TrustPay API

Version 1.6

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 Application

"TrustPay Application" - The application installed on the handset to mediate between the client application and the TrustPay backend.

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

Theory of Operation

The TrustPay application acts as an intermediary between the client application and the consumer who wants to pay for something.

This allows the client app to be installed with very few permissions, increasing the likelyhood that a consumer will install it.

When the client app wishes to bill the consumer, the request is passed on to the TrustPay app. This app 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.

Blackberry API

There are three separate mechanisms for communicating with the Blackberry 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.

Secondly, a Screen is launched for effect, to allow the consumer to interact with the TrustPay app 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 if Auto-Refresh is not enabled. If Auto-Refresh is enabled, this will happen automatically every time the client application binds to TrustPay.

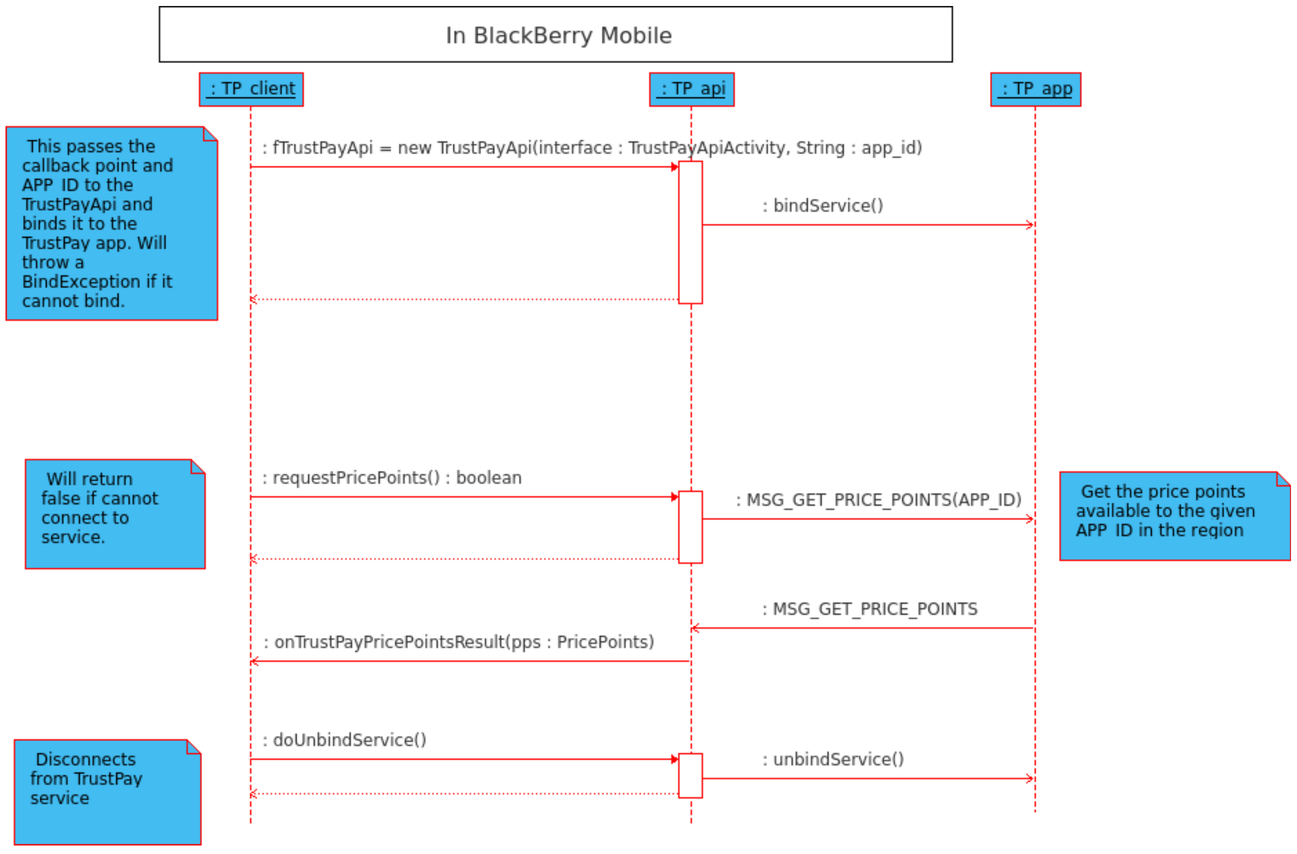
The overall process in the client app is managed in a Class – normally a Screen, but can be otherwise, which provides an entry point for the callbacks from the TrustPay app. The typical scenarios are presented below.

If the Client Application cannot bind to the TrustPay Application, the chance is that the TrustPay Application has not been installed yet. The API exposes a method to Download the appropriate TrustPay Application in this case.

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.


In BlackBerry Mobile
: TP client
: TP api
: TP app
This passes the callback point and APP ID to the TrustPayApi and binds it to the TrustPay app. Will throw a BindException if it cannot bind.
: fTrustPayApi = new TrustPayApi(interface : TrustPayApiActivity, String : app_id)
: bindService()
Will return false if cannot connect to service.
: requestPricePoints() : boolean
: MSG_GET_PRICE_POINTS(APP_ID)
Get the price points available to the given APP ID in the region
: MSG_GET_PRICE_POINTS
: onTrustPayPricePointsResult(pps : PricePoints)
Disconnects from TrustPay service
: doUnbindService()
: unbindService()

Code fragments from ApiDemo

```
 /** class with access to TrustPayApi must implement TrustPayApiActivity */

 public class myScreen extends MainScreen implements TrustPayApiActivity {
       ...

  }

  /** Called before Api can be used. */
  public void …. {
     ….
```

```
        try     {
                        fTrustPayApi = new TrustPayApi(this,CONST_APP_ID,true);
        // Third param is true for Auto-Refresh of Billing methods, false if not required.
                } catch (BindException ex) {
                // Could not bind to TrustPayApi,might not be installed, try to download
                        TrustPayApi.DownloadTrustPay();
                }
    }



    /** Called to disconnect from the Api */
    public void ... {
       fTrustPayApi.doUnbindService();
    }

    public boolean getPricePoints(boolean isTest) {

       fTrustPayApi.setTest(isTest);
       return fTrustPayApi.requestPricePoints();

    }


    public void onTrustPayPricePointsResult(PricePoints pps) {
       if (pps != null) {
          // Do something with the returned PricePoints structure
       }
    }
```

The returned PricePoints object has the following format

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

Each PricePoint has

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

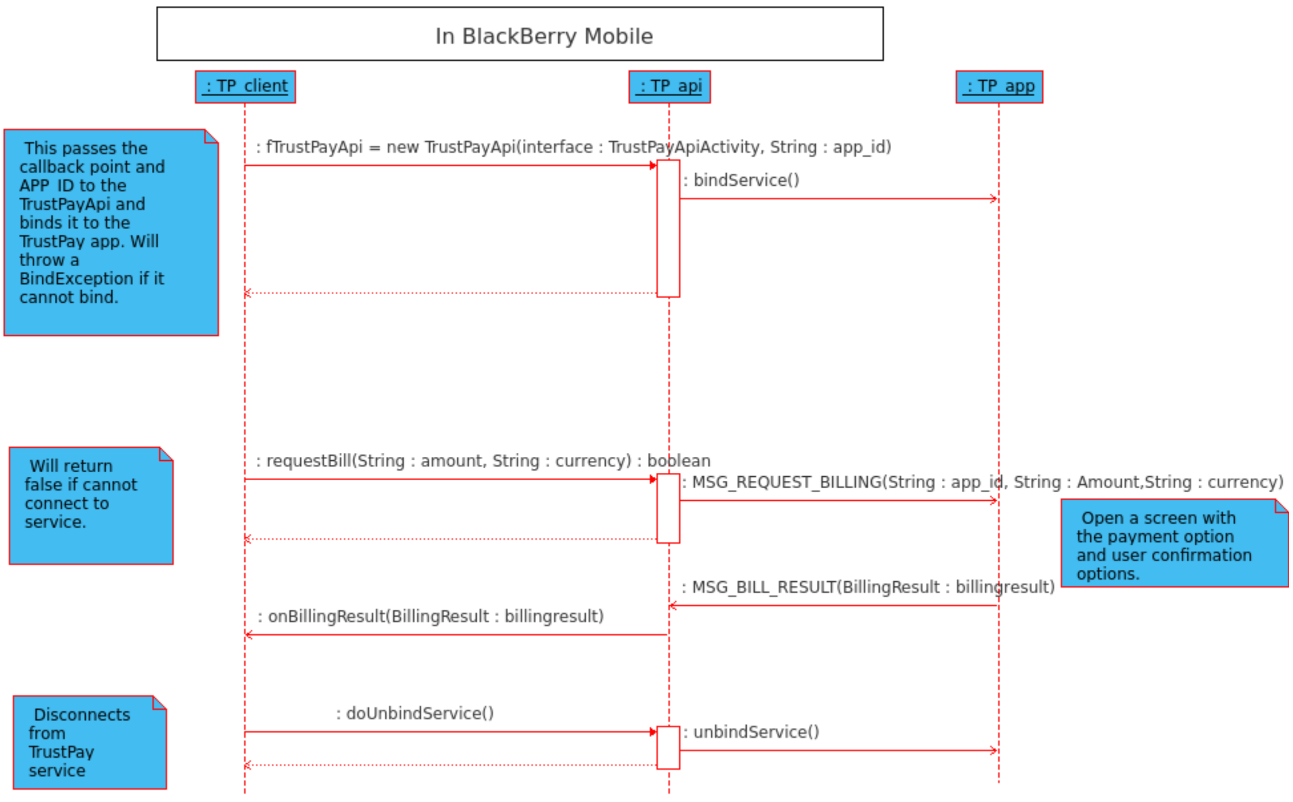
where: fCurrency is always populated with the currency code :
e.g. fCurrency = "ZAR" for South-African rands

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.


In BlackBerry Mobile
: TP client
: TP api
: TP app
This passes the callback point and APP ID to the TrustPayApi and binds it to the TrustPay app. Will throw a BindException if it cannot bind.
: fTrustPayApi = new TrustPayApi(interface : TrustPayApiActivity, String : app_id)
: bindService()
Will return false if cannot connect to service.
: requestBill(String : amount, String : currency) : boolean
: MSG_REQUEST_BILLING(String : app_id, String : Amount,String : currency)
Open a screen with the payment option and user confirmation options.
: MSG_BILL_RESULT(BillingResult : billingresult)
: onBillingResult(BillingResult : billingresult)
Disconnects from TrustPay service
: doUnbindService()
: unbindService()

Code fragments from ApiDemo for Payment

```
/** class with access to TrustPayApi must implement TrustPayApiActivity */

  public class myScreen extends MainScreen implements TrustPayApiActivity {
        ...

   /** Called before Api can be used.  */
```

```
    public void …. {
       ….
       try       {
                        fTrustPayApi = new TrustPayApi(this,CONST_APP_ID,true);
     // Third param is true for Auto-Refresh of Billing methods, false if not required.
                } catch (BindException ex) {
          // Could not bind to TrustPayApi,might not be installed, try to download
                        TrustPayApi.DownloadTrustPay();
                }
    }




    /** Called to disconnect from the Api */
        public void ... {

                fTrustPayApi.doUnbindService();

        }
```

/* 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,boolean isTest) {
                fTrustPayApi.setTest(isTest);
                if (!fTrustPayApi.requestBilling(amount, currency,"[*]", app_user, tx_Description,
app_reference);) {
                        response.setText("Cannot find TrustPay Application");
                }
        }

        // This method is called when the billing is done
        public void onBillingResult(BillingResult br) {

       }
    }
```

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
Error: No service	The handset is currently not connected to a mobile network
Error: Radio off	The handset is currently not connected to a mobile network, because the radio is off in e.g. Flight Mode

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 TrustPayApiActivity */

  public class myScreen extends MainScreen implements TrustPayApiActivity {
        ...

   /** Called before Api can be used. */
   public void …. {
      ….
      try       {
                        fTrustPayApi = new TrustPayApi(this,CONST_APP_ID,true);
    // Third param is true for Auto-Refresh of Billing methods, false if not required.
                } catch (BindException ex) {
          // Could not bind to TrustPayApi,might not be installed, try to download
                        TrustPayApi.DownloadTrustPay();
                }
   }

…..
```

The Refresh Payment Methods can be done by calling :

```
fTrustPayApi.UpdateBillingMethods();
```

When done, the result gets returned in

```
@Override
public void onUpdateMethods(Boolean arg0) {

}
```

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

Changes

Date	Version	Changes
2013-06-21	1.6	New Payment Methods
2013-01-04	1.5	Added App_Reference_number into API for making payment.
2012-12-14	1.4	Implemented caching of PaymentMethods and refreshing thereof.
2012-08-24	1.3	New pricepoint format supporting all payment methods
2012-07-30	1.2	Created BlackBerry 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