Samsung Pay

REST API
Barclays

cybersource
A Visa Solution

Developer Guide

Cybersource Contact Information

For general information about our company, products, and services, go to https://www.cybersource.com.

For sales questions about any Cybersource service, email sales@cybersource.com or call 650-432-7350 or 888-330-2300 (toll free in the United States).

For support information about any Cybersource service, visit the Support Center: https://www.cybersource.com/support

Copyright

Restricted Rights Legends

For Government or defense agencies: Use,duplication, or disclosure by the Government or defense agencies is subject to restrictions as set forth the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013 and in similar clauses in the FAR and NASA FAR Supplement.

For civilian agencies: Use, reproduction, or disclosure is subject to restrictions set forth in suparagraphs (a) through (d) of the Commercial Computer Software Restricted Rights clause at 52.227-19 and the limitations set forth in Cybersource Corporation's standard commercial agreement for this software. Unpublished rights reserved under the copyright laws of the United States.

Trademarks

Confidentiality Notice

This document is furnished to you solely in your capacity as a client of Cybersource and as a participant in the Visa payments system.

By accepting this document, you acknowledge that the information contained herein (the "Information") is confidential and subject to the confidentiality restrictions contained in Visa's operating regulations and/or other confidentiality agreements, which limity our use of the Information. You agree to keep the Information confidential and not to use the Information for any purpose other than its intended purpose and in your capacity as a customer of Cybersource or as a participant in the Visa payments system. The Information may only be disseminated within your organization on a need-to-know basis to enable your participation in the Visa payments system. Please be advised that the Information may constitute material non-public information under U.S. federal securities laws and that purchasing or selling securities of Visa Inc. while being aware of material non-public information would constitute a violation of applicable U.S. federal securities laws.

Revision

Version: 24.02

Contents

Samsung Pay Developer Guide **5**
Recent Revisions to This Document 6
Introduction **7**
Requirements for Using Samsung Pay 7
Supported Card Types and Optional Features 8
Transaction Endpoints 8
Getting Started **9**
Registering with Samsung Pay 9
Registering with Cybersource 10
Creating a Project 11
Integrating the Samsung Pay SDK 11
Using the API Key 12
Example: Debug Mode 12
Example: Release Mode 12
Verify That Your Application is Eligible for Samsung Pay 12
Example: Samsung Pay Class 13
Initiating a Payment 13
Required Fields for Initiating a Payment 13
Example: Transaction Request Structure 14
Requesting a Payment 15
Example: Request startSamsungPay() API Method 16
Services **17**
Authorization Service 17
Authorizing a Payment with American Express Using Cybersource Decryption Method 18
Authorizing a Payment with JCB Using Cybersource Decryption Method 20
Authorizing a Payment with Mastercard Using Cybersource Decryption Method 22
Authorizing a Payment with Visa Using Cybersource Decryption Method 24
Authorizing a Payment with American Express Using Merchant Decryption Method 26
Authorizing a Payment with JCB Using Merchant Decryption Method 29

Authorizing a Payment with Mastercard Using Merchant Decryption Method....31
Authorizing a Payment with Visa Using Merchant Decryption Method....33
Authorization Reversal Service....35
Required Fields for Reversing an Authorization....36
Reversing an Authorization....36
Example: Basic Credit Card Authorization Reversal Using the REST API....36
Capture Service....37
Required Fields for Capturing a Payment....37
Capturing a Payment....38
Example: Basic Credit Card Capture Using the REST API....38
Sale Service....39
Required Fields for Performing a Sale....39
Authorizing and Capturing a Payment....39
Example: Basic Credit Card Sale Using the REST API....40

Samsung Pay Developer Guide

Audience and Purpose

This document is written for merchants who want to enable customers to use Samsung Pay to pay for in-app purchases. This document provides an overview for integrating the Samsung Pay SDK and describes how to request the Cybersource API to process an authorization. Merchants must use the Samsung Pay SDK to receive the customer's encrypted payment data before requesting the Cybersource API to process the transaction.

Conventions

The following special statements are used in this document:

Important

An Important statement contains information essential to successfully completing a task or learning a concept.

Warning

A Warning contains information or instructions, which, if not heeded, can result in a security risk, irreversible loss of data, or significant cost in time or revenue or both.

Related Documentation

Refer to the Support Center for complete technical documentation:
http://www.cybersource.com/support_center/support_documentation

Customer Support

For support information about any service, visit the Support Center:
http://support.visaacceptance.com

Recent Revisions to This Document

24.02
This revision contains only editorial changes and no technical updates.

24.01
This revision contains only editorial changes and no technical updates.

23.01
This revision contains only editorial changes and no technical updates.

22.03
Removed **deviceInformation** fields from REST API sales request code examples.

22.02
This revision contains only editorial changes and no technical updates.

22.01
Added Diners Club and Discover as supported card types for Barclays. See *Supported Processors*.

Introduction

Requirements for Using Samsung Pay

In order to use the Cybersource platform to process Samsung Pay transactions, you must have:

- A Cybersource account. If you do not already have a Cybersource account, contact your local Cybersource sales representative.
- A merchant account with a supported processor.
- A profile on the Samsung Pay Partner Portal and an associated partner ID.

Important

Samsung Pay relies on authorizations with payment network tokens. You can sign up for Samsung Pay only when both of the following statements are true:

- Your processor supports payment network tokens.
- Cybersource supports payment network tokens with your processor.

If one or both of the preceding statements are not true, you must take one of the following actions before you can sign up for Samsung Pay:

- Obtain a new merchant account with a processor that supports payment network tokens.
- Wait until your processor supports payment network tokens.

Related concepts
- Supported Card Types and Optional Features on page 8

Related tasks
- Registering with Samsung Pay on page 9
- Registering with Cybersource on page 10

Supported Card Types and Optional Features

Processor	Card Types	Optional Features
Barclays	Diners Club Discover JCB Mastercard Visa	Multiple partial captures Recurring payments

Related Information

Payments Developer Guide

Transaction Endpoints

Test transactions:

- Akamai endpoint: *http://ics2testa.ic3.com/*
- Non-Akamai endpoint: *http://ics2test.ic3.com/*

Production transactions:

- Akamai endpoint: *http://ics2a.ic3.com/*
- Non-Akamai endpoint: *http://ics2.ic3.com/*

Getting Started

Follow these steps to set up Samsung Pay with Cybersource:

1. Registering with Samsung and Cybersource:
   a. *Registering with Samsung Pay* on page 9
   b. *Registering with Cybersource* on page 10
2. Integrating the Samsung SDK, which includes the following tasks:
   a. *Creating a Project* on page 11
   b. *Integrating the Samsung Pay SDK* on page 11
   c. *Using the API Key* on page 12
3. Using the Samsung SDK to:
   a. *Verify That Your Application is Eligible for Samsung Pay* on page 12
   b. *Initiating a Payment* on page 13
   c. *Requesting a Payment* on page 15

Registering with Samsung Pay

1. Create a profile by completing the merchant application on the Samsung Pay Partner Portal.

   After your merchant application is approved, you receive a unique partner ID. Include this ID in your application.

   Important

   You need the partner ID in order to generate a Certificate Signing Request (CSR) in the Business Center. Samsung requires the CSR file in order to encrypt sensitive payment data; it contains an identifier and public key.

2. Using the Samsung Pay Partner Portal, upload the CSR file.

3. Enter an application name and a package name. When you associate the CSR file with the application, Samsung generates a product ID.
4. Create login details for application developers on the Samsung Pay Partner Portal.
5. Download and integrate the Samsung Pay SDK into your application.

   The SDK contains:

   - A Javadoc
   - The Samsung Pay SDK files samsungpay.jar and sdk-v1.0.0.jar
   - A sample app
   - The branding guide
   - Image files
6. Register a Samsung account ID and request a debug-api-key file using the Samsung Pay Partner Portal. The Samsung account ID, the debug-api-key, and the product ID are used to validate your application so that you can use the Samsung Pay SDK for testing.
7. Submit your application for approval using the Samsung Pay Partner Portal. Upload the final version of the Android Application Package (APK) file using the Samsung Pay Partner Portal, and include screenshots of your checkout page displaying the Samsung Pay logo.

Related concepts
- Verify That Your Application is Eligible for Samsung Pay on page 12
- Required Fields for Initiating a Payment on page 13

Related tasks
- Registering with Cybersource on page 10
- Requesting a Payment on page 15
- Using the API Key on page 12

Related information
- Samsung Pay Partner Portal

Registering with Cybersource

1. Log in to the Business Center:
   - Create a CSR file for test transactions: *https://businesscentertest.cybersource.com*
   - Create a CSR file for production transactions: *https://businesscentertest.cybersource.com*
2. On the left navigation pane, click the **Payment Configuration** icon.
3. Click **Digital Payment Solutions**. The Digital Payments page opens.
4. Click **Configure**. The Samsung Pay Registration panel opens.
5. Enter your Samsung partner ID.
6. Click **Generate New CSR**.

7. To download your CSR, click the **Download** icon next to the key.
8. Follow your browser's instructions to save and open the file.

> **Important**
>
> Only one CSR is permitted for each unique Samsung partner ID. If you modify your Samsung partner ID, you must generate a new CSR.

9. Complete the enrollment process by submitting your CSR to Samsung.

Creating a Project

You use Android Studio to create a new Android Studio project, which is required to integrate the Samsung SDK.

1. Download Android Studio from the following website: *https://developer.android.com/studio/index.html*.
2. Open Android Studio and click **Start a new Android Studio project**.
3. In the New Project settings menu, enter the name of your application and the company domain.
4. To change the package name, click **Edit**. By default, Android Studio sets the last element of the project's package name to the name of your application.
5. Click **Next**.
6. In the Target Android Devices settings menu, choose the required API levels.
7. Click **Next**.
8. Choose the required activity and click **Finish**.

Integrating the Samsung Pay SDK

1. Add the samsungpay.jar and sdk-v1.0.0.jar files to the libs folder of your Android project.
2. Choose **Gradle Scripts > build.gradle** and enter the dependencies shown below.

```
dependencies {
   compile files('libs/samsungpay.jar')
   compile files(libs/sdk-v1.0.0.jar')
}
```

3. Import the package.

```
import com.samsung.android.sdk.samsungpay;
```

Using the API Key

The API key is used to verify that your app (in debug mode or release mode) can use the Samsung Pay SDK APIs with the Samsung Pay application. To get the API key, you must create a debug-api-key file and include it in the manifest file.

To use the API key, include it in the manifest file with a custom tag. This enables the merchant app android manifest file to provide the `DebugMode`, `spay_debug_api_key` values as metadata.

Related tasks
- Registering with Samsung Pay on page 9

Example: Debug Mode

```
<meta-data
  android:name="debug_mode"
  android:value="Y" />
<meta-data
  android:name="spay_debug_api_key"
  android:value="asdfggkndkeie17283094858" />
```

Example: Release Mode

```
<meta-data
  android:name="debug_mode"
  android:value="N" />
```

Verify That Your Application is Eligible for Samsung Pay

You must initialize the SSamsungPay class to verify that your application is eligible for Samsung Pay and to display the Samsung Pay button to the customer (refer to branding guidelines).
The SSamsungPay class provides the following API methods:

- `initialize()`—initializes the Samsung Pay SDK and verifies eligibility for Samsung Pay, including the device, software, and business area.

> **Important**
>
> Request the `initialize()` API method of the SSamsungPay class before using the Samsung Pay SDK.

- `getVersionCode()`—retrieves the version number of the Samsung Pay SDK as an integer.
- `getVersionName()`—retrieves the version name of the Samsung Pay SDK as a string.

After the `initialize()` API method request is successful, display the Samsung Pay button to the customer.
If the `initialize()` API method request fails, the method displays one of the following errors:

- `SsdkUnsupportedException`—the device is not a Samsung device or does not support the Samsung Pay package.
- `NullPointerException`—the context passed is null.

Example: Samsung Pay Class

```
SSamsungPay spay = new SSamsungPay();
try {
   spay.initialize(mContext);
} catch (SsdkUnsupportedException e1) {
   e1.printStackTrace();
   pay_button.setVisibility(View.INVISIBLE);
}
```

Initiating a Payment

You are required to use a specific transaction request structure and required fields to initiate a payment.

Required Fields for Initiating a Payment

The following fields are required for initiating a payment; include these fields in the `PaymentInfo` class:

Important

If the required fields are not included, you receive a `NullPointerException` error.

Merchant Name	The merchant name as it appears on the payment sheet of Samsung Pay and customer's bank statement.
Amount	
Payment Protocol	3-D Secure.
Permitted Card Brands	Specify the card brands that are supported such as American Express, JCB, Mastercard, or Visa.
Merchant ID	

Order Number	
Shipping Address	This field is required if **SEND_SHIPPING or NEED_BILLING_AND_SEND_SHIPPING** is set for AddressVisibilityOption.
Address Visibility Option	
Card Holder Name	
Recurring Option	

Example: Transaction Request Structure

```
private PaymentInfo makeTransactionDetails() {
// Supported card brands
ArrayList<CardInfo.Brand> brandList = new ArrayList<CardInfo.Brand>();
if (visaBrand.isChecked())
brandList.add(CardInfo.Brand.VISA);
if (mcBrand.isChecked())
brandList.add(CardInfo.Brand.Mastercard);
if (amexBrand.isChecked())
brandList.add(CardInfo.Brand.AMERICANEXPRESS);

// Basic payment information
PaymentInfo paymentReq = new PaymentInfo.Builder()
.setMerchantId(“merchantID”)
.setMerchantName("Test").setAmount(getAmount())
.setShippingAddress(getShippingAddressInfo())
.setOrderNumber(orderNoView.getText().toString())
.setPaymentProtocol(PaymentProtocol.PROTOCOL_3DS)
.setAddressInPaymentSheet(AddressInPaymentSheet.DO_NOT_SHOW)
.setAllowedCardBrands(brandList) .setRecurringEnabled(isRecurring)
.setCardHolderNameEnabled(isCardHolderNameRequired)
.build();
return paymentReq;
}

// Add shipping address details
private Address getShippingAddressInfo() {
Address address = new Address.Builder()
.setAddressee(name.getText().toString())
.setAddressLine1(addLine1.getText().toString())
.setAddressLine2(addline2.getText().toString())
.setCity(city.getText().toString())
.setState(state.getText().toString())
.setCountryCode(country.getSelectedItem().toString())
.setPostalCode(zip.getText().toString()).build(); return address;
}

// Add amount details private Amount getAmount() {
Amount amount = new Amount.Builder()
.setCurrencyCode(currencyType.getSelectedItem().toString())
.setItemTotalPrice(productPrice.getText().toString())
.setShippingPrice(shippingPrice.getText().toString())
.setTax(taxPrice.getText().toString())
```

```
.setTotalPrice(totalAmount.getText().toString()).build();
return amount;
}
```

Requesting a Payment

1. Use the `startSamsungPay()` API method in the `PaymentManager` class. The `PaymentManager` class includes the following API methods:
   - `startSamsungPay()`—requests to initiate payment with Samsung Pay.
   - `updateAmount()`—updates the transaction amount if shipping address or card information is updated by Samsung Pay.
   - `updateAmountFailed()`—returns an error code when the new amount cannot be updated because of a wrong address.
2. Request the `startSamsungPay()` API method and include the following data:
   - `PaymentInfo`—contains payment information.
   - `PID`—the product ID created in the Samsung Pay Partner Portal.
   - `StatusListener`—the result of the payment request is delivered to `StatusListener`. This listener should be registered before you call the `startSamsungPay()` API method.

   When you request the `startSamsungPay()` API method, the Samsung Pay online payment sheet is displayed on your application. The customer selects a registered card for payment and can also update the billing and shipping address.
   The payment reply is delivered as one of the following events to `StatusListener`:

   - `onSuccess()`—this event is requested when Samsung Pay confirms the payment. It includes `encryptedPaymentCredential` in JSON format:
     - method: Payment protocol: 3-D Secure.
     - merchant_ref: Merchant reference code.
     - billing_address.street: Number, street name.
     - billing_address.state_province: Two-letter state code.
     - billing_address.zip_postal_code: Five-character zip code.
     - billing_address.city: City name.
     - billing_address.county: Two-letter country code.
     - 3ds.type: `S` for Samsung Pay. Encrypted.
     - 3ds.version: Current version `100`. Encrypted.
     - 3ds.data: Base64-encoded payment data. Encrypted.

     Refer to the Samsung Pay developer website for information on how to decrypt the encrypted payment credential.
   - `onFailure()`—this event is requested when the transaction fails. It returns an error code and error message.

Related tasks

- Registering with Samsung Pay on page 9

Related information

- Samsung Pay Developers website: https://pay.samsung.com/developers

Example: Request startSamsungPay() API Method

```
public void onPayButtonClicked(View v) {
   // Call startSamsungPay() method of PaymentManager class.
   // To create a transaction request for makeTransactionDetails() in
   the following code, see Example: Transaction Request Structure on page 14.
   try {
      mPaymentManager.startSamsungPay(makeTransactionDetails(), "enter
      product ID",
mStatusListener);
   } catch (NullPointerException e) {
   e.printStackTrace();
   }
}

private PaymentManager.StatusListener mStatusListener = new
PaymentManager.StatusListener() {
   @Override
   public void onFailure(int errCode, String msg) {
      Log.d(TAG, " onFailed );
   }
   @Override
   public void onSuccess(PaymentInfo arg0, String result) {
      Log.d(TAG, "onSuccess ");
   };
```

Services

The following services are available:

- *Authorization Service* on page 17
- *Authorization Reversal Service* on page 35
- *Capture Service* on page 37
- *Sale Service* on page 39

Authorization Service

You can authorize a payment for Samsung Pay using two different types of decryption methods: Cybersource or Merchant. Each decryption method requires a different set of required API fields. In addition, depending on which card type is used, different fields are required for requesting the authorization service.

Processor-Specific Information About Authorizations and Captures

Payment Processor	Authorization and Capture Information
Barclays	The amount is rounded to the correct number of decimal p laces for the currency. Barclays does not support amounts of 0.01. Barclays supports zero amount authorizations and amounts greater than 0.01.Barclays supports enhanced re sponse codes in authorization response messages. Enhanc ed response codes provide detailed information about dec lined transactions. Contact Barclays customer support to have this capability enabled for your account.

Related concepts

- Authorizing a Payment with American Express Using Cybersource Decryption Method on page 18
- Authorizing a Payment with JCB Using Cybersource Decryption Method on page 20

- Authorizing a Payment with Mastercard Using Cybersource Decryption Method on page 22
- Authorizing a Payment with Visa Using Cybersource Decryption Method on page 24
- Authorizing a Payment with American Express Using Merchant Decryption Method on page 26
- Authorizing a Payment with JCB Using Merchant Decryption Method on page 29
- Authorizing a Payment with Mastercard Using Merchant Decryption Method on page 31
- Authorizing a Payment with Visa Using Merchant Decryption Method on page 33

Authorizing a Payment with American Express Using Cybersource Decryption Method

This section provides the following information:

- *Required Fields for Authorizing a Payment Using American Express and the Cybersource Decryption Method* on page 18
- *Authorizing a Payment* on page 18
- *Example: Cybersource Decryption with American Express Using the REST API* on page 19

Required Fields for Authorizing a Payment Using American Express and the Cybersource Decryption Method

The following fields are required when submitting an authorization request using the Cybersource decryption method:

- **descriptor**-set this field under the **fluidData** object to `RklEPUNPTU1PTi5TQU1TVU5HLklOQVBQLlBBWU1FTlQ=`.
- **processingInformation.commerceIndicator**-set this field to `aesk`.
- **paymentInformation.fluidData.value**
  - Set the field to the value that was returned from Samsung Pay in the 3ds.data block as follows:
    - Retrieve the payment data from Samsung Pay in JSON Web Encryption (JWE) format.
    - Encode it in Base64.
    - Add the value to the **paymentInformation.fluidData.value** field.
- **paymentInformation.tokenizedCard.transactionType**-set this field to `1`.
- **processingInformation.paymentSolution**-set this field to `008`.

Related Information

REST API Field Reference Guide

Authorizing a Payment

1. Send the service request to `https://api.cybersource.com/pts/v2/payments`.

2. Include the required fields in the request.
3. Include optional fields in the request as needed.
4. Check the response message to make sure that the request was successful. A 200-level HTTP response code indicates success. For information about response codes, see *Transaction Response Codes*.

Example: Cybersource Decryption with American Express Using the REST API

Authorization Request

```
{
  "clientReferenceInformation": {
    "code": "demorefnum"
  },
  "processingInformation": {
    "paymentSolution": "008",
    "commerceIndicator": "aesk"
  },
  "paymentInformation": {
    "tokenizedCard": {
      "transactionType": "1",
      "type": "003"
    }
  },
  "fluidData": {
   "descriptor": "ABCDEFabcdefABCDEFabcdef0987654321234567",
   "value": "RklEPUNPTU1PTi5TQU1TVU5HLklOQVBQLlBBWU1FTlQ="
  },
  "billTo": {
    "firstName": "James",
    "lastName": "Smith",
     "address1": "111 S. Division St.",
     "address2": "Suite 123",
     "locality": "Ann Arbor",
     "administrativeArea": "MI",
     "postalCode": "48104-2201",
     "country": "US",
     "email": "demo@example.com",
     "phoneNumber": "9999999999"
   },
   "orderInformation": {
     "amountDetails": {
       "currency": "USD",
       "totalAmount": "100.00"
     }
   }
}
```

Authorization Response

```
{
  "clientReferenceInformation": {
    "code": "demorefnum"
  },
  "orderInformation": {
```

```
      "amountDetails": {
        "currency": "USD",
        "authorizedAmount": "100.00"
      }
    },
    "paymentInformation": {
       "tokenizedCard": {
        "prefix": "593056",
        "suffix": "0842",
        "expirationMonth": "08",
        "expirationYear": "2021"
       }
    },
    "processingInformation": {
      "reconciliationID": "13209256CGJSMQCZ"
    },
    "processorInformation": {
      "approvalCode": "888888",
      "responseCode": "100",
      "avs": {
        "code": "I1"
      }
    },
    "submitTimeUtc": "2015-11-03T205202Z"
    }
}
```

Authorizing a Payment with JCB Using Cybersource Decryption Method

This section provides the following information:

- *Required Fields for Authorizing a Payment Using JCB and the Cybersource Decryption Method* on page 20
- *Authorizing a Payment* on page 18
- *Example: Cybersource Decryption with JCB Using the REST API* on page 21

Required Fields for Authorizing a Payment Using JCB and the Cybersource Decryption Method

The following fields are required when submitting an authorization request using the Cybersource decryption method:

- **descriptor**–set this field under the **fluidData** object to `RklEPUNPTU1PTi5TQU1TVU5HLklOQVBQLlBBWU1FTlQ=`.
- **paymentInformation.fluidData.value**–set this field to the Base64-encoded value obtained from the **paymentData** property of the **PKPaymentToken** object.
- **processingInformation.paymentSolution**–set this field to `008`.

Related Information

REST API Field Reference Guide

Authorizing a Payment

1. Send the service request to https://api.cybersource.com/pts/v2/payments.
2. Include the required fields in the request.
3. Include optional fields in the request as needed.
4. Check the response message to make sure that the request was successful. A 200-level HTTP response code indicates success. For information about response codes, see *Transaction Response Codes*.

Example: Cybersource Decryption with JCB Using the REST API

Authorization Request

```
{
  "processingInformation": {
    "paymentSolution": "008"
  },
  "consumerAuthenticationInformation": {
    "cavv": "EHuWW9PiBkWvqE5juRwDzAUFBAk=",
    "eciRaw": "05"
  },
  "paymentInformation": {
    "tokenizedCard": {
      "number": "xxxx55555555xxxx",
      "expirationMonth": "12",
      "expirationYear": "2031",
      "transactionType": "1",
      "type": "007"
    }
  },
  "billTo": {
    "firstName": "Jane",
    "lastName": "Smith",
    "address1": "123 Main Street",
    "address2": "Suite 12345",
    "locality": "Small Town",
    "administrativeArea": "CA",
    "postalCode": "98765",
    "country": "US",
    "email": "js@example.com",
    "phoneNumber": "9999999999"
  },
  "orderInformation": {
    "amountDetails": {
      "currency": "USD",
      "totalAmount": "100.00"
    }
  }
}
```

Authorization Response

```
{
  "clientReferenceInformation": {
    "code": "ref123"
```

```
  },
  "orderInformation": {
    "amountDetails": {
      "currency": "USD",
      "authorizedAmount": "100.00"
    }
  },
  "processingInformation": {
    "reconciliationID": "15356268CR2XF23X"
  },
  "processorInformation": {
    "approvalCode": "888888",
    "responseCode": "100",
    "paymentSolution": "008",
    "avs": {
      "codeRaw": "I1"
    }
  }
}
```

Authorizing a Payment with Mastercard Using Cybersource Decryption Method

This section provides the following information:

- *Required Fields for Authorizing a Payment Using Mastercard and the Cybersource Decryption Method* on page 22
- *Authorizing a Payment* on page 18
- *Example: Cybersource Decryption with Mastercard Using the REST API* on page 23

Required Fields for Authorizing a Payment Using Mastercard and the Cybersource Decryption Method

The following fields are required when submitting an authorization request using the Cybersource decryption method:

- **descriptor**–set this field under the **fluidData** object to `RklEPUNPTU1PTi5TQU1TVU5HLklOQVBQLlBBWU1FTlQ=`.
- **processingInformation.commerceIndicator**–set this field to `spa`.
- **paymentInformation.fluidData.value**
  - Set the field to the value that was returned from Samsung Pay in the 3ds.data block as follows:
    - Retrieve the payment data from Samsung Pay in JSON Web Encryption (JWE) format.
    - Encode it in Base64.
    - Add the value to the **paymentInformation.fluidData.value** field.
- **paymentInformation.tokenizedCard.transactionType**–set this field to `1`.
- **processingInformation.paymentSolution**–set this field to `008`.

Related Information

REST API Field Reference Guide

Authorizing a Payment

1. Send the service request to https://api.cybersource.com/pts/v2/payments.
2. Include the required fields in the request.
3. Include optional fields in the request as needed.
4. Check the response message to make sure that the request was successful. A 200-level HTTP response code indicates success. For information about response codes, see *Transaction Response Codes*.

Example: Cybersource Decryption with Mastercard Using the REST API

Authorization Request

```
{
  "clientReferenceInformation": {
    "code": "demorefnum"
  },
  "processingInformation": {
    "paymentSolution": "008",
    "commerceIndicator": "spa"
  },
  "paymentInformation": {
    "tokenizedCard": {
      "transactionType": "1"
    }
  },
  "fluidData": {
   "descriptor": "ABCDEFabcdefABCDEFabcdef0987654321234567",
   "value": "RklEPUNPTU1PTi5TQU1TVU5HLklOQVBQLlBBWU1FTlQ="
  },
  "billTo": {
    "firstName": "James",
    "lastName": "Smith",
     "address1": "111 S. Division St.",
     "address2": "Suite 123",
     "locality": "Ann Arbor",
     "administrativeArea": "MI",
     "postalCode": "48104-2201",
     "country": "US",
     "email": "demo@example.com",
     "phoneNumber": "9999999999"
   },
   "orderInformation": {
     "amountDetails": {
       "currency": "USD",
       "totalAmount": "100.00"
     }
   }
}
```

Authorization Response


```
{
  "clientReferenceInformation": {
    "code": "demorefnum"
  },
  "orderInformation": {
    "amountDetails": {
      "currency": "USD",
      "authorizedAmount": "100.00"
    }
  },
  "processingInformation": {
    "reconciliationID": "13209255CGJSMQCR"
  },
  "processorInformation": {
    "approvalCode": "888888",
    "responseCode": "100",
    "avs": {
      "code": "I1"
    }
  },
  "submitTimeUtc": "2015-11-03T205035Z"
  }
}
```


Authorizing a Payment with Visa Using Cybersource Decryption Method

This section provides the following information:

- *Required Fields for Authorizing a Payment Using Visa and the Cybersource Decryption Method* on page 24
- *Authorizing a Payment* on page 18
- *Example: Cybersource Decryption with Visa Using the REST API* on page 25

Required Fields for Authorizing a Payment Using Visa and the Cybersource Decryption Method

The following fields are required when submitting an authorization request using the Cybersource decryption method:

- **descriptor**-set this field under the **fluidData** object to `RklEPUNPTU1PTi5TQU1TVU5HLklOQVBQLlBBWU1FTlQ=`.
- **processingInformation.commerceIndicator**-set this field to `internet`.
- **paymentInformation.fluidData.value**
  - Set the field to the value that was returned from Samsung Pay in the 3ds.data block as follows:
    - Retrieve the payment data from Samsung Pay in JSON Web Encryption (JWE) format.
    - Encode it in Base64.

    - Add the value to the **paymentInformation.fluidData.value** field.
- **paymentInformation.tokenizedCard.transactionType**–set this field to `1`.
- **processingInformation.paymentSolution**–set this field to `008`.

Related Information

REST API Field Reference Guide

Authorizing a Payment

1. Send the service request to `https://api.cybersource.com/pts/v2/payments`.
2. Include the required fields in the request.
3. Include optional fields in the request as needed.
4. Check the response message to make sure that the request was successful. A 200-level HTTP response code indicates success. For information about response codes, see *Transaction Response Codes*.

Example: Cybersource Decryption with Visa Using the REST API

Authorization Request

```
{
  "clientReferenceInformation": {
    "code": "demorefnum"
  },
  "processingInformation": {
    "paymentSolution": "008",
    "commerceIndicator": "internet"
  },
  "paymentInformation": {
    "tokenizedCard": {
      "transactionType": "1"
    }
  },
  "fluidData": {
   "descriptor": "ABCDEFabcdefABCDEFabcdef0987654321234567",
   "value": "RklEPUNPTU1PTi5TQU1TVU5HLklOQVBQLlBBWU1FTlQ="
  },
  "billTo": {
    "firstName": "James",
    "lastName": "Smith",
     "address1": "111 S. Division St.",
     "address2": "Suite 123",
     "locality": "Ann Arbor",
     "administrativeArea": "MI",
     "postalCode": "48104-2201",
     "country": "US",
     "email": "demo@example.com",
     "phoneNumber": "9999999999"
   },
   "orderInformation": {
     "amountDetails": {
       "currency": "USD",
```

```
        "totalAmount": "100.00"
      }
    }
}
```

Authorization Response

```
{
  "clientReferenceInformation": {
    "code": "demorefnum"
  },
  "orderInformation": {
    "amountDetails": {
      "currency": "USD",
      "authorizedAmount": "100.00"
    }
  },
  "paymentInformation": {
     "tokenizedCard": {
      "prefix": "294672",
      "suffix": "4397",
      "expirationMonth": "08"
     }
  },
  "processingInformation": {
    "reconciliationID": "13209254CGJSMQCQ"
  },
  "processorInformation": {
    "approvalCode": "888888",
    "responseCode": "100",
    "avs": {
      "code": "I1"
    }
  }
}
```

Authorizing a Payment with American Express Using Merchant Decryption Method

This section provides the following information:

- *Required Fields for Authorizing a Payment Using American Express and the Merchant Decryption Method* on page 26
- *Authorizing a Payment* on page 18
- *Example: Merchant Decryption with American Express Using the REST API* on page 27

Required Fields for Authorizing a Payment Using American Express and the Merchant Decryption Method

The following fields are required when submitting an authorization request using the Merchant decryption method:

- **consumerAuthenticationInformation.cavv**–set this field to the 3-D Secure cryptogram of the payment network token.

> **Important**
>
> Include the whole 20-byte cryptogram in the cavv field. For a 40-byte cryptogram, split the cryptogram into two 20-byte binary values (block A and block B). Set the cavv field to the block A value and set the xid field to the block B value.

- **paymentInformation.card.number**-set this field to the payment network token value.
- **paymentInformation.card.expirationMonth/ paymentInformation.tokenizedCard.expirationMonth**-set this field to the payment network token expiration month value.
- **paymentInformation.card.expirationYear/ paymentInformation.tokenizedCard.expirationYear**-set this field to the payment network token expiration year value.
- **processingInformation.commerceIndicator**-set this field to `aesk`.
- **paymentinformation.tokenizedCard.cryptogram**-set this field to the network token cryptogram.
- **paymentInformation.tokenizedCard.transactionType**-set this field to `1`.
- **processingInformation.paymentSolution**-set this field to `008`.

Related Information

REST API Field Reference Guide

Authorizing a Payment

1. Send the service request to `https://api.cybersource.com/pts/v2/payments`.
2. Include the required fields in the request.
3. Include optional fields in the request as needed.
4. Check the response message to make sure that the request was successful. A 200-level HTTP response code indicates success. For information about response codes, see *Transaction Response Codes*.

Example: Merchant Decryption with American Express Using the REST API

Authorization Request

```
{
  "clientReferenceInformation": {
    "code": "demorefnum"
  },
  "consumerAuthenticationInformation": {
    "cavv": "ABCDEFabcdefABCDEFabcdef0987654321234567",
    "xid": "12345678900987654321ABCDEFabcdefABCDEF123"
  },
  "processingInformation": {
    "paymentSolution": "008"
  },
  "paymentInformation": {
```

```
      "tokenizedCard": {
        "expirationMonth": "12",
        "expirationYear": "2021",
        "number": "xxxx8224631Oxxxx",
        "transactionType": "1"
      }
   },
   "billTo": {
      "firstName": "James",
      "lastName": "Smith",
       "address1": "111 S. Division St.",
       "address2": "Suite 123",
       "locality": "Ann Arbor",
       "administrativeArea": "MI",
       "postalCode": "48104-2201",
       "country": "US",
       "email": "demo@example.com",
       "phoneNumber": "9999999999"
    },
    "orderInformation": {
      "amountDetails": {
         "currency": "USD",
         "totalAmount": "100.00"
      }
    }
}
```

Authorization Response

```
{
   "clientReferenceInformation": {
      "code": "demorefnum"
   },
   "orderInformation": {
      "amountDetails": {
         "currency": "USD",
         "authorizedAmount": "100.00"
      }
   },
   "processingInformation": {
      "reconciliationID": "13209256CGJSMQCZ"
   },
   "processorInformation": {
      "approvalCode": "888888",
      "responseCode": "100",
      "avs": {
         "code": "I1"
      }
   },
   "submitTimeUtc": "2015-11-03T205202Z"
   }
}
```

Authorizing a Payment with JCB Using Merchant Decryption Method

This section provides the following information:

- *Required Fields for Authorizing a Payment Using JCB and the Merchant Decryption Method* on page 29
- *Authorizing a Payment* on page 18
- *Example: Merchant Decryption with JCB Using the REST API* on page 29

Required Fields for Authorizing a Payment Using JCB and the Merchant Decryption Method

The following fields are required when submitting an authorization request using the Merchant decryption method:

- **consumerAuthenticationInformation.cavv**–set this field to the 3-D Secure cryptogram of the payment network token.
- **paymentInformation.card.number**–set this field to the payment network token value.
- **paymentInformation.card.expirationMonth/ paymentInformation.tokenizedCard.expirationMonth**–set this field to the payment network token expiration month value.
- **paymentInformation.card.expirationYear/ paymentInformation.tokenizedCard.expirationYear**–set this field to the payment network token expiration year value.
- **consumerAuthenticationInformation.eciRaw**–set this field to the ECI value contained in the Samsung Pay reply message.
- **paymentinformation.tokenizedCard.cryptogram**–set this field to the network token cryptogram.
- **paymentInformation.tokenizedCard.transactionType**–set this field to `1`.
- **processingInformation.paymentSolution**–set this field to `008`.

Related Information

REST API Field Reference Guide

Authorizing a Payment

1. Send the service request to `https://api.cybersource.com/pts/v2/payments`.
2. Include the required fields in the request.
3. Include optional fields in the request as needed.
4. Check the response message to make sure that the request was successful. A 200-level HTTP response code indicates success. For information about response codes, see *Transaction Response Codes*.

Example: Merchant Decryption with JCB Using the REST API

Authorization Request

```
{
  "consumerAuthenticationInformation": {
```

```
      "cavv": "EHuWW9PiBkWvqE5juRwDzAUFBAk=",
      "eciRaw": "05"
    },
    "processingInformation": {
      "paymentSolution": "008"
    },
    "paymentInformation": {
      "tokenizedCard": {
        "expirationMonth": "12",
        "expirationYear": "2031",
        "number": "xxxx11111111xxxx",
        "transactionType": "1",
        "type": "007"
      }
    },
    "billTo": {
      "firstName": "Jane",
      "lastName": "Smith",
      "address1": "123 Main St.",
      "address2": "Suite 12345",
      "locality": "Small Town",
      "administrativeArea": "CA",
      "postalCode": "98765",
      "country": "US",
      "email": "js@example.com",
      "phoneNumber": "9999999999"
    },
    "orderInformation": {
      "amountDetails": {
        "currency": "USD",
        "totalAmount": "100.00"
      }
    }
}
```

Authorization Response

```
{
    "clientReferenceInformation": {
      "code": "ref123"
    },
    "orderInformation": {
      "amountDetails": {
        "currency": "USD",
        "authorizedAmount": "100.00"
      }
    },
    "processingInformation": {
      "reconciliationID": "15356268CR2XF23X"
    },
    "processorInformation": {
      "approvalCode": "888888",
      "responseCode": "100",
      "avs": {
        "code": "X",
        "codeRaw": "I1"
```

```
    }
  }
}
```

Authorizing a Payment with Mastercard Using Merchant Decryption Method

This section provides the following information:

- *Required Fields for Authorizing a Payment Using Mastercard and the Merchant Decryption Method* on page 31
- *Authorizing a Payment* on page 18
- *Example: Merchant Decryption with Mastercard Using the REST API* on page 32

Required Fields for Authorizing a Payment Using Mastercard and the Merchant Decryption Method

The following fields are required when submitting an authorization request using the Merchant decryption method:

- **paymentInformation.card.number**-set this field to the payment network token value.
- **paymentInformation.card.expirationMonth/ paymentInformation.tokenizedCard.expirationMonth**-set this field to the payment network token expiration month value.
- **paymentInformation.card.expirationYear/ paymentInformation.tokenizedCard.expirationYear**-set this field to the payment network token expiration year value.
- **processingInformation.commerceIndicator**- set this field to `spa`.
- **paymentinformation.tokenizedCard.cryptogram**-set this field to the network token cryptogram.
- **paymentInformation.tokenizedCard.transactionType**-set this field to `1`.
- **processingInformation.paymentSolution**-set this field to `008`.
- **consumerAuthenticationInformation.ucafAuthenticationData**–set this field to the 3-D Secure cryptogram of the payment network token.
- **consumerAuthenticationInformation.ucafCollectionIndicator**-set this field to `2`.

Related Information

REST API Field Reference Guide

Authorizing a Payment

1. Send the service request to `https://api.cybersource.com/pts/v2/payments`.
2. Include the required fields in the request.
3. Include optional fields in the request as needed.
4. Check the response message to make sure that the request was successful. A 200-level HTTP response code indicates success. For information about response codes, see *Transaction Response Codes*.

Example: Merchant Decryption with Mastercard Using the REST API

Authorization Request

```
{
  "clientReferenceInformation": {
    "code": "demorefnum"
  },
  "consumerAuthenticationInformation": {
    "ucafAuthenticationData": "ABCDEFabcdefABCDEFabcdef0987654321234567",
    "ucafCollectionIndicator": "2"
  },
  "processingInformation": {
    "paymentSolution": "008"
  },
  "paymentInformation": {
    "tokenizedCard": {
      "expirationMonth": "12",
      "expirationYear": "2021",
      "number": "xxxx55555555xxxx",
      "transactionType": "1"
    }
  },
  "billTo": {
    "firstName": "James",
    "lastName": "Smith",
     "address1": "111 S. Division St.",
     "address2": "Suite 123",
     "locality": "Ann Arbor",
     "administrativeArea": "MI",
     "postalCode": "48104-2201",
     "country": "US",
     "email": "demo@example.com",
     "phoneNumber": "9999999999"
   },
   "orderInformation": {
     "amountDetails": {
       "currency": "USD",
       "totalAmount": "100.00"
     }
   }
}
```

Authorization Response

```
{
  "clientReferenceInformation": {
    "code": "demorefnum"
  },
  "orderInformation": {
    "amountDetails": {
      "currency": "USD",
      "authorizedAmount": "100.00"
    }
  },
  "processingInformation": {
```

```
    "reconciliationID": "13209255CGJSMQCR"
  },
  "processorInformation": {
    "approvalCode": "888888",
    "responseCode": "100",
    "avs": {
      "code": "I1"
    }
  },
  "submitTimeUtc": "2015-11-03T205035Z"
  }
}
```

Authorizing a Payment with Visa Using Merchant Decryption Method

This section provides the following information:

- *Required Fields for Authorizing a Payment Using Visa and the Merchant Decryption Method* on page 33
- *Authorizing a Payment* on page 18
- *Example: Merchant Decryption with Visa Using the REST API* on page 34

Required Fields for Authorizing a Payment Using Visa and the Merchant Decryption Method

The following fields are required when submitting an authorization request using the Merchant decryption method:

- **consumerAuthenticationInformation.cavv**-set this field to the 3-D Secure cryptogram of the payment network token.
- **paymentInformation.card.number**-set this field to the payment network token value.
- **paymentInformation.card.expirationMonth/ paymentInformation.tokenizedCard.expirationMonth**-set this field to the payment network token expiration month value.
- **paymentInformation.card.expirationYear/ paymentInformation.tokenizedCard.expirationYear**-set this field to the payment network token expiration year value.
- **consumerAuthenticationInformation.eciRaw**-for JCB transactions, set this field to the ECI value contained in the Samsung Pay reply message.
- **processingInformation.commerceIndicator**-set this field to `internet`.
- **paymentinformation.tokenizedCard.cryptogram**-set this field to the network token cryptogram.
- **paymentInformation.tokenizedCard.transactionType**-set this field to `1`.
- **processingInformation.paymentSolution**-set this field to `008`.

Related Information

REST API Field Reference Guide

Authorizing a Payment

1. Send the service request to https://api.cybersource.com/pts/v2/payments.
2. Include the required fields in the request.
3. Include optional fields in the request as needed.
4. Check the response message to make sure that the request was successful. A 200-level HTTP response code indicates success. For information about response codes, see *Transaction Response Codes*.

Example: Merchant Decryption with Visa Using the REST API

Authorization Request

```
{
  "clientReferenceInformation": {
    "code": "demorefnum"
  },
  "consumerAuthenticationInformation": {
    "cavv": "ABCDEFabcdefABCDEFabcdef0987654321234567"
  },
  "processingInformation": {
    "commerceIndicator": "internet",
    "paymentSolution": "008"
  },
  "paymentInformation": {
    "tokenizedCard": {
      "expirationMonth": "12",
      "expirationYear": "2021",
      "number": "xxxx100000000xxxx",
      "transactionType": "1"
    }
  },
  "billTo": {
    "firstName": "James",
    "lastName": "Smith",
    "address1": "111 S. Division St.",
    "address2": "Suite 123",
    "locality": "Ann Arbor",
    "administrativeArea": "MI",
    "postalCode": "48104-2201",
    "country": "US",
    "email": "demo@example.com",
    "phoneNumber": "9999999999"
   },
   "orderInformation": {
    "amountDetails": {
      "currency": "USD",
      "totalAmount": "100.00"
    }
   }
}
```

Authorization Response

```
{
```

```
  "clientReferenceInformation": {
    "code": "demorefnum"
  },
  "orderInformation": {
    "amountDetails": {
      "currency": "USD",
      "authorizedAmount": "100.00"
    }
  },
  "processingInformation": {
    "reconciliationID": "13209254CGJSMQCQ"
  },
  "processorInformation": {
    "approvalCode": "888888",
    "responseCode": "100",
    "avs": {
      "code": "I1"
    }
  },
  "submitTimeUtc": "2015-11-03T205035Z"
  }
}
```

Authorization Reversal Service

The authorization reversal service is a follow-on service that uses the request ID returned from the previous authorization. An authorization reversal releases the hold that the authorization placed on the customer's credit card funds. Use this service to reverse an unnecessary or undesired authorization.

Processor-Specific Information About Authorization Reversals

Payment Processor	Authorization Reversal Information
Barclays	You are responsible for complying with the processor's sp ecific requirements for full authorization reversals. Co ntact the processor for more information. Enhanced a uthorization reversals are supported on this processor; th erefore, extra data is included in the authorization rev ersal request. It is not necessary to process or monito r the extra data.

Related concepts

- Required Fields for Reversing an Authorization on page 36

Related tasks

- Reversing an Authorization on page 36

Required Fields for Reversing an Authorization

The following fields are required when creating an authorization reversal request:

clientReferenceInformation.code	
orderInformation.amountDetails.currency	
orderInformation.amountDetails.totalAmount	
processingInformation.paymentSolution	Set to `008`.

Related Information

REST API Field Reference Guide

Reversing an Authorization

1. Send the service request to `POST https://<url_prefix>/v2/payments/{id}/reversals`. Use one of these prefixes:

   - Test: `apitest.cybersource.com`
   - Production: `api.cybersource.com`
   - Production in India: `api.in.cybersource.com`

   Where `id` is the authorization ID returned in the authorization response.

   ```
   {
    "id": "6481692924466004003001"
   }
   ```

   The URL with the `id` value is included in the authorization response:

   ```
   {
     "_links": {
       "authReversal": {
         "method": "POST",
         "href": "/pts/v2/payments/6481692924466004003001/reversals"
       },
   ```

2. Check the response message to make sure that the request was successful. A 200-level HTTP response code indicates success. For information about response codes, see *Transaction Response Codes*.

Example: Basic Credit Card Authorization Reversal Using the REST API

Authorization Reversal Request

```
{
 "clientReferenceInformation": {
  "code": "TC50171_3"
 },
 "reversalInformation": {
```

```
    "amountDetails": {
     "totalAmount": "102.21"
    },
    "reason": "exception"
  }
}
```

Authorization Reversal Response

```
{
   "submitTimeUtc": "2021-04-22T16:44:03Z",
   "status": "approved",
   "errorInformation": {
      "reason": "EXCEPTION",
      "message": "The request was processed successfully."
   }
}
```

Capture Service

The capture service is a follow-on service that uses the request ID returned from the previous authorization. The request ID links the capture to the authorization. This service transfers funds from the customer's account to your bank and usually takes two to four days to complete.

Processor-Specific Information About Authorizations and Captures

Payment Processor	Authorization and Capture Information
Barclays	The amount is rounded to the correct number of decimal p laces for the currency. Barclays does not support amounts of 0.01. Barclays supports zero amount authorizations and amounts greater than 0.01.Barclays supports enhanced re sponse codes in authorization response messages. Enhanc ed response codes provide detailed information about dec lined transactions. Contact Barclays customer support to have this capability enabled for your account.

Related concepts
- Required Fields for Capturing a Payment on page 37

Related tasks
- Capturing a Payment on page 38

Required Fields for Capturing a Payment

The following fields are required when creating a capture request:

clientReferenceInformation.code

orderInformation.amountDetails.currency	
orderInformation.amountDetails.totalAmount	
processingInformation.paymentSolution	Set to `008`.

Related Information

REST API Field Reference Guide

Capturing a Payment

1. Pass the original authorization ID in the URL, and send the service request to `POST https://<url_prefix>/v2/payments/{id}/captures`. Use one of these URL prefixes:

   - Test: `apitest.cybersource.com`
   - Production: `api.cybersource.com`
   - Production in India: `api.in.cybersource.com`

   Where `id` is the authorization ID returned in the authorization response.

   ```
   {
    "id": "6481692924466004003001"
   }
   ```

   The URL with the `id` value is included in the authorization response:

   ```
   {
    "_links": {
     "capture": {
      "method": "POST",
      "href": "/pts/v2/payments/6481692924466004003001/captures"
     }
    }
   ```

2. Check the response message to make sure that the request was successful. A 200-level HTTP response code indicates success. For information about response codes, see *Transaction Response Codes*.

Example: Basic Credit Card Capture Using the REST API

Capture Request

```
{
 "clientReferenceInformation": {
  "code": "482046C3A7E94F5BD1FE3C66C"
 },
 "processingInformation": {
  "paymentSolution": "008"
 },
 "orderInformation": {
  "amountDetails": {
   "totalAmount": "49.95",
   "currency": "USD"
```

```
  }
 }
}
```

Capture Response

```
{
  "clientReferenceInformation": {
    "code": "482046C3A7E94F5BD1FE3C66C"
  },
  "processingInformation": {
    "reconciliationID": "02850840187309570"
  },
  "orderInformation": {
    "amountDetails": {
      "totalAmount": "49.95",
      "currency": "USD"
    }
  }
}
```

Sale Service

A sale is a bundled authorization and capture. Request the authorization and capture services at the same time. Cybersource processes the capture immediately.

Required Fields for Performing a Sale

The following fields are required when submitting a sale request:

Fields required for requesting the authorization service	Use the same values that are set for requesting the *Authorization Service* on page 17.

The sales request is sent to the following endpoint: https://api.cybersource.com/pts/v2/payments.

Related Information

REST API Field Reference Guide

Authorizing and Capturing a Payment

You can authorize and capture a payment at the same time, which is known as performing a sale.

1. Send the service request to https://api.cybersource.com/pts/v2/payments.
2. Check the response message to make sure that the request was successful. A 200-level HTTP response code indicates success. For information about response codes, see *Transaction Response Codes*.

Example: Basic Credit Card Sale Using the REST API

Authorization and Capture (Sale) Request

```
{
  "clientReferenceInformation": {
    "code": "demorefnum"
  },
  "processingInformation": {
    "paymentSolution": "008",
    "commerceIndicator": "aesk"
  },
  "paymentInformation": {
    "tokenizedCard": {
      "transactionType": "1",
      "type": "003"
    }
  },
  "fluidData": {
   "descriptor": "ABCDEFabcdefABCDEFabcdef0987654321234567",
   "value": "RklEPUNPTU1PTi5TQU1TVU5HLklOQVBQLlBBWU1FTlQ="
  },
  "billTo": {
    "firstName": "James",
    "lastName": "Smith",
     "address1": "111 S. Division St.",
     "address2": "Suite 123",
     "locality": "Ann Arbor",
     "administrativeArea": "MI",
     "postalCode": "48104-2201",
     "country": "US",
     "email": "demo@example.com",
     "phoneNumber": "9999999999"
   },
   "orderInformation": {
     "amountDetails": {
       "currency": "USD",
       "totalAmount": "100.00"
     }
   }
}
```

Authorization and Capture (Sale) Response

```
{
  "clientReferenceInformation": {
    "code": "demorefnum"
  },
  "orderInformation": {
    "amountDetails": {
      "currency": "USD",
      "authorizedAmount": "100.00",
      "totalAmount": "100.00"
    }
  },
  "paymentInformation": {
```

```
      "tokenizedCard": {
        "prefix": "593056",
        "suffix": "0842",
        "expirationMonth": "08",
        "expirationYear": "2021"
      }
    },
    "processingInformation": {
      "reconciliationID": "13209256CGJSMQCZ"
    },
    "processorInformation": {
      "approvalCode": "888888",
      "responseCode": "100",
      "avs": {
        "code": "I1"
      }
    },
    "submitTimeUtc": "2015-11-03T205202Z"
    }
}
```