



# Manual de Integração iOS

## Sumário

Versionamento .....	4
Introdução .....	5
Pareamento Bluetooth .....	6
Importando a biblioteca PlugPag .....	7
Classes .....	8
PlugPag .....	9
PlugPagDevice .....	10
PlugPagAbortResult .....	10
PlugPagPaymentData .....	10
PlugPagTransactionResult .....	10
PlugPagVoidData .....	11
Métodos .....	12
getVersionLib: .....	12
initBTConnection: .....	12
doPayment .....	12
voidPayment .....	13
getLastApprovedTransaction .....	13
abort .....	14
setVersionName: withVersion .....	14
isAuthenticated .....	14
invalidateAuthentication .....	15
requestAuthentication .....	15
startScanForPeripherals .....	15
setPeripheralName .....	16

setDelegate .....	16
pagSeguroPeripheralDiscover .....	16
userEventsInterface .....	17
pairPeripheralStatus .....	17
Exemplos de utilização .....	18
Venda, Crédito, Parcelado Vendedor, 7 parcelas, R\$ 1208,34 .....	18
Consulta da última transação .....	18
Venda, Crédito, A Vista, R\$ 12,34 .....	19
Estorno .....	20
Exemplo de utilização para pareamento iOS .....	20
Códigos de retorno .....	21

## Versionamento

<b>Versão Doc.</b>	<b>Data</b>	<b>Autor</b>	<b>Descrição</b>	<b>Versão PlugPag</b>
1.0.1	15/02/2018	Hildequias Junior	Criação do documento	3.0.0

## Introdução

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

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

## 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**.

- 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**.

## Importando a biblioteca PlugPag

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

- Inserir o arquivo libPlugPag.a no Build Phases -> Link Binary with Libraries do seu projeto.
- Inserir os arquivos a seguir no seu projeto, caso esteja usando Swift é preciso criar um arquivo Bridging-Header e importar o arquivo PlugPag.h.

**PlugPag.h**

**PlugPagAbortResult.h**

**PlugPagDevice.h**

**PlugPagPaymentData.h**

**PlugPagTransactionResult.h**

**PlugPagVoidData.h**

**PlugPag.bundle**

- 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 libPlugPag.

**NSBluetoothPeripheralUsageDescription** - Privacy - Bluetooth Peripheral Usage Description

**NSLocationWhenInUseUsageDescription** - Privacy - Location When In Use Usage Description

## 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.



## 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) startScanForPeripherals;
- (const char \*) getVersionLib;
- (PlugPagTransactionResult \*) initBTConnection:(PlugPagDevice \*) peripheral;
- (PlugPagTransactionResult \*) doPayment:(PlugPagPaymentData \*) paymentData;
- (PlugPagTransactionResult \*) voidPayment:(PlugPagVoidData \*) voidData;
- (PlugPagAbortResult \*) abort;
- (PlugPagTransactionResult \*) getLastApprovedTransaction;
- (int) plugPagAppIdentification:(NSString \*) appName withVersion:(NSString \*) appVersion;
- (BOOL) isAuthenticated;
- (void) invalidateAuthentication;
- (void) requestAuthentication:(UIViewController \*) viewController;

## PlugPagDevice

<b>Atributos:</b>	DeviceTypes mType;
	NSString *mPeripheralName;

---

## PlugPagAbortResult

<b>Atributo:</b>	int mResult;
------------------	--------------

---

## PlugPagPaymentData

<b>Atributos:</b>	PaymentMethod mType;
	int mAmount;
	InstallmentType mInstallmentType;
	int mInstallment;
	NSString *mUserReference;

---

## PlugPagTransactionResult

<b>Atributos:</b>	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;
```

---

#### PlugPagVoidData

<b>Atributos:</b>	NSString *mTransactionCode;
	NSString *mTransactionId;

---

## Métodos

### getVersionLib:

Retorna uma String 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 maquina que deseja realizar a transação

**Retorno:**

- PlugPagTransactionResult – Retorna um objeto com os dados da transação

### 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

#### plugPagApplIdentification: 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

## Exemplos de utilização

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

```
PlugPagDevice *device = [PlugPagDevice new]; // configura a maquina que realizara
a transação
device.mPeripheralName = @"PR0-68000001";

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

if (ret.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 *result = [[PlugPag sharedInstance] doPayment:data];

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

Consulta da última transação

```
PlugPagDevice *device = [PlugPagDevice new]; // configura a maquina que realizara
a transação
device.mPeripheralName = @"PR0-68000001";

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

if (ret.mResult == 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 maquina que realizara  
a transação  
device.mPeripheralName = @"PRO-68000001";  
  
[[PlugPag sharedInstance] plugPagAppIdentification:@"MyApp" withVersion:@"R001"];  
PlugPagTransactionResult *ret = [[PlugPag sharedInstance]  
setInitBTConnection:device];  
  
if (ret.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];  
  
    if (result.mResult == RET_OK) {  
        NSLog(@"%@", result.mMessage);  
    }  
}
```

## Estorno

```
PlugPagDevice *device = [PlugPagDevice new]; // configura a maquina que realizara a transação
device.mPeripheralName = @"PR0-68000001";

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

if (ret.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) viewWillAppear:(BOOL)animated {

    [super viewWillAppear: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 inválido	Valor deve ser um número inteiro sem vírgula
-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 <code>SetVersionName()</code>

-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
-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
-------	----------------------------------	---

---