



安全支付服务 Symbian_S60_v3 应用开发指南

文件版本：3.0.1.0

支付宝（中国）网络技术有限公司版权所有

2012-04-18

版权信息

本手册中所有的信息为支付宝公司提供。未经过支付宝公司书面同意，接收本手册的人不能复制，公开，泄露手册的部分或全部的内容。

前言

1. 面向读者

本文档主要面向需要接入支付宝安全支付的商户的开发人员。

2. 读者所需技能

读者需有基本的程序开发背景，掌握 c++ 及 symbian 程序开发。

3. 开发环境要求

OS: WinXP or win7

SDK: S60 3rd Edition, S60 3rd Edition FP1, S60 3rd Edition FP2, S60 5th Edition

IDE: Carbide

建议: 为保证兼容性 S60 V3 版本的程序推荐使用 S60 3rd Edition(9.1)版本的 SDK 开发;

S60 V5 及 symbian^3 版本的程序推荐使用 S60 5th Edition 版本的 SDK 开发

目录

第一章 安全支付服务简介	4
1.1 安全支付服务介绍	4
1.2 安全支付服务业务流程	5
1.3 调用安全支付数据流程图	5
第二章 安全支付接入流程	6
2.1 接入前期准备	6
2.1.1 商户签约	6
2.1.2 密钥配置	6
2.2 Demo	7
2.2.1 Demo 配置运行	7
2.2.2 Demo 结构说明	12
2.3 安全支付集成	15
2.4 应用发布	17
第三章 RSA 详解	18
3.1 RSA 和 OpenSSL 介绍	18
3.1.1 什么是 RSA	18
3.1.2 为什么要用 RSA	18
3.1.3 什么是 OpenSSL	18
3.1.4 为什么要用 OpenSSL	19
3.2 RSA 密钥详解 *	19
3.2.1 找到生成 RSA 密钥工具	19
3.2.2 生成商户密钥并获取支付宝公钥	19
3.3 RSA 签名和验签 *	22
3.3.1 RSA 签名	22
3.3.2 RSA 验签	23
第四章 通知结果	24
4.1 AlixPay 方法返回的结果	24
4.2 notify_url 通知说明	25
4.2.1 什么是 Notify_url	25
4.2.2 Notify_url 接收数据示例	26
第五章 常见问答	27
附录 A 错误代码列表	27
附录 B 安全支付服务接口	28
1 安全支付服务接口列表	28
2 AlixPay 方法描述	28
3 订单信息描述	29

第一章 安全支付服务简介

1.1 安全支付服务介绍

S60 安全支付服务主要用来向第三方应用程序提供便捷、安全以及可靠的支付服务。安全支付服务使用二种平台架构（ECom/CS）实现，保证了在不同场合下商户接入场景。本文主要描述安全支付服务应用开发接口的使用方法，供合作伙伴的开发者接入使用。

1.2 安全支付服务业务流程

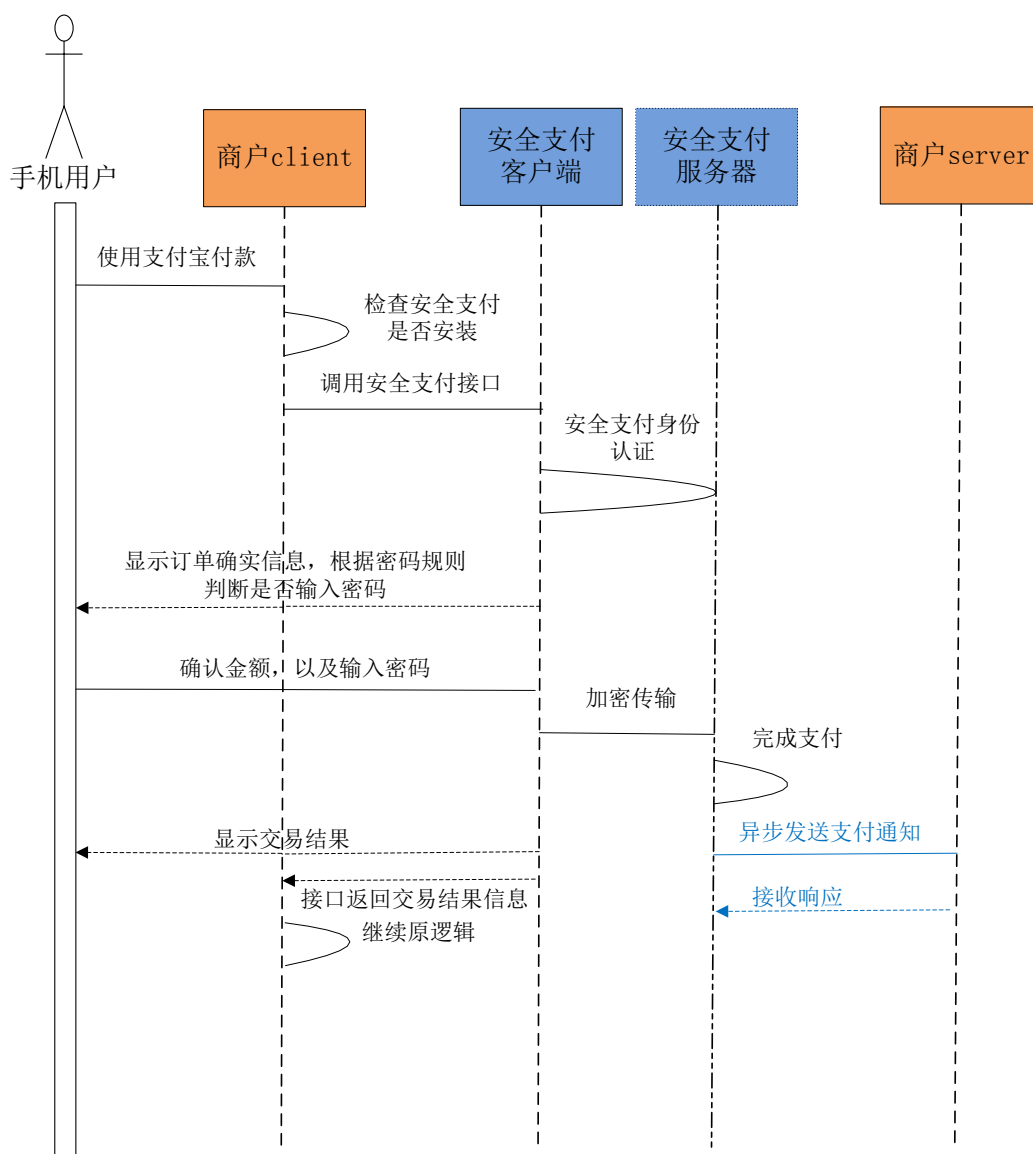


图 1-1 安全支付业务流程图

1.3 调用安全支付数据流程图

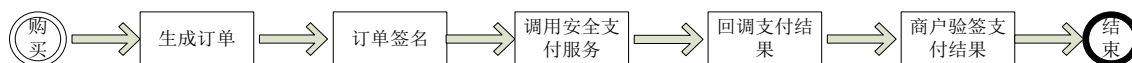


图 1-2 安全支付数据流程图

第二章 安全支付接入流程

2.1 接入前期准备

接入前期准备工作包括**商户签约**和**密钥配置**，已完成商户可略过。

2.1.1 商户签约

首先，商户需要在 <https://ms.alipay.com> 进行注册，并签约安全支付服务。签约成功后可获取支付宝分配的合作商户 ID (PartnerID)，账户 ID (SellerID)，如图：



图 2-1 商户 ID 获取示意图

签约过程中需要任何帮助请致电：**0571-88158090**（支付宝商户服务专线）

2.1.2 密钥配置

签约成功后，商户可登陆 <https://ms.alipay.com> 获取商户账号对应的支付宝公钥，具体获取步骤请见 [3.2 RSA 密钥详解](#)

接着，商户生成商户公钥和商户私钥（具体生成步骤请见 [3.2 RSA 密钥详解](#)），并登陆 <https://ms.alipay.com>，上传商户公钥（具体上传步骤请见 [3.2 RSA 密钥详解](#)）。

至此，接入前期准备工作完成，下一节将使用 demo 测试准备工作是否正确。

2.2 Demo

为了便于商户的接入，我们提供了安全支付 demo。通过本 demo，商户可测试 2.1 节的前期准备工作是否正确完成，同时还可参考 demo 的代码完成接入。

2.2.1 Demo 配置运行

步骤 1:

解压下载的安全支付开发资料压缩包 WS_SECURE_PAY，进入目录“WS_SECURE_PAY\Symbian_S60_v3”，将文件夹“AppDemo11_0413”拷贝到与 SDK 在同一磁盘分区的某目录下，路径切勿过深，本文以目录“D:\Symbian\AppDemo11_0413\”为例。然后在 Carbide.c++中导入该项目，步骤如图：

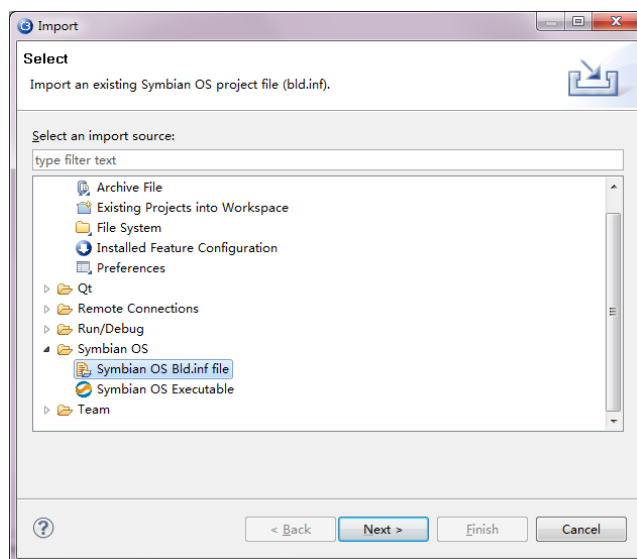


图 2- 2 Demo 导入示意图 a

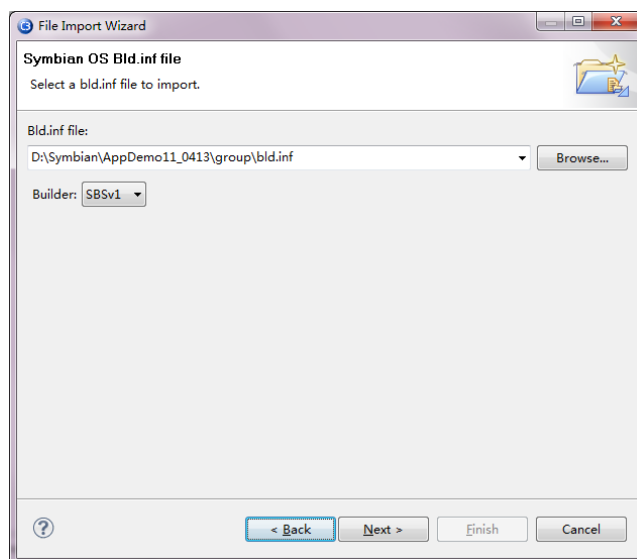


图 2- 3 Demo 导入示意图 b

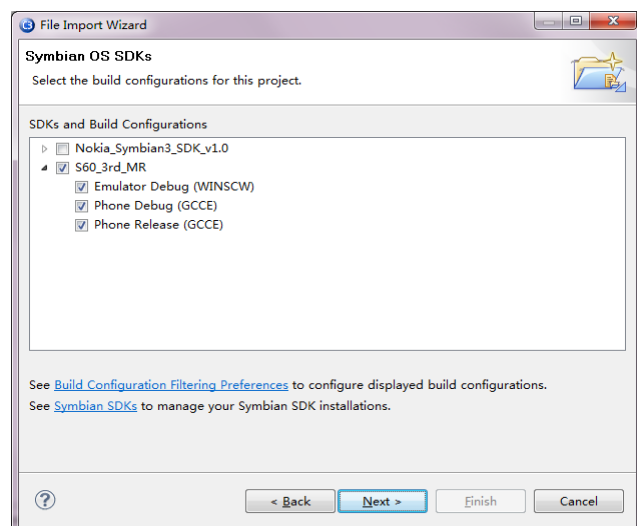


图 2- 4 Demo 导入示意图 c

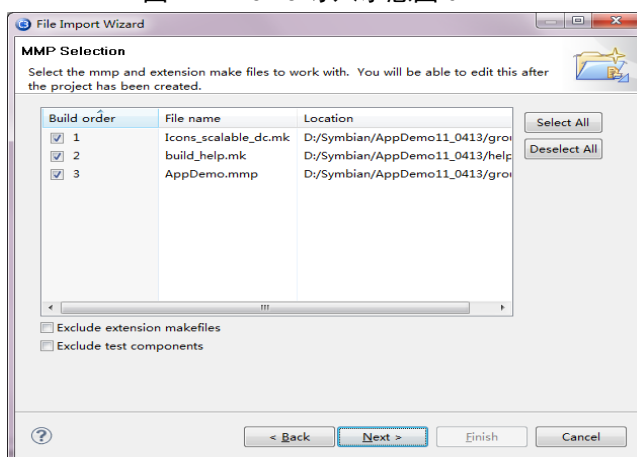


图 2- 5 Demo 导入示意图 d

项目结构如图：

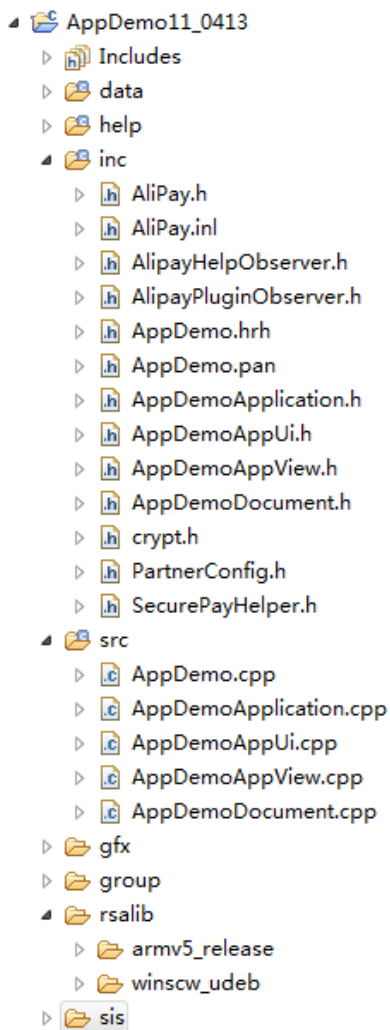


图 2- 6 Demo 项目结构图

步骤 2:

进入 “\AppDemo11_0413\rsalib” ,将 “armv5_release” 或 “winscw_udeb” 目录内的 “RsaLib.lib” 文件复制到 “\$(EPOCROOT)Epoc32\release\\$(PLATFORM)\\$(TARGET)\”, 例如将 “ armv5_release ” 目 录 下 的 “ RsaLib.lib ” 复 制 到 “D:\S60\devices\Fp2\S60_3rd_FP2_SDK_v1.1\epoc32\release\armv5\urel”, 注意平台的对应。

步骤 3:

打开 *PartnerConfig.h*, 按照注释添加商户账号信息, 具体包括: 合作商户 ID、账户 ID、支付宝公钥 (即服务器公钥)、商户公钥、商户私钥。如下图:

```

* Created on: 2010-10-14
* Author: vip
*/

#ifdef PARTNERCONFIG_H_
#define PARTNERCONFIG_H_

//合作商户ID。用签约支付宝账号登录ms.alipay.com后，在账户信息页面获取。
_LIT8(PartnerID,"2088102000947391");

//账户ID。用签约支付宝账号登录ms.alipay.com后，在账户信息页面获取。
_LIT8(SellerID,"2088102000947391");

//服务器公钥，商户用于验签
//_LIT8(RSER_PUB_KEY,"");
_LIT8(RSER_PUB_KEY,"");
//商户私钥，用于加密数据
_LIT8(RUSER_PRI_KEY,"");

//商户公钥，服务端验签使用
_LIT8(RUSER_PUB_KEY,"");

#endif /* CONFIGURATION_H_ */

```

图 2-7 商户信息配置截图

步骤 4:

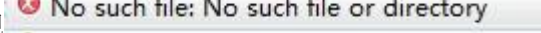
Build 项目，若出现  说明项目所在的文件目录太深，请更改“AppDemo11_0413”所在目录。接着在真机（务必真机，否则无法调用安全支付服务）上运行 demo，如下图：



图 2-8 Demo 效果 a

点击商品购买，此时若提示“Parameter is null”，说明步骤 3 未正确完成，请检查；若提示“没有安全支付插件”，请先在手机上安装压缩包中 *Symbian_S60_v3* 目录下的 *alipay_plugin_v3_230_0727.sisx*。若配置无误，将跳转到安全支付界面，如图：



图 2-9 Demo 效果 b

若出现如下提示，说明密钥配置有误，请仔细阅读 [3.2 RSA 密钥详解](#)



图 2-10 Demo 效果 c

出现输入支付宝账号页面，如图：



图 2- 11 Demo 效果 d

说明接入前期准备工作全部正确完成。接下来将正式进行安全支付的接入集成。

2.2.2 Demo 结构说明

Demo 程序为 S60 GUI Application,项目目录结构如下



其中除 rsalib 目录保存 RSA 静态库外皆由向导生成。

可能用到的代码集中在 `CAppDemoAppView` 类（AppDemoAppView.h、AppDemoAppView.cpp）；

关键函数：

```
TBool CheckPlugin();
```

检测插件是否被安装，如果已经安装，则返回 ETrue,反之返回 EFalse，请下载安装（下载安装模块需要商户完成）。

```
void AlipayPluginEvent(const TDesC8& resultStatus);
```

继承自 `MAlipayPluginObserver`，用来通知支付结果，resultStatus 为支付结果字符串。

```
HBufC8* DoRSA(const TDesC8& aString);
```

对传入的字符串（aString）用商户私钥（RUSER_PRI_KEY）进行 RSA 签名，并将结果存放在返回值（签名失败返回结果为 NULL，需商户判断）。

```
void GenerateKey(TDes8& aKey, TInt aLen);
```

生成长度 aLen 的订单号（[B-3 订单信息描述表](#)中的 **out_trade_no**）并将其存入 aKey 中；需要商户自行重写该函数，规则参考表 B-3 订单信息描述表。

```
HBufC8* GetOrderInfo(const TDesC8& aSubject,const TDesC8&  
aTotalFee);
```

根据传入的商品名（aSubject）和总价（aTotalFee）生成订单详细信息并在函数返回值中返回（各参数请根据实际情况填写）。

```
void OrderPay(const TDesC8& aSubject,const TDesC8& aTotalFee);
```

根据传入的商品名（aSubject）和总价（aTotalFee）生成订单并调用安全支付进行支付。

```
void HandleResult(const TDesC8& aResult);
```

处理支付结果（aResult）。

```
void EscapeChar(TDes8& aData);
```

处理支付结果（aResult），去掉 JSON 字符串中的转义字符（“\”）。

```
TInt GetPayStatus(const TDesC8& aResult);
```

从支付结果（aResult）中获取支付状态码（状态码含义参考附录 A 错误代码列表）。

```
TBool IsSuccess(const TDesC8& aResult);
```

从支付结果（aResult）中获取支付状态描述信息（**success="true"**返回 **ETrue**，否则返回 **EFalse**）。

```
void GetVerifyText(const TDesC8& aResult, HBufC8** aSign, HBufC8**  
aPlaintText);
```

从支付结果（aResult）中获取订单信息（aPlaintText）和签名结果（aSign），用来验签。

```
TInt VerifyL ( const TDesC8& aSignPubKey, const TDesC8&  
aVerifyText, const TDesC8& aPlaintText );
```

验签函数，使用支付宝公钥 aSignPubKey 和签名字串（aVerifyText）验证本次支付的订单信息（aPlaintText）是否被篡改，如果验签通过返回 1，反之返回 0。

2.3 安全支付集成

本章指导在商户项目中集成安全支付，关键代码以 Demo 为例。

步骤 1：添加头文件

复制 `AppDemo11_0413\inc` 目录下的 `AlipayPluginObserver.h`，`AliPay.h`，`AliPay.inl` 到商户的工程下。同时确保“`RsaLib.lib`”已经复制到 SDK 目录（[详见](#)）

步骤 2：初始化安全支付服务句柄

在调用安全支付进行支付前，需要先初始化安全支付服务句柄，代码如下：

```
CInterfaceDefinition* interface = CInterfaceDefinition::NewL(_L("alipay"));
```

步骤 3：订单数据生成

在调用安全支付时，需要提交订单信息 `aOrderInfo`，其中参数以“`key=value`”形式呈现，参数之间以“&”分割，所有参数不可缺。示例如下：（[红色参数](#)表示该参数值需与示例一致，不可自定义；[蓝色参数](#)表示值可自定义。具体参数说明请见[订单信息描述](#)）

```
partner="2088002007260245"&seller="2088002007260245"&out_trade_no=
"500000000006548"&subject="商品名称"&body="这是商品描述
"&total_fee="30"&notify_url="http://notify.java.jpxx.org/index.jsp
"&sign="kU2Fa3x6V985g8ayTozI1eJ5fHtm8%2FJGeJQf9in%2BcVmRJjHaExbirn
GGKJ%2F7B63drqc4Kj1k%2FSg6vtSIkOtdvVBrRDpYaKxXVqkJTzRYgUwrrpMudbIj
9aMS203dHG0GPyl4Zb6jKDyXHabGG0aBJY3QA7JuTJ23t6SqV%2B5f1xg%3D"&sign
_type="RSA"
```

其中 `sign` 值的生成，请见[对商品信息进行 RSA 签名](#)。需要[特别注意](#)的是：对数据签名后得到的 `sign` 值必须进行 `URLEncode`，之后才可作为参数。

步骤 4：调用安全支付

准备好参数后，即可调用安全支付进入支付流程，代码如下：

```
TRAPD(error,interface->AliXPay(REINTERPRET_CAST(CCoeAppUi*,iCoeEnv->Appui())),this,info,id);
```

`AliXPay` 函数具体说明请见[AliXPay 方法描述](#)。

步骤 5：支付结果获取和处理

调用安全支付后，将通过两种途径获得支付结果：

1、`AliXpay` 方法的返回：

该方法返回的结果通过该方法的参数 `MAliPayPluginObserver* aObserver` 获取，

若该参数为 `null`，则无支付结果。支付结果的详细信息[详见](#)

2、支付宝服务器通知

商户需要提供一个 http 协议的接口，包含在参数里传递给安全支付，即 notify_url。

支付宝服务器在支付完成后，会用 POST 方法调用 notify_url，以 xml 为数据格式传输

支付结果，[详见](#)

步骤 6：析构安全支付服务句柄

支付完成后需要清空安全支付服务句柄，并且关掉 ECom 服务。

```
delete interface;
```

```
REComSession::FinalClose();
```

接下来以 Demo 代码为例，介绍整个流程：

FileName: AppDemoAppView.cpp

```
void CAppDemoAppView::ConstructL(const TRect& aRect)
{
    // Create a window for this application view
    CreateWindowL();

    if(CheckPlugin())
    {
        TRAPD(err, iInterface =
CInterfaceDefinition::NewL(KAliXPayOperationName));
    } //初始化安全支付服务句柄
    CreateListBox();
    // Set the windows size
    SetRect(aRect);
    // Activate the window, which makes it ready to be drawn
    ActivateL();
}

. . . . .
. . . . .
//点击商品后调用的支付函数
void CAppDemoAppView::OrderPay(const TDesC8& aSubject,
    const TDesC8& aTotalFee)
{
    HBufC8* order = GetOrderInfo(aSubject, aTotalFee); //订单信息的生成可参考
GetOrderInfo函数
    if (order == NULL)
        return;
    TBuf8<2048> info;
```

```
HBufC8* sign = NULL;

sign = DoRSA(order->Des()); //获取签名值sign

HBufC8* signEncoded = EscapeUtils::EscapeEncodeL(sign->Des(),
    EscapeUtils::EEscapeUrlEncoded); //务必对sign值进行Encode

info.Append(order->Des());
info.Append(_L8("&sign=\""));
info.Append(signEncoded->Des());
info.Append(_L8("\\"));
info.Append(_L8("&sign_type=\"RSA\""));
delete sign;
delete signEncoded;
delete order;
TUint32 id = 0;
if(iInterface)
{
    TRAPD(error, iInterface->AliXPay(REINTERPRET_CAST(CCoeAppUi*,
iCoeEnv->AppUi()), this, info, id)); //若服务句柄初始化成功，则调用安全支付
}
else
{
    NotifyUser(_L("Unable to pay, please download and install Alipay
SecurePay plugin!"));
}
}
```

2.4 应用发布

目前，我们为第三方应用客户端提供了捆绑安装的方式。即将安全支付服务安装包内嵌入到第三方应用的 sis 包中。在第三方应用客户端安装过程中一并安装安全支付服务。

备注：安全支付服务没有强制需要用户做 SymbianSigned 的快速签名，减少了用户成本。

捆绑安装的具体步骤如下所示：

1. 将 *alipay_plugin.sisx* 拷贝到第三方应用程序客户端 pkg 文件所在目录。
2. 修改 pkg 文件，添加行：

例：@`"alipay_plugin2022_022101.sisx", (0x2003AB2E)`

其中 `alipay_plugin2022_022101.sisx` 是 `sisx` 文件的全路径

`0x2003AB2E` 是该 `sisx` 文件的 UID

3. 生成 `sis` 文件，签名并发布。

第三章 RSA 详解

以下内容加 * 号为重点

3.1 RSA 和 OpenSSL 介绍

3.1.1 什么是 RSA

RSA 是一种非对称的签名算法，即签名密钥（私钥）与验签密钥（公钥）是不一样的，私钥用于签名，公钥用于验签。

在与支付宝交易中，会有 2 对公私钥，即商户公私钥，支付宝公钥。

商户公私钥：由商户生成，商户私钥用于对商户发往支付宝的数据签名；商户公钥需要上传至支付宝，当支付宝收到商户发来的数据时用该公钥验证签名。

支付宝公钥：支付宝提供给商户，当商户收到支付宝发来的数据时，用该公钥验签。

3.1.2 为什么要用 RSA

使用这种算法可以起到防止数据被篡改的功能，保证支付订单和支付结果不可抵赖(商户私钥只有商户知道)。

3.1.3 什么是 OpenSSL

一句话概括：OpenSSL 是基于众多的密码算法、公钥基础设施标准以及 SSL 协议安全开发包。

3.1.4 为什么要用 OpenSSL

通过 OpenSSL 生成的签名和内置的算法可以做到跨平台，这样在不同的开发语言中均可以签名和验签。

3.2 RSA 密钥详解 *

3.2.1 找到生成 RSA 密钥工具

(1) 下载开发指南和集成资料，如下图，您能看到此文档说明指南和集成包已经下载了。



图 3-1 文档下载

(2) 解压下载的压缩包(WS_SECURE_PAY)，找到并解压 openssl-0.9.8k_WIN32(RSA 密钥生成工具).zip 工具包



图 3-2 openssl

3.2.2 生成商户密钥并获取支付宝公钥

(1) 生成原始 RSA 商户私钥文件

假设解压后的目录为 c:\alipay，命令行进入目录 C:\alipay\bin，执行 “*openssl genrsa*

```
c:\alipay\bin>openssl genrsa -out rsa_private_key.pem 1024
Loading 'screen' into random state - done
Generating RSA private key, 1024 bit long modulus
.....+++++
.....+++++
e is 65537 (0x10001)

c:\alipay\bin>
```

图 3-3 生成原始 RSA 商户私钥文件

命令行执行 “ `openssl pkcs8 -topk8 -inform PEM -in rsa_private_key.pem -outform PEM -nocrypt` ” 得到转换为 pkcs8 格式的私钥。复制下图红框内的内容至新建 txt 文档，去掉换行，最后另存为 “private key.txt”（请妥善保管，签名时使用）。

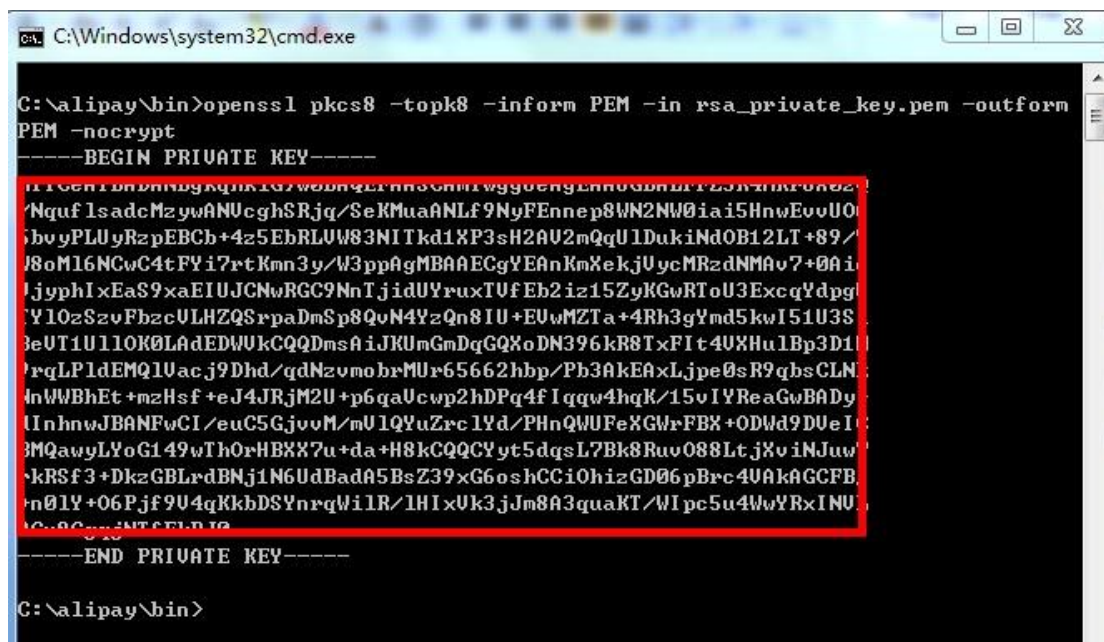


图 3-4 转换私钥格式

命令行执行 “*openssl rsa -in rsa_private_key.pem -pubout -out rsa_public_key.pem*”，在 C:\alipay\bin 文件夹下生成文件 rsa_public_key.pem。接着用记事本打开 rsa_public_key.pem，复制全部内容至新建的 txt 文档，删除文件头 “-----BEGIN PUBLIC KEY-----” 与文件尾 “-----END PUBLIC KEY-----” 及空格、换行，如下图。最后得到一行字符串并保存该 txt 文件为 “public key.txt”。

```
1 -----BEGIN PUBLIC KEY-----
2 MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCuQBQMjMX+ossoDXoi5DlcD0sf
3 6hVT6twgwfuVbyouTSI/cHjH2xpu1S/RD4xXHBI/60GNmewAro2T70i1wxuMpgcD
4 S+3S/0z+4xyrW8ewXfeGmUVPK1yPbkmFeL/OuKWNdhpObOmCyByZPts01kFKDPb9
5 B51xZQzj6b+82L31kQIDAQAB
6 -----END PUBLIC KEY-----
```

去掉头尾注释、换行、空格↵

IGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCuQBQMjMX+ossoDXoi5DlcD0sf6hVT6twgwfuVbyouTSI/cHjH2xpu1S

图 3-5 生成公钥

(4) 上传商户公钥至支付宝

浏览器访问 <https://ms.alipay.com/index.htm> 并用**签约帐号**登录，点击菜单栏“我的产品”，右侧点击“密钥管理”，见下图红色框内



图 3-6 商户公钥上传

点击“上传”，选择步骤(3)生成的“public_key.txt”并完成上传。

(5) 获取 RSA 支付宝公钥

成功上传公钥至支付宝后，页面显示如下：

交易安全检验码（RSA）

支付宝公钥 **public key of Alipay**

MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCORBndEe18p0qgo4KKBfRAHtFzT8BVMddnzp
T/ 2aWldw2I1XUZKAbAjrmYBPsf/lUdC4sGhkoS2CvmQNjx2lu6/xsOQRW70sUIEvpZMo1RUUlh3Cxcg
3f88cmN2Zwz2aCZWVXNSsGmQOQdeyKQviVtNxQ6S2hm4OAW4KIgxQzvhZQIDAQAB

商户公钥 **public key of merchant**

MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQC6d2p9DdbBAGjoWWBiQQiG3rJSzarAeaAHgju
kBhAkdxMhftWCd7qaWbgcpDy/7b42C9Mu5JBly5LEO5apxWI+859HCtT8LTkSNCCNNQj26xW4iIo
6Iz99gwVYIOVRpApHhcbUADPQsZF0SwUVf37EqVRAdCUTeHYVY6nc1JLZcQIDAQAB

更改商户公钥

浏览

上传

请上传文本格式的文件。

图 3-7 支付宝公钥获取

其中红色框内部分即支付宝公钥，请复制至新建 txt 文档，**去掉换行和空格**，妥善保存（用于验签收到的支付宝通知）。

3.3 RSA 签名和验签 *

建议：签名和验签尽量在商户服务器端进行，同时一些敏感数据（如公私钥等）也应存储在服务器端，避免可能的安全隐患。

3.3.1 RSA 签名

- (1) 在 MMP 添加下面的库。

```
LIBRARY estlib.lib imut.lib //libc 库和 base64 编码库
```

```
STATICLIBRARY RsaLib.LIB
```

- (2) 在 CPP 中添加头文件。

```
#include "crypt.h"
```

- (3) 生成商品订单：

可参考 demo 中 `CAppDemoAppView::OrderPay` 函数；

例子：出售商品（`subject`）“Iphone4”，价格（`total_fee`）“1”元，外部交易号（`out_trade_no`）“zzzz”，商品描述（`body`）为“秒杀”，订单支付完成通知 URL（`notify_url`）为“`http://notify.java.jpax.org/index.jsp`”

则生成如下商品信息字符串：

```
partner="xxxx"&seller="yyyy"&out_trade_no="zzzz"&subject="Iphone4"
&body="秒杀"
"&total_fee="1"&notify_url="http%3A%2F%2Fnotify.java.jpxx.org%2Findex.jsp"
```

备注：notify_url 的值需要进行 URLEncode 编码。

(4) 对商品信息进行 RSA 签名：

```
HBufC8* signTest = NULL;
```

```
Crptpt::RsaSha1SignL ( plaintText, KPrivKey,&signTest );
```

参数说明：

plaintext: 待签名的字符串(红色部分)

KPrivKey: 商户私钥(pkcs8 转换后的商户私钥)

signTest: 签名值 (蓝色部分，传递给安全支付前需要 URL 编码)

备注：signTest 必须在使用后删除掉，以免引起内存泄露。

签名后的订字符串示例：

```
partner="xxxx"&seller="yyyy"&out_trade_no="zzzz"&subject="Iphone4"&body="秒杀"
"&total_fee="1"&notify_url="http%3A%2F%2Fnotify.java.jpxx.org%2Findex.jsp"&sign_type="RSA"&sign="00I1APPVQcK5bbSgdeFx9HB3Yu/U2+akTZ3T0/P7v3g7XD7TsQCprb609Nybr8CDIrztdUseQN/TCXuEvCU2cvCt1xX9UUyI6fOxXxQf1DWx7IE2S7Zo5wOeVWmMBnQCQV8iDjcNxGHwhCT09bVVf0wba0iHXvAYzWlvPhyR+0="
```

3.3.2 RSA 验签

(1) 使用 RSA 库进行验签：

```
Crptpt::VerifyL ( signPubKey,verifyText,plaintText );
```

参数说明：

signPubKey: 支付宝公钥

verifyText: 签名(蓝色部分)

plaintText: 待验签字符串(红色部分)

返回值：验签成功则返回1，反之返回0

备注：AlipayPluginEvent返回的json字符串需要处理转义字符(可参考Demo中EscapeChar函数的处理方式)否则可能导致验签无法通过。

以下是一个订单支付成功完整信息的示例：

```
resultStatus={9000};memo={      交      易      成
功 };result={partner="2088002007260245"&seller="2088002007260245"&out
_trade_no="6000000000006891"&subject="商品名称"&body="这是商品描述
"&total_fee="1"&notify_url="http%3A%2F%2Fnotify.java.jp%2F
index.jsp"&success="true"&sign_type="RSA"&sign="00I1APPVQcK5bbSgdeF
x9HB3Yu/U2+akTZ3T0/P7v3g7XD7TsQCprb609Nybr8CDIrztdUseQN/TCXuEvCU2cvC
t1xX9UUYI6f0xXxQF1DWx7IE2S7Zo5wOeVWmMBnQCQCV8iDjcNxGHwhtCT09bVVf0wbaO
iHXvAYzWlvPhyR+0="}
```

- (2) 为了方便商户接入安全支付服务，我们将签名和验签的方法封装成 *Rsalib.lib* 提供给大家使用。*Rsalib.lib* 从文件目录 *Symbian_S60_v3\AlipayLib* 中获取。

第四章 通知结果

4.1 AlixPay 方法返回的结果

支付结果的处理可以参考类 *CAppDemoAppView* 中的 *HandleResult* 函数：

```
void HandleResult(const TDesC8& aResult);
```

结果信息详细描述如下：

表 4-1 AlixPay 返回结果描述表

字段名称	描述	属性	备注
resultStatus	本次操作的状态返回值。	字符串	用来标识本次调用的结果，具体可能的取值，请查看 错误代码列表
memo	本次操作的状态描述。	字符串	
result	本次操作返回的结果数据。	字符串	订单支付结果信息。字符串格式，形式一般如下： <i>partner=""&seller=""&out_trade_no=""&subject=""&body=""&total_fee="30"&notify_url=""&success="true"&sign_type="RSA"&sign="xxx"</i>

			<p>其中：</p> <p><code>&success="true"&sign_type="RSA"</code></p> <p><code>&sign="xxx"</code></p> <p>之前的部分为商户的原始数据。</p> <p><code>success</code>，用来标识本次支付结果。</p> <p><code>sign="xxx"</code>为支付宝对本次支付结果(红色部分)的签名，商户可以使用签约时支付宝提供的公钥进行验证。</p>
--	--	--	--

结果判断说明：

需要通过 `resultStatus` 以及 `result` 字段的值来综合判断并确定支付结果。在 `resultStatus=9000`，并且 `success="true"`以及 `sign="xxx"`校验通过的情况下，证明**支付成功**。其它情况归为失败。较低安全级别的场合，也可以只通过检查 `resultStatus` 以及 `success="true"`来判定支付结果。以下为订单支付成功的完成信息示例：

```
resultStatus={9000};memo={          交          易          成
功};result={partner="2088002007260245"&seller="2088002007260245"&out_trade_no="60000
0000006891"&subject=" 商 品 名 称 "&body=" 这 是 商 品 描 述
"&total_fee="1"&notify_url="http://notify.java.jpxx.org/index.jsp"&success="true"&s
ign_type="RSA"&sign="00I1APPVQcK5bbSgdeFx9HB3Yu/U2+akTZ3T0/P7v3g7XD7TsQCprb609Nybr8
CDIrztdUseQN/TCXuEvCU2cvCt1xX9UUyI6f0xXxQF1DWx7IE2S7Zo5wOeVWmMBnCCQCV8iDjcNxGHwhtCT0
9bVVf0wbaOiHXvAYzW1vPhyR+0="}
```

4.2 notify_url 通知说明

4.2.1 什么是 Notify_url

支付宝通过访问商户提供的地址的形式，将交易状态信息发送给商户服务器。商户通过支付宝的通知判断交易是否成功，具体如下：

商户地址：提供一个 http 的 URL(例:`http://www.partneretest.com/servlet/NotifyReceiver`)，支付宝将以 **POST** 方式调用该地址。

通知触发条件：交易状态发生改变，如交易从“创建”到“成功”或“关闭”。

商户返回信息：商户服务器收到通知后需返回**纯字符串 “success”**，不能包含其他任何 HTML 等语言的文本。

通知重发：若支付宝没有收到商户返回的“success”，将对同一笔订单的通知进行周期性重发 (间隔时间为：2 分钟,10 分钟,10 分钟,1 小时,2 小时,6 小时,15 小时共 7 次)。

交易判断条件：收到 `trade_status=TRADE_FINISHED`（如果签有高级即时到帐协议则 `trade_status=TRADE_SUCCESS`）的请求后才可判定交易成功（其它 `trade_status` 状态请求可以不作处理）

4.2.2 Notify_url 接收数据示例

```
notify_data=<notify><partner>2088201564809153</partner><discount>0.00</discount><payment_type>1</payment_type><subject> 测 试 商 品</subject><trade_no>2012041821018998</trade_no><buyer_email>xxxx@xx.com</buyer_email><gmt_create>2012-04-18 11:06:52</gmt_create><quantity>1</quantity><out_trade_no>0418110644-1034</out_trade_no><seller_id>2088201564809153</seller_id><trade_status>TRADE_SUCCESS</trade_status><is_total_fee_adjust>N</is_total_fee_adjust><total_fee>0.01</total_fee><gmt_payment>2012-04-18 11:06:52</gmt_payment><seller_email>alipay@alipay.com</seller_email><price>0.01</price><buyer_id>2088302175666987</buyer_id><use_coupon>N</use_coupon></notify>&sign=ZPZULntRpJwFmGNIVKwjLEF2Tze7bqs60rxQ22CqT5J1U1vGo575QK9z/+p+7E9cOoRoWzqR6xHZ6WVv3dloyGKDR0btvrdqPgUAoeaX/Y0WzTh00vwcQ+HBtXE+vPTfAqjCTxiiSJE0Y7ATCF1q7iP3sfQxhS0nDUug1LP30Lk=
```

参数说明（注意，具体接收到的数据可能与例子有细微出入，仅确保下表内参数不变）：

notify_data：待验签数据(红色部分)，主要参数说明请见下表

sign：签名(蓝色部分)

备注：在调用验签方法时，需要将“**notify_data=**”这几个字符加上，一并验签，以上红色部分

具体的验签方法请参考 [3.3.2 RSA 验签](#)

Notify_data 参数说明

参数名	说明
trade_status	用于判断交易状态，值有： TRADE_FINISHED：表示交易成功完成 WAIT_BUYER_PAY：表示等待付款 TRADE_SUCCESS：表示交易成功（高级即时到帐）
total_fee	交易金额
subject	商品名称
out_trade_no	外部交易号（商户交易号）
trade_no	支付宝交易号
gmt_create	交易创建时间
gmt_payment	交易付款时间 若交易状态是“WAIT_BUYER_PAY”则无此参数

第五章 常见问题

1. 客户端验签，报“订单信息被篡改”是什么问题？

可能有以下2种情况

- a) 有可能数据在传输过程中被黑客截取和篡改
- b) 检查`plaintext`(待签名的字符串)中是否有以下四个符号，如果参数当中包含了这四个字符也会报“订单信息被篡改”：
 - + 加号
 - & 连接符
 - “ 双引号
 - = 等号

2. 客户端调用安全支付时对 body 和 subject 进行 URLEncode 会报签名错误，到底哪些需要 URLEncode？

调用安全支付接口时，只需要对参数`sign`进行`URLEncode`，其他参数都不能`URLEncode`，安全支付服务插件会对所有参数进行`URLEncode`，所以不用担心中文乱码

3. 上传商户公钥报格式错误怎么办？

首先确认上传的位置是否是`RSA`的下面，注意不要是`DSA`，无线目前不支持`DSA`加密；另外请检查上传的文件中是否去除注释、空格、换行等，必须是一行的字符串

附录 A 错误代码列表

以下为安全支付服务所定义的错误代码：

表 A-1 系统定义的错误代码表

错误代码	含义
9000	操作成功
4000	系统异常
4001	数据格式不正确
4003	该用户绑定的支付宝账户被冻结或不允许支付
4004	该用户已解除绑定
4005	绑定失败或没有绑定

4006	订单支付失败
4010	重新绑定账户。
6000	支付服务正在进行升级操作。
6001	用户中途取消支付操作。
6002	网络连接异常。

附录 B 安全支付服务接口

1 安全支付服务接口列表

目前 S60 平台上的安全支付服务接口如下表所示：

表B-1 支付服务接口表

接口名称	接口描述
CInterfaceDefinition::NewL()	初始化安全支付句柄

2 AlixPay 方法描述

表B-2 AlixPay()方法信息描述

方法原型	
void AliXPay(CCoeAppUi* aAppUi, MAlipayPluginObserver* aObserver, const TDesC8& aOrderInfo, TUint32 aIpld)	
方法功能	
提供外部商户订单的支付	
参数说明	
CCoeAppUi* aAppUi	CCoeAppUi 对象指针不能为空，供安全支付服务获得用户事件，响应用户按键使用。
MAlipayPluginObserver* aObserver	安全支付插件 observer 对象指针，返回支付结果，不可以为空，否则拿不到支付结果。相应的结果参考 支付结果信息描述表 。

TDesC8& aOrderInfo	<p>主要包含外部商户的订单信息，key="value"形式，以&连接。示例如下：</p> <pre>partner="2088002007260245"&seller="2088002007260245"&out_trade_no="500000000006548"&subject="商品名称"&body="这是商品描述"&total_fee="30"&notify_url="http://notify.jva.jp/xx.org/index.jsp"&sign="kU2Fa3x6V985g8ayTozI1eJ5fHtm8%2FJGeJQf9in%2BcVmRJjHaExbirnGGKJ%2F7B63drqc4Kjlk%2FSg6vtSIk0tdvVBrRDpYaKxXVqkJTzRYgUwrrpMudbIj9aMS203dHG0GPyL4Zb6jKDYXHabGG0aBJY3QA7JuTJ23t6SqV%2B5f1xg%3D"&sign_type="RSA"</pre> <p>参考订单信息描述表查看各个字段的含义。</p>
TUint32 alaplId	接入点 Id，如果第三方应用程序已经拥有了自己的接入点，请将此接入点传给安全支付服务，安全支付服务将使用用户传进来的接入点。如果不传接入点，安全支付再联网时会弹出接入点供用户选择。
备注	
<p>该方法是安全支付插件提供的核心方法，是线程安全的。多个线程对此方法的调用会按顺序排队进行，前一个调用返回后，下一个调用才会开始。</p>	

3 订单信息描述

表 B-3 订单信息描述表

字段名称	描述	属性	备注
partner	合作商户 ID。应用开发商与支付宝签约接入支付服务时，由支付宝分配。	字符串	非空、16 位数字。商户可以用签约支付宝账号登录 https://ms.alipay.com 获取。
seller	账户 ID。订单付款成功后，钱会打到本账号中。	字符串	非空、16 位数字。商户可以用签约支付宝账号登录 https://ms.alipay.com 获取。
out_trade_no	商家自己产生的订单编号，由商家统一定义，但此号	字符串	非空、64 位字母、数字和下划线组成

	不能有重复。		
subject	商品名称。 由商家统一定义， 可重复。	字符串	非空、64 位字符（含 中文字符）
body	商品的具体描述 信息。	字符串	非空、1024 位字符（含 中文字符）
total_fee	本次支付的总费用。所有商品的 费用总和，以人民币 元 为 单 位 。 如 1.50。	字符串	非空、大于0的数字（精 度不超过两位的小数， 如 1.00）
notify_url	商家提供的 url。 订单支付结束时， 支付宝服务端会 回调这个 url，通 知商家本次支付 的结果。	字符串	非空、255 位，需要符 合 url 编码规则。
sign	上述订单信息的 签名。对整个订单 按支付宝约定的 方式的签名，签名 需包括如下参数： partner=""&sel ler=""&out_tra de_no=""&subje ct="" "&body=" " &total_fee="3 0"¬ify_url= ""	字符串	非空。 商家自己生成一对公 私钥。用这个私钥对订 单信息签名，公钥提供 给支付宝，供支付宝在 验证订单签名时使用。 需要符合 url 编码规 则，并且 字符编码为 UTF-8.
sign_type	签名类型。由商户 选择，以下算法目 前可用（RSA）。	字符串	非空，定值