TrustPay API

Version 1.9

2013-06-21

Terminology

"Client Application" - The application installed on the handset that wishes to make use of the TrustPay billing services.

"TrustPay API" - A library that is linked to the Client Application and which provided facilities to interact with the TrustPay Backend

"TrustPay Backend" - The server infrastructure taking care of operations, billing and administration around the TrustPay service.

Theory of Operation

When the client app wishes to bill the consumer, the request is passed on to the TrustPay API library. This library then communicates with the backend to present a screen with all available billing methods for the amount. The consumer selects a method. A method-specific screen gets displayed to the consumer to supply information and/or to validate the request. If the consumer accepts, the billing takes place and the result is returned to the client application.

J2ME API

There are three separate mechanisms for communicating with the J2ME TrustPay app and backend, depending on the purpose of the communication.

Firstly, a silent API-call is used to interrogate it to find out what options are available, without requiring consumer interaction.

Some J2ME devices allows the developer to programmaticaly determine the Mobile Operator of the mobile device. This is required for Operator billing. If the device do not allow us to do this, the developer will be informed of the lack of information and then has the choice to display a Country/Operator screen via the TrustPayApi. Thus the developer is in control of the user experience at all times.

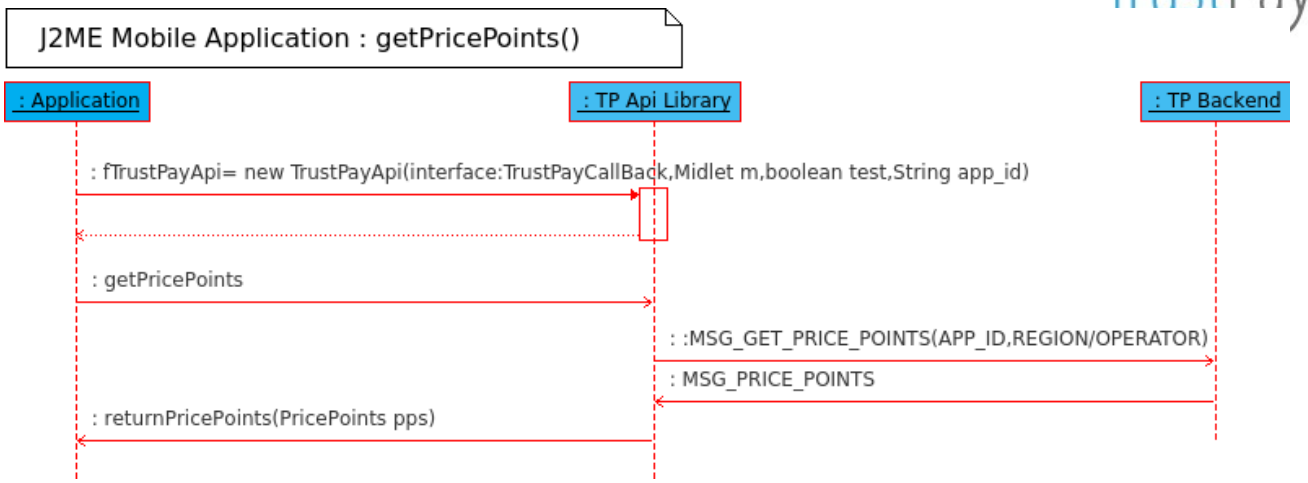
Secondly, a Screen is launched for effect, to allow the consumer to interact with the TrustPay library to select a payment method and to actually bill.

Thirdly, a silent API-call is used to instruct the TrustPay Application to refresh all cached billing methods in the background.

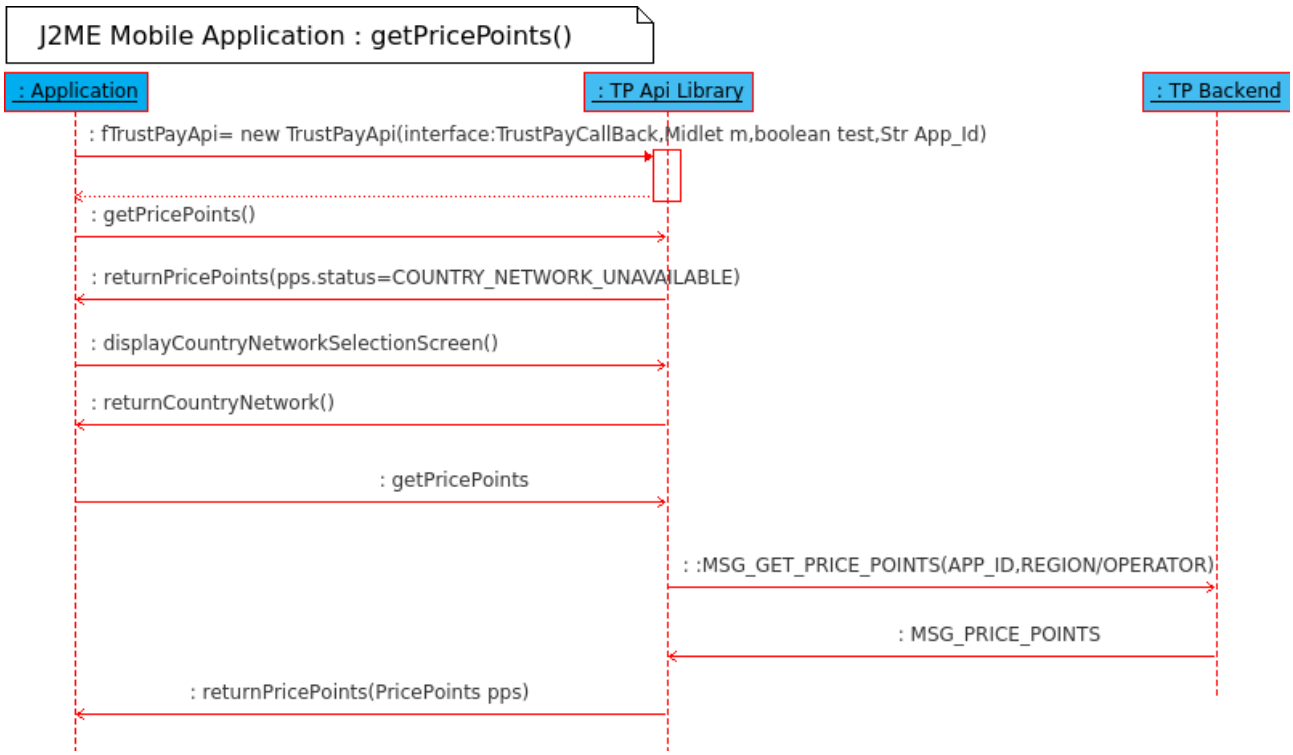
The overall process in the client app is managed in a Class – normally a Screen, but can be otherwise, which provides an entry point for the callbacks from the TrustPay app. The typical scenarios are presented below.

Requesting Price Points

This function is used for the Client App to get the price points supported by TrustPay for the current handset territory and network.



If the Country and Operator network cannot be obtained programmatically:



Code fragments from ApiDemo

```
/** class with access to TrustPayApi must implement TrustPayCallback
interface */

public class myScreen extends Form implements TrustPayCallback {
        ...
```

```
}

/** Called before Api can be used. We have to pass a handle to the midlet in here*/
public void …. {
   ….

   fTrustPayApi = new TrustPayApi(this, midlet, false, CONST_APP_ID);
}


public void ... {

        fTrustPayApi.getPricePoints();

 }


 public void returnPricePoints(PricePoints pps) {

        // Do something with the returned PricePoints structure
            If (pps.status == PricePoints.COUNTRY_NETWORK_UNAVAILABLE) {
            // This will cause the API to open a Country/Network selection screen
                fTrustPayApi.displayCountryNetworkSelectionScreen();
            } else {
            // Do something with the returned PricePoints

            }

 }

 // When the Country/Network selection has been obtained, this method gets called.
 public void returnCountryNetwork() {
        // getPricePoints can be called again
        fTrustPayApi.getPricePoints();
 }
```

The returned PricePoints object has the following format

```
public class PricePoints  {
        public int status;
        public PricePoint[] fPricePoints;
….
}
```

status can have the values:

PricePoints.NONE
PricePoints.SUCCESS
PricePoints.COUNTRY_NETWORK_UNAVAILABLE

Each PricePoint has

```
String fCurrency
String fLower
String fUpper
String fStep
```

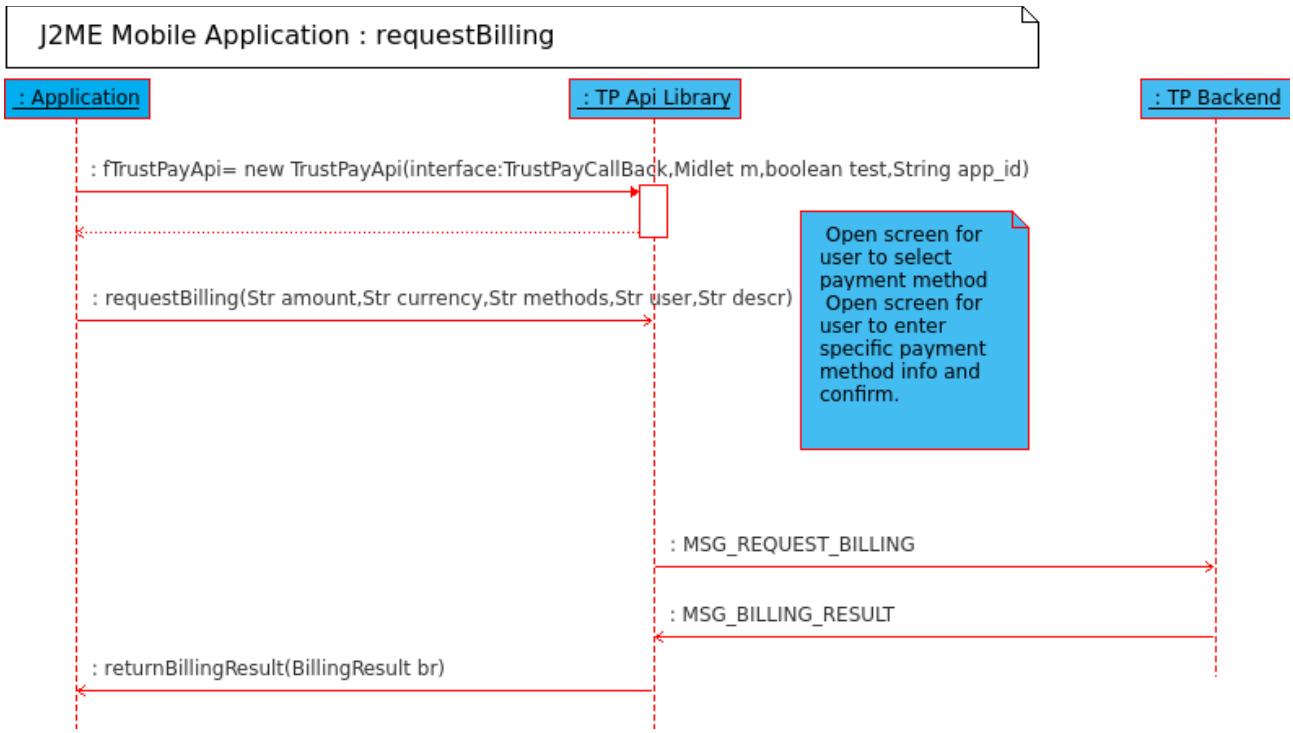
String fType

where: fCurrency is always populated with the currency code :
e.g. fCurrency = "ZAR" for South-African rands

fLower is the lower value of the pricepoint range
fUpper is the upper value of the pricepoint range
fStep is the increment values between lower and upper
e.g.: for all full rand values between 20 and 99, the values will be:
fLower = "20.00" fUpper = "99.00" fStep = "1.00"
for all cent values between 1 and 100 rand, the values will be:
fLower = "1.00" fUpper ="100.00" fStep = "0.01"
for a single pricepoint of 5 rand, the values will be:
fLower = "5.00" fUpper = "5.00" fStep = "0.00"
fType is the transaction type, values currently supported are:
CARRIER BILLING
VOUCHER
CREDIT CARD
WALLET

Request Payment

This function is used to actually bill the consumer. It does this by presenting a screen where the consumer must select a billing method. After selecting the method, another screen will be displayed where the consumer may enter some information and displaying information explaining the amount to be billed and the name of the application requesting it. The consumer can either agree to bill, or cancel the transaction. The status is returned to the calling app.



Code fragments from ApiDemo for Payment

/** class with access to TrustPayApi must implement TrustPayCallback


```
interface */

public class myScreen extends Form implements TrustPayCallback {
      ...

}


   /** Called before Api can be used. We have to pass a handle to the midlet in here*/
   public void …. {
      ….

      fTrustPayApi = new TrustPayApi(this, midlet, false, CONST_APP_ID);
   }
```

/* The third parameter in the fTrustPayApi. requestBilling call is the payment methods that you want to support the [*] wildcard means 'All methods supported for my transaction'
If the pricepoints returned indicated that you can do Credit Card, Debit Card and Carrier billing, and you only want to support Credit and Debit Card, you pass the parameter as: [CC;DC] and only those two options will be displayed to the user. (See PricePoint description earlier for Payment Method codes.
app_user, is a paramater that will identify the specific mobile user,
tx_Description is a human-readble decription of the transaction for reporting purposes,
app_reference is a transaction number assigned by the developer.*/

```
        public void pay(String amount,String currency) {

                fTrustPayApi.requestBilling(amount, currency,"[*]", app_user, tx_Description,
app_reference);

        }

        // This method is called when the billing is done
        public void returnBillingResult(BillingResult br) {
                // Do something with the returned BillingResult


       }
   }
```

The BillingResult object has the following format

```
        public class BillingResult {

           // Request values
           public String fAppid     = null;
           public String fOperation = null;
           public String fAmount    = null;
           public String fCurrency  = null;
           public String fAppref    = null;
           public Boolean fIsTest   = null;

           // Returned values
           public String fTrustPayRef = null;
           public String fMessage = null;
           public boolean fSuccess = false;
```


```
....
}
```

The important field is fSuccess, which will be true if the billing succeeded. fMessage provides more information on the kind of problem that occurred. Possible values are listed in the table below:

Value	Interpretation
INVALID_OPERATION	The billing intent was delivered with an invalid operation. If the API is used, this should never occur.
Test success	A successful simulated billing occurred. In other words, the consumer clicked on accept when asked to pay, but no payment was actually made.
Cancelled by user	The consumer chose not to pay
Success	Successful billing occurred. In other words, the consumer clicked on accept when asked to pay, and payment was made.
Error	A generic error occurred that prevented billing

Refresh Payment Methods

This function is used to actually refresh the Payment Methods and Price Points on the device by doing a delta synchronization with the TrustPay backend environment.

Code fragments from ApiDemo

```
/** class with access to TrustPayApi must implement TrustPayCallback
interface */

public class myScreen extends Form implements TrustPayCallback {
      ...
      }
 /** Called before Api can be used. We have to pass a handle to the midlet in here*/
public void .... {
    ....
    fTrustPayApi = new TrustPayApi(this, midlet, false, CONST_APP_ID);
}
```

.....
The Refresh Payment Methods can be done by calling :

```
fTrustPayApi.updateBillingMethods();
```

When done, the result gets returned in

```
public void onUpdateMethods(Boolean arg0) {

}
```

where arg0 is true when successful or false on a failure.

Changes

Date	Version	Changes
2013-06-21	1.9	New PaymentMethods
2013-01-04	1.8	Added app_reference to the API.
2012-12-14	1.7	Added caching of Paymentmethods and updating thereof.
2012-11-06	1.6	Changed doc to only support single application solution.
2012-10-26	1.5	Implemented Payment Method selection, Appuser Transaction Description params in the preparePayIntent method.
2012-08-24	1.4	New pricepoint format supporting all payment methods
2012-08-01	1.3	Added Network/Country Selection
2012-07-30	1.2	Created J2ME specific version
2012-04-25	1.1	Added fValueNett field to the PricePoints structure, to give an indication of nett payout for the pricepoint
2012-04-10	1.0	Initial Version