invalidateAuthentication;

-(void) requestAuthentication:(UIViewController *) viewController;

PlugPagDevice

Atributos:	DeviceTypes mType;
	NSString *mPeripheralName;
	NSString *mPeripheralModel;

PlugPagAbortResult

Atributo:	int mResult;

PlugPagPaymentData

Atributos:	PaymentMethod mType;
	int mAmount;
	InstallmentType mInstallmentType;
	int mInstallment;
	NSString *mUserReference;
	UIViewController *mViewController;

PlugPagTransactionResult

Atributos:	int mResult;
	NSString *mMessage;

	NSString *mTransactionCode; NSString *mDate; NSString *mTime; NSString *mHostNsu; NSString *mCardBrand; NSString *mBin; NSString *mHolder; NSString *mUserReference; NSString *mTerminalSerialNumber; NSString *mTransactionId; NSString *mAmount; NSString *mAvailableBalance; NSString *mCardApplication; NSString *mCardCryptogram; NSString *mLabel; NSString *mHolderName; NSString *mExtendedHolderName; NSString *mErrorCode;

PlugPagVoidData

Atributos:	NSString *mTransactionCode; NSString *mTransactionId;

PlugPagInitializationResult

Atributos:	int mResult;
	NSString *mErrorMessage;
	NSString *mErrorCode;

PlugPagInstallmentResult

Atributos:	int mResult;
	NSString *mMessage;
	NSString *mErrorCode;
	NSMutableArray *installments;
	NSString *rate;

PlugPagInstallments

Atributos:	int mQuantity;
	int mAmount;
	int mTotal;

Métodos

getVersionLib:

Retorna um char com a versão da biblioteca PlugPag.

Parâmetro:

- nenhum

Retorno:

- char – versão da biblioteca do PlugPag

setInitBTConnection:

Configura dispositivo bluetooth para ser usado em transações.

Parâmetro:

- PlugPagDevice – Contém os dados da máquina que deseja realizar a transação

Retorno:

- Int – Resultado da operação

doPayment

Inicia a transação de venda. Em caso de sucesso, retorna os dados da transação no objeto PlugPagTransactionResult.

Parâmetro:

- PlugPagPaymentData – Contém os dados da transação que deseja realizar

Retorno:

- PlugPagTransactionResult – Retorna um objeto com os dados da transação

voidPayment

Inicia a transação de estorno. Em caso de sucesso, retorna os dados da transação no objeto PlugPagTransactionResult.

Parâmetro:

- PlugPagPaymentData – Contém os dados da transação que deseja realizar

Retorno:

- PlugPagTransactionResult – Retorna um objeto com os dados da transação

getLastApprovedTransaction

Recebe dados da última transação finalizada com sucesso pelo terminal.

Caso exista uma transação em andamento no momento da consulta, aguarda a finalização da mesma e retorna os dados desta se for aprovada.

Parâmetro:

- Nenhum

Retorno:

- PlugPagTransactionResult - Retorna um objeto com os dados da ultima transação

abort

Aborta a transação em andamento quando utilizado pinpad.

Parâmetro:

- Nenhum

Retorno:

- PlugPagAbortResult – Retorna um objeto com resultado da operação

plugPagAppIdentification: withVersion

Informa o nome e a versão da automação que está utilizando a PlugPag.

Parâmetro:

- appName – Nome da automação comercial que está usando a biblioteca
- appVersion – Versão da automação comercial que está usando a biblioteca

Retorno:

- int - resultado da operação

isAuthenticated

Retorna *se o usuário está autenticado em sua conta Pagseguro*

Parâmetro:

- Nenhum

Retorno:

- BOOL – Retorna verdadeiro para autenticado e falso para não autenticado

invalidateAuthentication

Esse método efetua um logout na conta autenticada do *Pagseguro*, sendo necessário uma nova autenticação para o uso de pinpad.

Parâmetro:

- Nenhum

Retorno:

- Nenhum

requestAuthentication

Inicia o processo de autenticação na conta PagSeguro. Uma UIViewController é executada no device para autenticação na conta PagSeguro.

Parâmetro:

- UIViewController – A UI exibida no device deve ser passada como parâmetro para a inicialização da UI de login do PagSeguro

Retorno:

- Nenhum

startScanForPeripherals

Inicia o processo de scanner para encontrar maquinas PagSeguro que estejam próximas.

Parâmetro:

- Nenhum

Retorno:

- Nenhum

pairPeripheral

Inicia o processo de pareamento com a máquina PagSeguro informada por parametro

Parâmetro:

- PlugPagDevice – Objeto contendo as informações para a PlugPag encontrar o device selecionado.

Retorno:

- Nenhum

setDelegate

Define o delegate do destinatário para um determinado objeto.

Parâmetro:

- Nenhum

Retorno:

- String - bin

peripheralDiscover

Delegate disparado após a PlugPag encontrar uma nova máquina PagSeguro próxima durante o scanner

Parâmetro:

- PlugPagDevice – Objeto com as informações do device encontrado.

Retorno:

- Nenhum

userEventsInterface

Delegate disparado contendo os eventos durante uma transação com pinpad.

Parâmetro:

- event – int que representa o evento Ex: (PP_STATUS_WAITING_CARD = Esperando a inserção do cartão)

Retorno:

- Nenhum

pairPeripheralStatus

Delegate disparado contendo o resultado do pareamento do device iOS com a máquina PagSeguro.

Parâmetro:

- event – int que representa o evento Ex: (BT_PAIR_STATE_OK = Pareamento efetuado com sucesso)

Retorno:

- Nenhum

stopScanForPeripherals

finaliza o processo de scanner para encontrar máquinas PagSeguro que estejam próximas.

Parâmetro:

- Nenhum

Retorno:

- Nenhum

stopScanForPeripherals

finaliza o processo de scanner para encontrar máquinas PagSeguro que estejam próximas.

Parâmetro:

- Nenhum

Retorno:

- Nenhum

isPlugPagDeviceConnected

Informa se o device passado por parâmetro está conectado.

Parâmetro:

- device – Objeto do tipo PlugPagDevice

Retorno:

- BOOL – Retorna verdadeiro para conectado e falso para não conectado

Exemplos de utilização para terminais Moderninha Pro, Minizinha Chip e Moderninha Plus

Venda, Crédito, Parcelado Vendedor, 7 parcelas, R$ 1208,34

```
PlugPagDevice *device = [PlugPagDevice new]; // configura a máquina que realizara a transação
device.mPeripheralName = @"PRO-68000001";

[[PlugPag sharedInstance] plugPagAppIdentification:@"MyApp" withVersion:@"R001"];
int ret = [[PlugPag sharedInstance] setInitBTConnection:device];

if (ret == RET_OK) {

    NSString *value = @"128034"; // Transação de R$ 1.208,34
    int numeroParcelas = 7; // Venda, Crédito, Parcelado Vendedor, 7 parcelas, R$ 1.208,34

    PlugPagPaymentData *data = [PlugPagPaymentData new];
    data.mType = CREDIT;
    data.mAmount = [value intValue];
    data.mInstallmentType = PARC_VENDEDOR;
    data.mInstallment = numeroParcelas;
    data.mUserReference = @"CODIGVENDA";

    PlugPagTransactionResult *result = [[PlugPag sharedInstance] doPayment:data];

    if (result.mResult == RET_OK) {
        NSLog(@"%@", result.mMessage);
    }
}
```

Consulta da última transação

```
    PlugPagDevice *device = [PlugPagDevice new]; // configura a máquina que realizara a transação
device.mPeripheralName = @"PRO-68000001";

[[PlugPag sharedInstance] plugPagAppIdentification:@"MyApp" withVersion:@"R001"];
int ret = [[PlugPag sharedInstance] setInitBTConnection:device];


if (ret == RET_OK) {


    PlugPagTransactionResult *result = [[PlugPag sharedInstance] getLastApprovedTransaction];
    if (result.mResult == RET_OK) {
        NSLog(@"%@", result.mMessage);
    }
}
```

Autenticação

```
[[PlugPag sharedInstance] plugPagAppIdentification:@"MyApp" withVersion:@"R001"];
[[PlugPag sharedInstance] requestAuthentication:self];
```

Venda, Crédito, A Vista, R$ 12,34

```
    PlugPagDevice *device = [PlugPagDevice new]; // configura a máquina que realizara a transação
device.mPeripheralName = @"PRO-68000001";

[[PlugPag sharedInstance] plugPagAppIdentification:@"MyApp" withVersion:@"R001"];
int ret = [[PlugPag sharedInstance] setInitBTConnection:device];


if (ret == RET_OK) {


    NSString *value = @"1234"; // Transação de R$ 12,34
    int numeroParcelas = 1; // Venda, Crédito, A Vista, R$ 12,34

    PlugPagPaymentData *data = [PlugPagPaymentData new];
    data.mType = CREDIT;
    data.mAmount = [value intValue];
    data.mInstallmentType = A_VISTA;
    data.mInstallment = numeroParcelas;
    data.mUserReference = @"CODIGVENDA";

    PlugPagTransactionResult *result = [[PlugPag sharedInstance] doPayment:data];

    if (result.mResult == RET_OK) {
        NSLog(@"%@", result.mMessage);
    }
}
```

Estorno

```
    PlugPagDevice *device = [PlugPagDevice new]; // configura a máquina que realizara a transação
device.mPeripheralName = @"PRO-68000001";

[[PlugPag sharedInstance] plugPagAppIdentification:@"MyApp" withVersion:@"R001"];
int ret = [[PlugPag sharedInstance] setInitBTConnection:device];


if (ret == RET_OK) {


    PlugPagVoidData *voidData = [PlugPagVoidData new];
    voidData.mTransactionId = @"TRANSACTIONID";
    voidData.mTransactionCode = @"TRANSACTIONCODE";

    PlugPagTransactionResult *result = [[PlugPag sharedInstance] voidPayment:voidData];
    if (result.mResult == RET_OK) {
        NSLog(@"%@", result.mMessage);
    }
}
```

Exemplos de utilização para pinpad Mini, Minizinha, Mobi Pin 10 e D200

Venda, Crédito, Parcelado Vendedor, 7 parcelas, R$ 1208,34

```
PlugPagDevice *device = [PlugPagDevice new]; // configura a máquina que realizara a transação
device.mPeripheralName = @"PAX-68000001";

[[PlugPag sharedInstance] setInitBTConnection:device];


PlugPagActivationData *data = [PlugPagActivationData new];

data.mActivationCode = ACTIVATE_CODE;

PlugPagInitializationResult *result = [[PlugPag sharedInstance] initializeAndActivate:data]; // inicializa a
plugpag para realizar a transação


if (result.mResult == RET_OK) {


    NSString *value = @"128034"; // Transação de R$ 1.208,34
    int numeroParcelas = 7; // Venda, Crédito, Parcelado Vendedor, 7 parcelas, R$ 1.208,34

    PlugPagPaymentData *data = [PlugPagPaymentData new];
    data.mType = CREDIT;
    data.mAmount = [value intValue];
    data.mInstallmentType = PARC_VENDEDOR;
    data.mInstallment = numeroParcelas;
    data.mUserReference = @"CODIGVENDA";

    PlugPagTransactionResult *transactionResult = [[PlugPag sharedInstance] doPayment:data]; // Inicia o
processo de transação

    if (transactionResult.mResult == RET_OK) {
        NSLog(@"%@", result.mMessage);
    }
}
```

Venda, Crédito, A Vista, R$ 12,34

```
    PlugPagDevice *device = [PlugPagDevice new]; // configura a máquina que realizara a transação
  device.mPeripheralName = @"PAX-68000001";

  [[PlugPag sharedInstance] setInitBTConnection:device];


  PlugPagActivationData *data = [PlugPagActivationData new];

  data.mActivationCode = ACTIVATE_CODE;

  PlugPagInitializationResult *result = [[PlugPag sharedInstance] initializeAndActivate:data]; // inicializa a
plugpag para realizar a transação

  if (result.mResult == RET_OK) {


    NSString *value = @"1234"; // Transação de R$ 12,34
    int numeroParcelas = 1; // Venda, Crédito, A Vista, R$ 12,34

    PlugPagPaymentData *data = [PlugPagPaymentData new];
    data.mType = CREDIT;
    data.mAmount = [value intValue];
    data.mInstallmentType = A_VISTA;
    data.mInstallment = numeroParcelas;
    data.mUserReference = @"CODIGVENDA";

    PlugPagTransactionResult *result = [[PlugPag sharedInstance] doPayment:data]; // Inicia o processo de
transação

    if (result.mResult == RET_OK) {
      NSLog(@"%@", result.mMessage);
    }
  }
```

Estorno

```
    PlugPagDevice *device = [PlugPagDevice new]; // configura a máquina que realizara a transação
  device.mPeripheralName = @"PAX-68000001";

  [[PlugPag sharedInstance] setInitBTConnection:device];


  PlugPagActivationData *data = [PlugPagActivationData new];

  data.mActivationCode = ACTIVATE_CODE;

  PlugPagInitializationResult *result = [[PlugPag sharedInstance] initializeAndActivate:data];


  if (result.mResult == RET_OK) {

    PlugPagVoidData *voidData = [PlugPagVoidData new];
    voidData.mTransactionId = @"TRANSACTIONID";
    voidData.mTransactionCode = @"TRANSACTIONCODE";

    PlugPagTransactionResult *result = [[PlugPag sharedInstance] voidPayment:voidData];
    if (result.mResult == RET_OK) {
      NSLog(@"%@", result.mMessage);
    }
  }
```

Exemplo de utilização para pareamento iOS

```
@interface ViewController : UIViewController <PP_PlugPagDelegate>

@end

@implementation ViewController

#pragma mark - UIViewController cycle
- (void)viewDidLoad {
  [super viewDidLoad];
  // Do any additional setup after loading the view, typically from a nib.

  [[PlugPag sharedInstance] plugPagAppIdentification:@"MyApp" withVersion:@"R001"]; // Informa o nome do app
e versão da automação
  [[PlugPag sharedInstance] setDelegate:self]; // Informa que essa UIViewController deve responder aos delegates
do PlugPag
}

- (void) viewDidAppear:(BOOL)animated {

  [super viewDidAppear:animated];

  [[PlugPag sharedInstance] startScanForPeripherals]; // Inicia o scanner por Peripheral PagSeguro
}

#pragma mark - PlugPag Delegates
-(void) peripheralDiscover:(PlugPagDevice *) plugPagDevice {

  NSLog (@"Successfully received %@", plugPagDevice.mPeripheralName);
```

```
    [[PlugPag sharedInstance] pairPeripheral:plugPagDevice]; // Inicia o pareamento com o Peripheral encontrado
}

-(void) pairPeripheralStatus:(int)status {

    NSLog (@"Successfully received %i notification!", status);

    switch (status) {

        case BT_PAIR_STATE_OK:
            NSLog (@"Successfully Peripheral Connected!"); // Pareamento feito com sucesso
            break;

        case BT_PAIR_STATE_FAIL:
            NSLog (@"Failure Peripheral Connect!"); // Não foi possível efetuar o pareamento
            break;

        default:
            break;
    }
}
```

Códigos de retorno

Valor	Descrição	Ação
0	Transação autorizada	
-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 invalido	Valor deve ser um número inteiro sem virgula
-1013	Comprimento do REF superior a 10 dígitos	Truncar REF 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()

-1018	Não existe 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).
-1025	Erro de comunicação bluetooth	Verificar se o dispositivo está pareado e tentar novamente.
-1030	Falta de autenticação	Realizar o login na aplicação
-1031	Valor inválido	Informar um valor válido.
-1032	Parcelamento inválido	Informar o valor da parcela válido.
-1033	Erro de autenticação	Realizar o login na aplicação
-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	Adaptador Null	Verificar implementação.
-2023	Erro em DeviceToUse	Coletar log (se existir) e enviar para o suporte.
-2024	Erro no serviço RfcommSocket	Coletar log (se existir) e enviar para o suporte.
-2026	Close exception	Coletar log (se existir) e enviar para o suporte.
-4046	Não existe dados de autenticação	Efetuar a autenticação na lib PlugPag através do método requestAuthentication