2015

Manual de integração com Android

stone

JOÃO GABRIEL – DESENVOLVIMENTO DE ANDROID
CRISTIAN DANNER – DESENVOLVIMENTO DE ANDROID

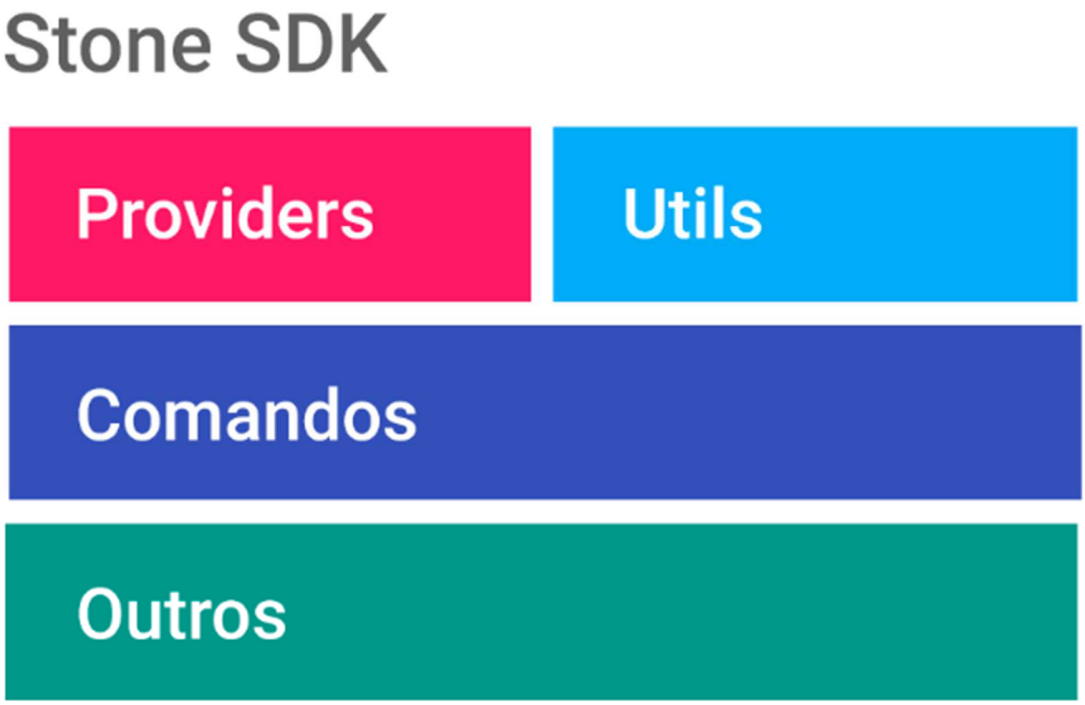
STONE PAGAMENTOS | Rua da Quitanda, 199 10º - Rio de Janeiro – RJ
Build 2.0.3

Índice

1 CONCEITOS BÁSICOS

1.1 COMO FUNCIONA SDK V2

A SDK V2 segue estrutura da imagem abaixo:



A SDK foi dividida em 4 módulos:

- Providers
- Utils
- Comandos
- Outros

1.2 PROVIDERS

Todos os providers seguem o mesmo modelo. Todos herdam de uma classe modelo que extende da classe AsyncTask do Android. Então toda escritas e leitura de comandos com o Pinpad, requisições de transações e ativações, conexões e etc, serão executadas em uma thread secundária do método 'doInBackground'.

Todas podem rodar em background ou com um simples feedback para o seu usuário (um dialog com um título e uma mensagem de sua escolha).

Após a execução do provider, ele realizará uma chamada de retorno para a sua aplicação (Callback), posteriormente iremos ver um exemplo de como isso funciona.

Métodos genéricos dos Providers:

MÉTODO	FUNÇÃO
setWorkInBackground (boolean)	Com este método você irá informar se o provider deverá ou não ser executado em segundo plano. Todos rodam em background por default.
setActivity(Activity)	Para que uma mensagem de feedback seja exibida (caso você passe **false** no método setWorkInBackground) o Android irá pedir uma activity. Desta forma, ele exibá o Dialog para o usuário com a sua mensagem.
setDialogMessage(String)	Esse método informa qual mensagem será exibida no Dialog de feedback.
setDialogTitle(String)	Esse método informa qual mensagem será exibida no título do Dialog de feedback.
setConnectionCallback(StoneCallbackInterface)	Cria uma classe anônima que possui dois métodos onSuccess (método chamado se o provider for executado com sucesso) e onError (método que será chamado se ocorrer uma Exception).
getListOfErrors()	Esse método retornará uma lista com os erros que ocorreram.

Lista de Providers disponíveis:

- ActiveApplicationProvider (Responsável por ativar a sua aplicação)
- BluetoothConnectionProvider (Responsável por criar conexão bluetooth com um pinpad a partir de um mac address)
- CancellationProvider (Responsável por realizar um cancelamento de uma transação)
- DisplayMessageProvider (Exibe uma mensagem de até 32 caracteres na tela do Pinpad)
- DownloadTablesProvider (Responsável por acessar os servidores da Stone e fazer o download das tabelas AIDs e CAPKs)
- LoadTablesProvider (Responsável por enviar as tabelas AIDs e CAPKs para os pinpads, utilize esse provedor caso a transação dê erro 20)
- PrintProvider (Responsável por realizar impressões nos Pinpads que possuem suporte para impressão)
- PrintReceipterProvider (Responsável por realizar impressões de comprovantes de vendas nos Pinpad que possum suporte para impressão)
- SendEmailProvider (Responsável por enviar uma nota eletrônica por email)
- TransactionProvider (Responsável por realizar a captura dos cartões, enviar as transações e manipular as mesmas no banco de transações do dispositivo)
- ValidateTransactionByCardProvider (Verifica se existe transação para o cartão utilizado no TransactionDAO);

1.3 UTILS

No modulo Utils, você possui ferramentas que podem lhe ajudar na criação da sua aplicação.

Com esse modulo você terá as seguintes ferramentas:

- ConnectionValidator (teste de conexão)
- Utilities (utilitários para a sua aplicação, por exemplo, nomalizador de Strings)
- GlobalInformations (informações que ficam compartilhadas entre a sua aplicação e a SDK)
- StopWatch (um simples cronômetro para benchmark)
- Criptografia e descriptografia
- ConnectionPost (realiza post para uma aplicação REST utilizando XML)
- TransactionDAO (banco de transações, as transação são salvas e atualizadas durante a execução do TransactionProvider)
- PinpadDAO (banco de pinpads já conectados)
- ApplicationCache (CRUD para a sua aplicação, ele trabalha na pasta ‘file’ da sua aplicação)

1.4 COMANDOS E OUTROS

Esse módulo contém os comandos de leitura e escrita que são utilizados para se comunicar com os Pinpads.
Os mesmos utilizados pelos Providers sempre que necessário.

2 Tutorial de integração

2.1 PREPARANDO A SUA APLICAÇÃO

Para você iniciar a integração com a Stone, existe um repositório no GitHub (https://github.com/stone-pagamentos/sdk-android-V2) no qual você pode obter o nosso app demo que te ajudará a integrar de uma forma rápida com a SDK e os JARs de integração.

No repositório, os JARs de integração estão divididos em duas partes, **/JARs/Stone/** contém um único jar com as seguintes dependências:

Para envio de email

- Mail.jar
- Activation.jar
- Additional.jar

Base64 Encoder

- Commons-codec.jar

JSON

- *Gson_v1.7.2.jar*

XML

- *XStream_v1.4.7.jar*

Nós recomendamos que os integradores utilizem esse único .jar

Os JARs que estão na pasta **/JARs/Dependências/** são todos os JARs que foram mencionados e o .jar da Stone. Você pode utilizer isso caso você já possua uma destas libs em sua aplicação.

2.2 INICIANDO A INTEGRAÇÃO

Após adicionar o .jar de integração no seu projeto, é importante que você chame o método na Main do seu projeto:
StoneStart.init([SUA MAIN ACTIVITY])

Esse método irá retornar uma lista de objetos do tipo UserModel.
UserModel representa o seu usuário, ele irá carregar as informações em cache (estas informações são utilizadas constantemente pela SDK). Caso esse método retorne null, indica que você não fez a chamada do provider ActiveApplicationProvider, como temos de exemplo o código abaixo:

```
/* Este deve ser, obrigatoriamente, o primeiro metodo
   * a ser chamado. E um metodo que trabalha com sessao.
   */
  List<UserModel> listOfUser = StoneStart.init(SUA_MAIN_ACTIVITY_AQUI);


  // se retornar nulo, voce provavelmente nao ativou a SDK
  // ou as informacoes da Stone SDK foram excluidas
  if (listOfUser == null) {

    List<String> stoneCodeToActiveList = new ArrayList<String>();
    stoneCodeToActiveList.add(STONE_PRODUCTION_KEY);

    ActiveApplicationProvider activeApplicationProvider = new ActiveApplicationProvider(SUA_MAIN_ACTIVITY_AQUI, stoneCodeToActiveList); // voce deve colocar o seu StoneCode aqui
    activeApplicationProvider.setDialogMessage("Ativando o aplicativo...");
    activeApplicationProvider.setDialogTitle("Aguarde");
    activeApplicationProvider.setWorkInBackground(false); // informa se este provider ira rodar em background ou nao
    activeApplicationProvider.setConnectionCallback(new StoneCallbackInterface() {

      /* Sempre que utilizar um provider, intancie esta Interface.
       * Ela ira lhe informar se o provider foi executado com sucesso ou nao
       */

      /* Metodo chamado se for executado sem erros */
      public void onSuccess() {
        Toast.makeText(getApplicationContext(), "Ativado com sucesso, iniciando o aplicativo", Toast.LENGTH_SHORT).show();
        continueApplication();
      }

      /* metodo chamado caso ocorra alguma excecao */
      public void onError() {
        Toast.makeText(getApplicationContext(), "Erro na ativacao do aplicativo, verifique a lista de erros do provider", Toast.LENGTH_SHORT).show();
        /* Chame o metodo abaixo para verificar a lista de erros. Para mais detalhes, leia a documentacao:
           activeApplicationProvider.getListOfErrors(); */
      }
    });
    activeApplicationProvider.execute();
  } else {

    /* caso ja tenha as informacoes da SDK e chamado o ActiveApplicationProvider anteriormente
       sua aplicacao podera seguir o fluxo normal */
    continueApplication();

  }
```

Feito isso e sua SDK estando ativada, você está pronto para iniciar suas transações.

2.3 CONEXÃO BLUETOOTH

A conexão Bluetooth é realizada também pela SDK, você deverá utilizar o BluetoothConnectionProvider para criar
conexões com os Pinpads.

Nós recomendamos que você liste todos os aparelhos que já foram conectados com o dispositivo na sua aplicação com a função “BluetoothAdapter.getBondedDevices()” do Android. Ela retornará o histórico de aparelhos do Bluetooth.

Para utilizar o BluetoothConnectionProvider, você precisará passar um parâmetro do tipo PinpadObject. Este objeto representa um Pinpad com os seguintes atributos:

TIPO	NOME	FUNÇÃO
Integer	id	ID do Pinpad quando for carregado pelo PinpadDAO
String	name	Nome do Pinpad (format: MODELO-SERIAL)
String	macAddress	Mac address do dispositivo bluetooth
boolean	printSupport	Se o Pinpad possui suporte para impressão
String	timeToConnect	Tempo que levou para criar a última conexão
int	majorDevice	O tipo do Bluetooth (headset, phone, pc…..)

Para passar criar uma conexão, basta instanciar um objeto do tipo PinpadObject e informar o nome e o mac address do dispositivo. Estas informações pode ser obtidas com os itens que são retornados da função BluetoothAdapter.getBondedDevices();” do Android.

Feito isso e conexão realizada com sucesso, o Pinpad será adicionado em uma lista de Pinpads conectados na classe GlobalInformations, para obter a lista, você pode utilizar o método GlobalInformations .getPinpadListSize(). Esse método pode te retornar null caso o Bluetooth esteja desligado ou não havendo conexão.

Sempre que a SDK solicitar um Pinpad como parâmetro, você pode passar GlobalInformations .getPinpadFromListAt(0), se você estiver conectado somente com um pinpad.

2.4 PROVEDOR DE TRANSAÇÃO

Passar transações com a nova SDK é simples e rápido. A primeira coisa que deve se fazer é instanciar um objeto do tipo StoneTransaction.

```
// passe o Pinpad que você está conectado. Apenas para lembrar, se você estiver conectado
// com apenas UM Pinpad, você deve passar "GlobalInformations.GlobalInformations.getPinpadFromListAt(0)"
StoneTransaction stoneTransaction = new StoneTransaction(O_PINPAD_QUE_VOCE_ESTA_CONECTADO);
```

Esse objeto representa a sua transação que será enviada. Ela possui os seguintes atributos:

TIPO	NOME	FUNÇÃO
String	amount	Valor da transação
String	requestId	Id do pedido
String	initiatorTransactionKey	Esse é um identificador da transação que aparece no portal. A SDK possui um padrão próprio para gerar um identificador único e usará esse padrão se um identificador não for definido. Caso você queira passar o seu próprio identificador, certifique-se que ele será sempre único para todos os seus clientes.
EmailClient	emailClient	Seu email de noReply para envio do comprovante eletrônico
InstalmentTransactionEnum	instalmentTransactionEnum	Quantidade de parcelas, mais informações na tabela abaixo
TypeOfTransactionEnum	typeOfTransactionEnum	Tipo da transação, débito ou crédito
PinpadObject	pinpadObject	O pinpad que você deseja passar a transação
String	shortName	Nome de exibição no extrato do cliente (máximo de 14 caracteres)

Para a StoneTransaction, a quantidade de parcelas é passada por um enum chamado InstalmentTransactionEnum,
segue a tabela com cada nome e valor:

NOME	VALOR REFERENTE
ONE_INSTALMENT	À vista
TWO_INSTALMENT_NO_INTEREST	2 vezes sem juros
THREE_INSTALMENT_NO_INTEREST	3 vezes sem juros
FOUR_INSTALMENT_NO_INTEREST	4 vezes sem juros
FIVE_INSTALMENT_NO_INTEREST	5 vezes sem juros
SIX_INSTALMENT_NO_INTEREST	6 vezes sem juros
SEVEN_INSTALMENT_NO_INTEREST	7 vezes sem juros
EIGHT_INSTALMENT_NO_INTEREST	8 vezes sem juros
NINE_INSTALMENT_NO_INTEREST	9 vezes sem juros
TEN_INSTALMENT_NO_INTEREST	10 vezes sem juros
ELEVEN_INSTALMENT_NO_INTEREST	11 vezes sem juros
TWELVE_INSTALMENT_NO_INTEREST	12 vezes sem juros
TWO_INSTALMENT_WITH_INTEREST	2 vezes com juros
THREE_INSTALMENT_WITH_INTEREST	3 vezes com juros
FOUR_INSTALMENT_WITH_INTEREST	4 vezes com juros
FIVE_INSTALMENT_WITH_INTEREST	5 vezes com juros
SIX_INSTALMENT_WITH_INTEREST	6 vezes com juros
SEVEN_INSTALMENT_WITH_INTEREST	7 vezes com juros
EIGHT_INSTALMENT_WITH_INTEREST	8 vezes com juros
NINE_INSTALMENT_WITH_INTEREST	9 vezes com juros
TEN_INSTALMENT_WITH_INTEREST	10 vezes com juros
ELEVEN_INSTALMENT_WITH_INTEREST	11 vezes com juros
TWELVE_INSTALMENT_WITH_INTEREST	12 vezes com juros

```
// a seguir deve-se popular o objeto
stoneTransaction.setAmount(valueEditText.getText().toString());
stoneTransaction.setEmailClient(null);
stoneTransaction.setRequestId(null);

/* AVISO IMPORTANTE: Nao e recomendado alterar o campo abaixo do
* ITK, pois ele gera um valor unico. Contudo, caso seja
* necessario, faca conforme a linha abaixo. */
stoneTransaction.setInitiatorTransactionKey("SEU_IDENTIFICADOR_UNICO_AQUI");

// informa a quantidade de parcelas
stoneTransaction.setInstalmentTransactionEnum(instalmentsSpinner.getSelectedItemPosition());

// verificacao de que forma de pagamento foi selecionada
if (debitRadioButton.isChecked() == true){
  stoneTransaction.setTypeOfTransaction(TypeOfTransactionEnum.DEBIT);
} else {
  stoneTransaction.setTypeOfTransaction(TypeOfTransactionEnum.CREDIT);
}
```

Após instanciar e popular o objeto StoneTransaction, você poderá passar o mesmo para o TransactionProvider, que será o responsável por enviar a transação para o autorizador e trabalhar com a inserção e atualização no TransactionDAO.

Lembrando que TransactionProvider segue a mesma estrutura do ActiveApplicationProvider, BluetoothConnectionProvider e os demais. Esse provider pede uma Activity e a StoneTransaction que foi criada anteriormente.

Seguindo o exemplo abaixo:

```
// processo para envio da transação
final TransactionProvider provider = new TransactionProvider([SUA_ACTIVITY_AQUI], stoneTransaction);
provider.setDialogMessage("Enviando.."); // mensagem do Dialog
provider.setDialogTitle("Aguarde"); // título do Dialog
provider.setWorkInBackground(false); // para dar um feedback para o usuário

provider.setConnectionCallback(new StoneCallbackInterface() { // cria o callback
  public void onSuccess() {
    // Transação enviada com sucesso e salva no banco. Para acessar, use o TransactionDAO
  }
  public void onError() {
    // Erro na transação
  }
});
provider.execute();
```

Após a execução do provider, a sua aplicação receberá uma chamada no método onSuccess() ou onError().

Para ter acesso ao status da transação, você deve utilizar o método getTransactionStatus() que retornará um enum com os seguintes valores:

NOME	VALOR
UNKOWN	Ocorreu um erro antes de ser enviada para o autorizador
APPROVED	Transação aprovada com sucesso
DECLINED	Transação negada
CANCELLED	Transação cancelada (ocorre no cancelamento, CancellationProvider)
PARTIAL_APPROVED	Transação foi parcialmente aprovada
TECHNICAL_ERROR	Erro técnico (ocorreu um erro ao processor a mensagem no autorizador)
REJECTED	Transação rejeitada

Em caso de **REJECTED** e **DECLINED**, você pode capturar a mensagem do autorizador com o método getMessageFromAuthorize(). Mensagens como por exemplo: “Saldo insuficiente”, “Cartão inválido”, “Violação de segurança”..

OBS: O Pinpad que será utilizado na transação, será o que foi passado como parâmetro na instancia do StoneTransaction.

2.5 PROVEDOR DE IMPRESSÃO

O provedor de impressão, PrintProvider, pode ser utilizado apenas com pinpads que possuem suporte a print.

Na assinatura do construtor, o PrintProvider irá pedir três parâmetros, uma Activity, uma lista de PrintObject e o Pinpad que irá imprimir o seu comprovante.

PrintObject representa cada linha que será impressa pelo PrintProvider.

TIPO	NOME	FUNÇÃO
String	message	Mensagem que será impressa (por linha)
Integer	size	Tamanho da impressão, SMALL, MEDIUM ou BIG, todos dentro do objeto PrintObject
Integer	align	Alinhamento da impressão, LEFT, CENTER ou RIGHT, todos dentro do objeto PrintObject

OBS: Para impressão de QR Code, é necessário passar ‘TAG_PRINT’ como size e align. Sua mensagem que será enviada dentro do QR Code deve conter no máximo 512 caracteres.

```
// logica da impressão
// declara a lista de impressão,
// lembrando que cada item representa
List<PrintObject> listToPrint = new ArrayList<PrintObject>();
for (int i = 0; i < 10; i++) {
  listToPrint.add(new PrintObject("Teste de impressão linha " + i, PrintObject.MEDIUM, PrintObject.CENTER));
}

final PrintProvider printProvider = new PrintProvider(SUA_ACTIVITY_AQUI, listToPrint, O_PINPAD_QUE_VOCE_DESEJA_UTILIZAR);
printProvider.setWorkInBackground(false); // não rodar em background
printProvider.setDialogMessage("Imprimindo..."); // mensagem do Dialog
printProvider.setConnectionCallback(new StoneCallbackInterface() {
  public void onSuccess() {
    // notinha impressa com sucesso
  }

  public void onError() {
    // ocorreu um erro na impressão
  }
});
printProvider.execute();
```

Com o código que tivemos de exemplo anteriormente, nós teremos o seguinte comprovante:

A SDK irá interpreter cada item da sua lista (cada PrintObject) como uma linha a ser impressa.

Com o código que foi passado na imagem anterior, ele rodará um for e criar 10 linhas de impressão. O provider será executado e terá o seguinte resultado:

Por padrão, todas as notinhas são impressas com a logo da Stone.

2.6 PROVEDOR DE IMPRESSÃO – COMPROVANTE DE TRANSAÇÃO (RECIBO)

O provedor de impressão, PrintReceiptProvider, pode ser utilizado apenas com pinpads que possuem suporte a print.

Na assinatura do construtor, o PrintProvider irá pedir quatro parâmetros, um Context, um PinpadObject (Pinpad que irá imprimir o seu comprovante), o ID da transação que deseja imprimir e um UserModel que representa o seu usuário, que contém as informações necessárias para a impressão.

```
// GlobalInformations.getPinpadFromListAt(0) e o pinpad conectado, que esta na posicao zero.
// GlobalInformations.getUserModel(0) e o usuario atual do aplicativo, que esta na posicao zero.
// idTransaction e o id (int) da transacao que se deseja impimir e que esta salvo no TransactionDAO
final PrintReceiptProvider printReceiptProvider = new PrintReceiptProvider(SEU_CONTEXT_AQUI, GlobalInformations.getPinpadFromListAt(0),
                                                                            idTransaction, GlobalInformations.getUserModel(0));
printReceiptProvider.setWorkInBackground(false); // não rodar en background
printReceiptProvider.setDialogMessage("Imprimindo..."); // mensagem do Dialog
printReceiptProvider.setConnectionCallback(new StoneCallbackInterface() {
  public void onSuccess() {
     // notinha impressa com sucesso
  }
  public void onError() {
     // ocorreu um erro na impressão
  }
});
printReceiptProvider.execute();
```

A SDK montará todo o comprovante automaticamente (de acordo com o id de transação fornecido e contato que o transação exista no TransactionDAO).

Por padrão, todas as notinhas são impressas com a logo da Stone.

2.7 PROVEDOR DE CANCELAMENTO

O provider de Cancelamento, CancellationProvider, é um dos provedores mais simples.
Para efetuar um cancelamento, você deve instanciar o CancellationProvider e passar sua Activity e o ID da transação que deseja passar (você pode obter todos os IDs das transações com o TransactionDAO).
OBS: Recomendamos que use o provedor ValidateTransactionByCardProvider (**Tópico 2.11)** para validar se a transação que será cancelada foi realizada com o mesmo cartão que consta no TransactionDAO.

2.8 PROVEDOR DE DISPLAY

DisplayMessageProvider exibe uma mensagem de até 32 caracteres a tela do Pinpad.

2.9 PROVEDOR DE EMAIL

SendEmailProvider é o provedor responsável pelo envio do email. Ele serve apenas para enviar um email, não o comprovante eletrônico da sua transação.

Esse provider pedirá um objeto do tipo EmailClient, esse objeto represente o seu client de noReply SMTP.
Configurando ele de forma correta, o provedor funcioná normalmente.

TIPO	NOME	FUNÇÃO
String	smtpHost	Host do seu cliente de email
String	smtpPort	Porta do host (default: 587)
String	sport	Secure port do host (default: 587)
String	user	Seu usuário de login
String	password	Sua senha de login
String	from	Seu email
String	sendTo	Email do seu cliente
String	subject	Assunto do email
String	messageToSent	Mensagem que será enviada

OBS: Por questões de esclarecimentos, a Stone não faz nenhum log dos e-mails enviados. Ou seja, seu login e senha utilizados para enviar o e-mail, não serão salvos em momento nenhum por parte da SDK.

Para criar o comprovante eletrônico, o seu objeto de EmailClient possui um método chamado receiptAsHtml(Context, transactionId). Ele retornará uma String e a mesma deve ser passada como parâmetro no SendEmailProvider como mensagem para enviar.

2.10 PROVEDOR DE TABELAS AID E CAPK E CARGA DE TABELAS PARA OS PINPADS

O DownloadTablesProvider é o provedor de tabelas, isso é **muito** importante de ser chamado sempre na Main da sua aplicação. As tabelas AID e CAPK, resumidamente falando, são tabelas que ficam armazenadas nos Pinpads e os dizem como trabalhar com cada tipo de cartão. Esse provider só possui a função de acessar os serviços da Stone e baixa-las para o dispositivo.

Utilize um objeto do tipo ApplicationCache (módulo Utils) para verificar se existem ou não as tabelas. Segue o exemplo:

```
// IMPORTANTE: Mantenha esse provider na sua main, pois
// ele ira baixar as tabelas AIDs e CAPKs dos servidores
// da Stone e sera utilizada quando necessario.
ApplicationCache applicationCache = new ApplicationCache(getApplicationContext());
if (applicationCache.checkIfHasTables() == false) {

  // realiza todo processo de download das tabelas
  DownloadTablesProvider downloadTablesProvider = new DownloadTablesProvider(SUA_ACTIVITY_AQUI);
  downloadTablesProvider.setDialogMessage("Baixando as tabelas, por favor aguarde");
  downloadTablesProvider.setWorkInBackground(false); // para dar feedback ao usuario ou nao.
  downloadTablesProvider.setConnectionCallback(new StoneCallbackInterface() { // chamada de retorno.
    public void onSuccess() {
      // tabelas baixadas com sucesso
    }
    public void onError() {
      // erro no download das tabelas
    }
  });
  downloadTablesProvider.execute();
}
```

É importante deixar claro também que esse provider só possui a função de baixar as tabelas. A função de carregar as tabelas nos Pinpads é responsabilidade do LoadTablesProvider. Este deve ser chamado, caso o TransactionProvider retorne um erro de **DIFFERENT_TABLES_EMV** (veremos logo o que isso quer dizer).

2.11 PROVEDOR DE VALIDAÇÃO DE TRANSAÇÃO

O ValidateTransactionByCardProvider é o provedor de que verifica se existe no TransactionDAO, transações realizadas para um determinado cartão.

Na assinatura do construtor, o ValidateTransactionByCardProvider irá pedir dois parâmetros, um Context, um PinpadObject (Pinpad em que será inserido o cartão).

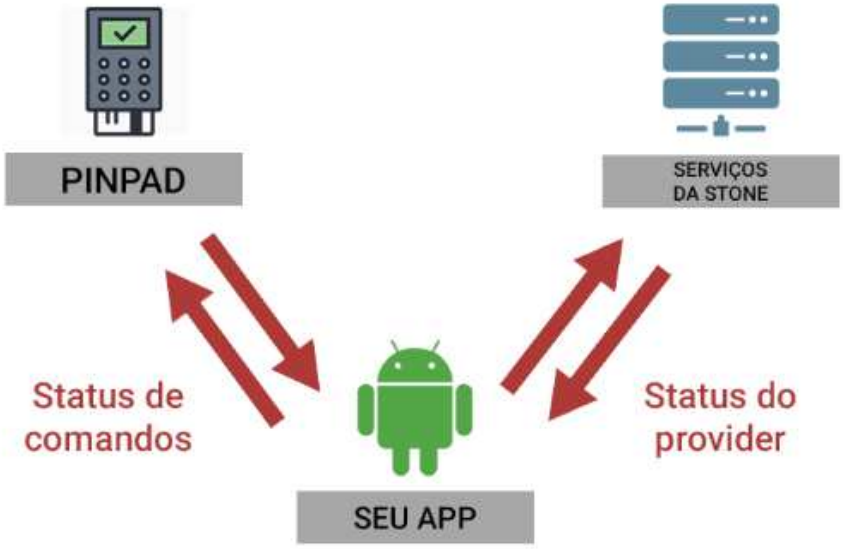
```
final ValidateTransactionByCardProvider validateTransactionByCardProvider = new ValidateTransactionByCardProvider(SUA_ACTIVITY_AQUI, O_PINPAD_QUE_VOCE_DESEJA_UTILIZAR);
validateTransactionByCardProvider.setDialogMessage("Validando transação");
validateTransactionByCardProvider.setWorkInBackground(false);
validateTransactionByCardProvider.setDialogTitle("Aguarde");
validateTransactionByCardProvider.setConnectionCallback(new StoneCallbackInterface() {
  public void onSuccess() {
    // para saber se existem transações com esse cartão:
     List<TransactionObject> transactionsWithThisCard = validateTransactionByCardProvider.getTransactionsWithCurrentCard();

  }

  public void onError() {
    // erro na execução do provider
  }
});
validateTransactionByCardProvider.execute();
```

3 Tipos de response

3.1 UM POUCO SOBRE RESPONSE

Quando nos referimos a “Response”, não estamos falando exatamente de response de algum request para os servidores da Stone. Vamos dividir o assunto entre responses da biblioteca compartilhada e responses da SDK.

Observando a seguinte imagem:



Os Status de comandos são referentes as respostas que são dadas pela biblioteca compartilhada, mais abaixo você verá uma lista com as possíveis respostas e o que cada uma significa.

Os Status do provider são referentes aos erros que podem ter durante a execução de determinados providers.

3.2 STATUS DE COMANDOS

NOMES	VALOR	SIGNIFICADOS
COMMAND_OK	0	Comando executado com sucesso
NO_SECURE_COMMUNICATION	3	Não foi estabelecida uma conexão segura com o Pinpad
PRESS_1	4	Pressione F1
PRESS_2	5	Pressione F2
PRESS_3	6	Pressione F3
PRESS_4	7	Pressione F4
PRESS_BACKSPACE	8	Pressione espaço
DEC_ERROR	9	Erro de decodificação
INVALID_CALL	10	Chamada inválida
INVALID_PARAMETER	11	Parâmetro inválido
TIME_OUT	12	Time out
OPERATION_CANCELLED_BY_USER	13	Operação cancelada pelo usuário
NOT_OPEN	15	Comando OPN não enviado
PARAMETER_NOT_FOUND	19	Falta de um parâmetro mandatório
DIFFERENT_TABLES_EMV	20	Tabelas AID e CAPK diferentes
TABLES_UPLOAD_ERROR	21	Erro na carga de tabelas
PINPAD_INTERNAL_ERROR	40	Erro interno do Pinpad
MAGNETIC_CARD_ERROR	41	Erro na leitura da tarja
MK_DUKPT_NOT_FOUND	42	DUKPT não encontrada
NO_ICC_PRESENCE	43	Não há presença de ICC
PINPAD_CANT_PROCESS_PIN_CAPTURE	44	O pinpad não conseguiu processar a captura da senha
RESPONSE_DATA_LIMIT_EXCEEDED	45	Resposta excedeu o limite de informações
SAM_NOT_FOUND	51	SAM não encontrado
ICC_NOT_FOUND	60	ICC não encontrado
COMMUNICATION_ICC_OR_CTLS_ERROR	61	Erro de comunicação ICC ou CTLS
INVALID_CARD	67	Cartão inválido
ICC_WITH_PROBLEM	68	Problemas no ICC
INVALID_DATAS_IN_ICC	69	Informações inválidas no ICC

ICC_EMV_WITHOUT_AVAILABLE_APPLICATION	70	ICC sem aplicação disponível
SELECTED_APPLICATION_CANT_BE_USED	71	Aplicação selecionada não pode ser usada
HIGH_ICC_EMV_ERROR	76	Alto erro no ICC
MORE_THEN_ONE_CTLS_DETECTED	80	Mais de um CTLS detectado
COMMUNICATION_BETWEEN_TERMINAL_CTLS_ERROR	81	Erro na comunicação com CTLS e o Pinpad
INVALID_CTLS	82	CTLS inválido
CTLS_WITH_PROBLEM	83	CTLS com problema
CTLS_WITHOUT_APPLICATION_AVAILABLE	84	CTLS sem aplicação disponível
APPLICATION_SELECTED_CANT_BE_USED	85	Aplicação selecionada não pode ser usada
FILE_NOT_FOUND	100	Arquivo não encontrado
FILE_FORMAT_ERROR	101	Erro no formato do arquivo
FILE_UPLOAD_ERROR	102	Erro na carga do arquivo
COMMAND_CONSTRUCTOR_ERROR	999	Erro no construtor da resposta

3.3 STATUS DO PROVIDER

NOME	SOLUÇÃO
CONNECTION_NOT_FOUND	Sem conexão com a internet, ligue a internet
DEVICE_NOT_COMPATIBLE	Dispositivo bluetooth não possui a biblioteca compartilhada
UNEXPECTED_STATUS_COMMAND	Status de um comando inexperado, tente novamente
GENERIC_ERROR	Um erro genérico
IO_ERROR_WITH_PINPAD	Erro de leitura e escrita com o Pinpad
PINPAD_CONNECTION_NOT_FOUND	Sem conexão com um Pinpad, conecte-se com um dispositivo
NO_ACTIVE_APPLICATION	SDK não foi ativada. Chame o ActiveApplicationProvider
TABLES_NOT_FOUND	Tabelas AID e CAPK não encontradas, chame o DownloadTablesProvider
UNKNOWN_TYPE_OF_USER	Um tipo de usuário desconhecido, Usuário final ou Desenvolvedor (User ou Partner)
INVALID_AMOUNT	Valor inválido para passar a transação
NEED_LOAD_TABLES	Chame o LoadTablesProvider para realizar a carga de tabelas
TIME_OUT	Time out, tente novamente

OPERATION_CANCELLED_BY_USER	Operação cancelada pelo usuário, provavelmente o botão X foi pressionado
CARD_REMOVED_BY_USER	Cartão removido pelo usuário indevidamente
CANT_READ_CARD_HOLDER_INFORMATION	Erro na leitura das informações do cartão, tente novamente
INVALID_TRANSACTION	Transação inválida, talvez um cartão de crédito esteja passando uma transação de débito, ou vice-versa.
PASS_TARGE_WITH_CHIP	Cartão de chip passou tarja
NULL_RESPONSE	Não houve resposta do Pinpad
ERROR_RESPONSE_COMMAND	Erro na resposta do comando
ACCEPTOR_REJECTION	Transação rejeitada pelo autorizador
EMAIL_MESSAGE_ERROR	Erro no envio do email
INVALID_EMAIL_CLIENT	Email do cliente é inválido
INVALID_STONE_CODE_OR_UNKNOWN	StoneCode foi digitado de forma errada ou não existe
TRANSACTION_NOT_FOUND	Não se pode cancelar uma transação que não foi aprovada

4 Usuários e Desenvolvedores

4.1 PARA DESENVOLVEDORES

Umas das novidades da SDK 2.0 é que agora quando você chama a StoneStart.init no início do seu projeto e iniciar o projeto como parceiro (Partner), esse é um perfil que roda na sessão da SDK. Isso permite que ao passar uma transação ou executar um cancelamento, aponte para os ambientes de testes da Stone (a troca de endpoint é feita dentro da própria SDK, o desenvolvedor parceiro não precisa se preocupar em trocar os endpoints).

Como trocar de Usuário final para modo desenvolvedor?
Segue como mudar para o modo desenvolvedor:

```
// no início do seu projeto
UserModel userModel = StoneStart.init(SUA_ACTIVITY_AQUI);

// diz que é um desenvolvedor com essas duas linhas de código
Partner partnerDeveloper = new Partner(userModel);
GlobalInformations.sessionApplication.setUserModel(partnerDeveloper);
```

IMPORTANTE: Não esqueça de colocar este código APENAS na sua build de debug, NUNCA suba o seu código com esse modo em produção.