Samsung Pay

Simple Order API
FDC Compass

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 Unpublished rights reserved under the copyright laws of the United States.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

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
Removed American Express and Discover as supported card types.

22.03
This revision contains only editorial changes and no technical updates.

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

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
FDC Compass	JCB Master card 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 31
- *Capture Service* on page 32
- *Sale Service* on page 34

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

Related concepts

- Authorizing a Payment with American Express Using Cybersource Decryption Method
- Authorizing a Payment with JCB Using Cybersource Decryption Method on page 18
- Authorizing a Payment with Mastercard Using Cybersource Decryption Method on page 19
- Authorizing a Payment with Visa Using Cybersource Decryption Method on page 22
- Authorizing a Payment with American Express Using Merchant Decryption Method
- Authorizing a Payment with JCB Using Merchant Decryption Method on page 24
- Authorizing a Payment with Mastercard Using Merchant Decryption Method on page 26
- Authorizing a Payment with Visa Using Merchant Decryption Method on page 28

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

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

- **encryptedPayment_data**-set this field to the Base64-encoded value obtained from the **paymentData** property of the **PKPaymentToken** object.
- **encryptedPayment_descriptor**-set this field to `RklEPUNPTU1PTi5TQU1TVU5HLklOQVBQLlBBWU1FTlQ=`.
- **paymentSolution**-set this field to `008`.

Related Information

Simple Order API Field Reference Guide

Authorizing a Payment

1. Request the service. Set the **ccAuthService_run** field to `true`, and send the request to one of these endpoints:
   - *https://ics2ws.ic3.com/commerce/1.x/transactionProcessor*
   - *https://ics2wsa.ic3.com/commerce/1.x/transactionProcessor*
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 value of `ACCEPT` for the **decision** field indicates success. For information about reason codes, see *Reason Codes for the Simple Order API*.

Example: Cybersource Decryption with JCB Using the Simple Order API

Authorization Request

```
<requestMessage xmlns="urn:schemas-cybersource-com:transaction-data-1.121">
  <merchantID>demomerchant</merchantID>
  <merchantReferenceCode>demorefnum</merchantReferenceCode>
  <billTo>
    <firstName>Jane</firstName>
    <lastName>Smith</lastName>
    <street1>123 Main Street</street1>
    <city>Small Town</city>
```

```
    <state>CA</state>
    <postalCode>98765</postalCode>
    <country>US</country>
    <email>jsmith@example.com</email>
  </billTo>
  <purchaseTotals>
    <currency>USD</currency>
    <grandTotalAmount>5.00</grandTotalAmount>
  </purchaseTotals>
  <encryptedPayment>
    <descriptor>RklEPUNPTU1PTi5TQU1TVU5HLklOQVBQLlBBWU1FTlQ=</descriptor>
    <data>ABCDEFabcdefABCDEFabcdef0987654321234567</data>
    <encoding>Base64</encoding>
  </encryptedPayment>
  <card>
    <cardType>007</cardType>
  </card>
  <ccAuthService run="true"/>
    <paymentSolution>008</paymentSolution>
</requestMessage>
```

Authorization Response

```
<c:replyMessage>
  <c:merchantReferenceCode>demorefnum</c:merchantReferenceCode>
  <c:requestID>4465840340765000001541</c:requestID>
  <c:decision>ACCEPT</c:decision>
  <c:reasonCode>100</c:reasonCode>
  <c:token>
    <c:expirationMonth>07</c:expirationMonth>
    <c:expirationYear>2025</c:expirationYear>
    <c:prefix>239845</c:prefix>
    <c:suffix>2947</c:suffix>
  </c:token>
  <c:purchaseTotals>
    <c:currency>USD</c:currency>
  </c:purchaseTotals>
  <c:ccAuthReply>
    <c:reasonCode>100</c:reasonCode>
    <c:amount>5.00</c:amount>
    <c:authorizationCode>888888</c:authorizationCode>
    <c:avsCode>X</c:avsCode>
    <c:avsCodeRaw>I1</c:avsCodeRaw>
    <c:processorResponse>100</c:processorResponse>
    <c:reconciliationID>11267051CGJSMQDC</c:reconciliationID>
  </c:ccAuthReply>
</c:replyMessage>
```

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 20

- *Authorizing a Payment* on page 18
- *Example: Cybersource Decryption with Mastercard Using the Simple Order API* on page 20

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:

- **ccAuthService_commerceIndicator**-set this field to `spa`.
- **encryptedPayment_data**
  - 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 **encryptedPayment_data** field.
- **encryptedPayment_descriptor**-set this field to `RklEPUNPTU1PTi5TQU1TVU5HLklOQVBQLlBBWU1FTlQ=`.
- **paymentNetworkToken_transactionType**-set this field to `1`.
- **paymentSolution**-set this field to `008`.

Related Information

Simple Order API Field Reference Guide

Authorizing a Payment

1. Request the service. Set the **ccAuthService_run** field to `true`, and send the request to one of these endpoints:
   - *https://ics2ws.ic3.com/commerce/1.x/transactionProcessor*
   - *https://ics2wsa.ic3.com/commerce/1.x/transactionProcessor*
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 value of `ACCEPT` for the **decision** field indicates success. For information about reason codes, see *Reason Codes for the Simple Order API*.

Example: Cybersource Decryption with Mastercard Using the Simple Order API

Authorization Request

```
<requestMessage xmlns="urn:schemas-cybersource-com:transaction-data-1.121">
  <merchantID>demomerchant</merchantID>
  <merchantReferenceCode>demorefnum</merchantReferenceCode>
  <billTo>
```

```
      <firstName>James</firstName>
      <lastName>Smith</lastName>
      <street1>1295 Charleston Road</street1>
      <city>Test City</city>
      <state>CA</state>
      <postalCode>99999</postalCode>
      <country>US</country>
      <email>demo@example.com</email>
    </billTo>
    <purchaseTotals>
      <currency>USD</currency>
      <grandTotalAmount>5.00</grandTotalAmount>
    </purchaseTotals>
    <ccAuthService run="true">
      <commerceIndicator>spa</commerceIndicator>
    </ccAuthService>
    <encryptedPayment>
      <data>ABCDEFabcdefABCDEFabcdef0987654321234567</data>
      <descriptor>RklEPUNPTU1PTi5TQU1TVU5HLklOQVBQLlBBWU1FTlQ=</descriptor>
    </encryptedPayment>
    <paymentSolution>008</paymentSolution>
    <paymentNetworkToken>
      <transactionType>1</transactionType>
    </paymentNetworkToken>
</requestMessage>
```

Authorization Response

```
<c:replyMessage>
    <c:merchantReferenceCode>demorefnum</c:merchantReferenceCode>
    <c:requestID>4465840340765000001541</c:requestID>
    <c:decision>ACCEPT</c:decision>
    <c:reasonCode>100</c:reasonCode>
    <c:purchaseTotals>
      <c:currency>USD</c:currency>
    </c:purchaseTotals>
    <c:ccAuthReply>
      <c:reasonCode>100</c:reasonCode>
      <c:amount>5.00</c:amount>
      <c:authorizationCode>888888</c:authorizationCode>
      <c:avsCode>X</c:avsCode>
      <c:avsCodeRaw>I1</c:avsCodeRaw>
      <c:authorizedDateTime>2015-11-03T20:53:54Z</c:authorizedDateTime>
      <c:processorResponse>100</c:processorResponse>
      <c:reconciliationID>11267051CGJSMQDC</c:reconciliationID>
    </c:ccAuthReply>
    <c:token>
      <c:prefix>128945</c:prefix>
      <c:suffix>2398</c:suffix>
      <c:expirationMonth>08</c:expirationMonth>
      <c:expirationYear>2021</c:expirationYear>
    </c:token>
</c:replyMessage>
```

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

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

- **ccAuthService_commerceIndicator**-set this field to `internet`.
- **encryptedPayment_data**
  - 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 **encryptedPayment_data** field.
- **encryptedPayment_descriptor**-set this field to `RklEPUNPTU1PTi5TQU1TVU5HLklOQVBQLlBBWU1FTlQ=`.
- **paymentNetworkToken_transactionType**-set this field to `1`.
- **paymentSolution**-set this field to `008`.

Related Information

Simple Order API Field Reference Guide

Authorizing a Payment

1. Request the service. Set the **ccAuthService_run** field to `true`, and send the request to one of these endpoints:
   - *https://ics2ws.ic3.com/commerce/1.x/transactionProcessor*
   - *https://ics2wsa.ic3.com/commerce/1.x/transactionProcessor*
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 value of `ACCEPT` for the **decision** field indicates success. For information about reason codes, see *Reason Codes for the Simple Order API*.

Example: Cybersource Decryption with Visa Using the Simple Order API

Authorization Request

```
<requestMessage xmlns="urn:schemas-cybersource-com:transaction-data-1.121">
  <merchantID>demomerchant</merchantID>
  <merchantReferenceCode>demorefnum</merchantReferenceCode>
  <billTo>
    <firstName>James</firstName>
    <lastName>Smith</lastName>
    <street1>1295 Charleston Road</street1>
    <city>Test City</city>
    <state>CA</state>
    <postalCode>99999</postalCode>
    <country>US</country>
    <email>demo@example.com</email>
  </billTo>
  <purchaseTotals>
    <currency>USD</currency>
    <grandTotalAmount>5.00</grandTotalAmount>
  </purchaseTotals>
  <ccAuthService run="true">
    <commerceIndicator>internet</commerceIndicator>
  </ccAuthService>
  <encryptedPayment>
    <data>ABCDEFabcdefABCDEFabcdef0987654321234567</data>
    <descriptor>RklEPUNPTU1PTi5TQU1TVU5HLklOQVBQLlBBWU1FTlQ=</descriptor>
  </encryptedPayment>
  <paymentSolution>008</paymentSolution>
  <paymentNetworkToken>
    <transactionType>1</transactionType>
  </paymentNetworkToken>
</requestMessage>
```

Authorization Response

```
<c:replyMessage>
  <c:merchantReferenceCode>demorefnum</c:merchantReferenceCode>
  <c:requestID>4465840340765000001541</c:requestID>
  <c:decision>ACCEPT</c:decision>
  <c:reasonCode>100</c:reasonCode>
  <c:purchaseTotals>
    <c:currency>USD</c:currency>
  </c:purchaseTotals>
  <c:ccAuthReply>
    <c:reasonCode>100</c:reasonCode>
    <c:amount>5.00</c:amount>
    <c:authorizationCode>888888</c:authorizationCode>
    <c:avsCode>X</c:avsCode>
    <c:avsCodeRaw>I1</c:avsCodeRaw>
    <c:authorizedDateTime>2015-11-03T20:53:54Z</c:authorizedDateTime>
    <c:processorResponse>100</c:processorResponse>
    <c:reconciliationID>11267051CGJSMQDC</c:reconciliationID>
  </c:ccAuthReply>
  <c:token>
    <c:prefix>294672</c:prefix>
```

```
    <c:suffix>4397</c:suffix>
    <c:expirationMonth>08</c:expirationMonth>
    <c:expirationYear>2021</c:expirationYear>
  </c:token>
</c:replyMessage>
```

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 24
- *Authorizing a Payment* on page 18
- *Example: Merchant Decryption with JCB Using the Simple Order API* on page 25

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:

- **ccAuthService_cavv**-set this field to the 3-D Secure cryptogram of the payment network token.
- **card_accountNumber**-set this field to the payment network token value.
- **card_expirationMonth**-set this field to the payment network token expiration month value.
- **card_expirationYear**-set this field to the payment network token expiration year value.
- **ccAuthService_eciRaw**-set this field to the ECI value contained in the Samsung Pay reply message.
- **ccAuthService_networkTokenCryptogram**-set this field to the network token cryptogram.
- **paymentNetworkToken_transactionType**-set this field to `1`.
- **paymentSolution**-set this field to `008`.

Related Information

Simple Order API Field Reference Guide

Authorizing a Payment

1. Request the service. Set the **ccAuthService_run** field to `true`, and send the request to one of these endpoints:
   - *https://ics2ws.ic3.com/commerce/1.x/transactionProcessor*
   - *https://ics2wsa.ic3.com/commerce/1.x/transactionProcessor*
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 value of ACCEPT for the **decision** field indicates success. For information about reason codes, see *Reason Codes for the Simple Order API*.

Example: Merchant Decryption with JCB Using the Simple Order API

Authorization Request

```
<requestMessage xmlns="urn:schemas-cybersource-com:transaction-data-1.121">
  <merchantID>demomerchant</merchantID>
  <merchantReferenceCode>demorefnum</merchantReferenceCode>
  <billTo>
    <firstName>Jane</firstName>
    <lastName>Smith</lastName>
    <street1>123 Main Street</street1>
    <city>Small Town</city>
    <state>CA</state>
    <postalCode>98765</postalCode>
    <country>US</country>
    <email>jsmith@example.com</email>
  </billTo>
  <purchaseTotals>
    <currency>USD</currency>
    <grandTotalAmount>5.00</grandTotalAmount>
  </purchaseTotals>
  <card>
    <accountNumber>xxxx11111111xxxx</accountNumber>
    <expirationMonth>12</expirationMonth>
    <expirationYear>2020</expirationYear>
    <cvNumber>123</cvNumber>
    <cardType>007</cardType>
  </card>
  <ccAuthService run="true">
    <cavv>ABCDEFabcdefABCDEFabcdef0987654321234567</cavv>
    <eciRaw>5</eciRaw>
  </ccAuthService>
  <paymentNetworkToken>
    <transactionType>1</transactionType>
  </paymentNetworkToken>
  <paymentSolution>008</paymentSolution>
</requestMessage>
```

Authorization Response

```
<c:replyMessage>
  <c:merchantReferenceCode>demorefnum</c:merchantReferenceCode>
  <c:requestID>4465840340765000001541</c:requestID>
  <c:decision>ACCEPT</c:decision>
  <c:reasonCode>100</c:reasonCode>
  <c:purchaseTotals>
    <c:currency>USD</c:currency>
  </c:purchaseTotals>
  <c:ccAuthReply>
    <c:reasonCode>100</c:reasonCode>
    <c:amount>5.00</c:amount>
    <c:authorizationCode>888888</c:authorizationCode>
```

```
    <c:avsCode>X</c:avsCode>
    <c:avsCodeRaw>I1</c:avsCodeRaw>
    <c:authorizedDateTime>2015-11-03T20:53:54Z</c:authorizedDateTime>
    <c:processorResponse>100</c:processorResponse>
    <c:reconciliationID>11267051CGJSMQDC</c:reconciliationID>
  </c:ccAuthReply>
</c:replyMessage>
```

NVP Request

```
merchantID=demomerchant
merchantReferenceCode=demorefnum
billTo_firstName=Jane
billTo_lastName=Smith
billTo_street1=123 Main Street
billTo_city=Small Town
billTo_state=CA
billTo_postalCode=98765
billTo_country=US
billTo_email=jsmith@example.com
purchaseTotals_currency=USD
purchastTotals_grandTotalAmount=5.00
card_accountNumber=xxxx00202036xxxx
card_expirationYear=2020
card_cvnNumber=123
cardType=007
ccAuthService_cavv=ABCDEFabcdefABCDEFabcdef0987654321234567
ccAuthService_eciRaw=5
paymentNetworkToken_transactionType=1
paymentSolution=008
```

NVP Response

```
merchantReferenceCode=demorefnum
requestID=4465840340765000001541
decision=accept
reasonCode=100
requestToken=Ahj/7wSR5C/4Icd2fdAKakGLadfg5535r/ghx3Z90AoBj3u
purchaseTotals_currency=USD
ccAuthReply_reasonCode=100
ccAuthReply_amount=5.00
ccAuthReply_authorizationCode=888888
ccAuthReply_avsCode=X
ccAuthReply_avsCodeRaw=I1
ccAuthReply_authorizedDateTime=2015-11-03T20:53:54Z
ccAuthReply_processorResponse=100
ccAuthReply_reconciliationID=11267051CGJSMQDC
```

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 27

- *Authorizing a Payment* on page 18
- *Example: Merchant Decryption with Mastercard Using the Simple Order API* on page 27

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:

- **card_accountNumber**-set this field to the payment network token value.
- **card_expirationMonth**-set this field to the payment network token expiration month value.
- **card_expirationYear**-set this field to the payment network token expiration year value.
- **ccAuthService_commerceIndicator**- set this field to `spa`.
- **ccAuthService_networkTokenCryptogram**-set this field to the network token cryptogram.
- **paymentNetworkToken_transactionType**-set this field to `1`.
- **paymentSolution**-set this field to `008`.
- **ucaf_authenticationData**–set this field to the 3-D Secure cryptogram of the payment network token.
- **ucaf_collectionIndicator**-set this field to `2`.

Related Information

Simple Order API Field Reference Guide

Authorizing a Payment

1. Request the service. Set the **ccAuthService_run** field to `true`, and send the request to one of these endpoints:
   - *https://ics2ws.ic3.com/commerce/1.x/transactionProcessor*
   - *https://ics2wsa.ic3.com/commerce/1.x/transactionProcessor*
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 value of `ACCEPT` for the **decision** field indicates success. For information about reason codes, see *Reason Codes for the Simple Order API*.

Example: Merchant Decryption with Mastercard Using the Simple Order API

Authorization Request

```
<requestMessage xmlns="urn:schemas-cybersource-com:transaction-data-1.121">
  <merchantID>demomerchant</merchantID>
  <merchantReferenceCode>demorefnum</merchantReferenceCode>
  <billTo>
    <firstName>James</firstName>
    <lastName>Smith</lastName>
```

```
      <street1>1295 Charleston Road</street1>
      <city>Test City</city>
      <state>CA</state>
      <postalCode>99999</postalCode>
      <country>US</country>
      <email>demo@example.com</email>
    </billTo>
    <purchaseTotals>
      <currency>USD</currency>
      <grandTotalAmount>5.00</grandTotalAmount>
    </purchaseTotals>
    <card>
      <accountNumber>xxxx55555555xxxx</accountNumber>
      <expirationMonth>12</expirationMonth>
      <expirationYear>2020</expirationYear>
    </card>
    <ucaf>
      <authenticationData>ABCDEFabcdefABCDscdef0987654321234567</authenticationData>
      <collectionIndicator>2</collectionIndicator>
    </ucaf>
    <ccAuthService run="true">
      <commerceIndicator>spa</commerceIndicator>
    </ccAuthService>
    <paymentNetworkToken>
      <transactionType>1</transactionType>
    </paymentNetworkToken>
    <paymentSolution>008</paymentSolution>
</requestMessage>
```

Authorization Response

```
<c:replyMessage>
  <c:merchantReferenceCode>demorefnum</c:merchantReferenceCode>
  <c:requestID>4465840340765000001541</c:requestID>
  <c:decision>ACCEPT</c:decision>
  <c:reasonCode>100</c:reasonCode>
  <c:purchaseTotals>
    <c:currency>USD</c:currency>
  </c:purchaseTotals>
  <c:ccAuthReply>
    <c:reasonCode>100</c:reasonCode>
    <c:amount>5.00</c:amount>
    <c:authorizationCode>888888</c:authorizationCode>
    <c:avsCode>X</c:avsCode>
    <c:avsCodeRaw>I1</c:avsCodeRaw>
    <c:authorizedDateTime>2015-11-03T20:53:54Z</c:authorizedDateTime>
    <c:processorResponse>100</c:processorResponse>
    <c:reconciliationID>11267051CGJSMQDC</c:reconciliationID>
  </c:ccAuthReply>
</c:replyMessage>
```

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 29
- *Authorizing a Payment* on page 18
- *Example: Merchant Decryption with Visa Using the Simple Order API* on page 29

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:

- **ccAuthService_cavv**-set this field to the 3-D Secure cryptogram of the payment network token.
- **card_accountNumber**-set this field to the payment network token value.
- **card_expirationMonth**-set this field to the payment network token expiration month value.
- **card_expirationYear**-set this field to the payment network token expiration year value.
- **ccAuthService_eciRaw**-for JCB transactions, set this field to the ECI value contained in the Samsung Pay reply message.
- **ccAuthService_commerceIndicator**-set this field to `internet`.
- **ccAuthService_networkTokenCryptogram**-set this field to the network token cryptogram.
- **paymentNetworkToken_transactionType**-set this field to `1`.
- **paymentSolution**-set this field to `008`.

Related Information

Simple Order API Field Reference Guide

Authorizing a Payment

1. Request the service. Set the **ccAuthService_run** field to `true`, and send the request to one of these endpoints:
   - *https://ics2ws.ic3.com/commerce/1.x/transactionProcessor*
   - *https://ics2wsa.ic3.com/commerce/1.x/transactionProcessor*
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 value of `ACCEPT` for the **decision** field indicates success. For information about reason codes, see *Reason Codes for the Simple Order API*.

Example: Merchant Decryption with Visa Using the Simple Order API

Authorization Request

```
<requestMessage xmlns="urn:schemas-cybersource-com:transaction-data-1.121">
  <merchantID>demomerchant</merchantID>
  <merchantReferenceCode>demorefnum</merchantReferenceCode>
  <billTo>
```

```
      <firstName>James</firstName>
      <lastName>Smith</lastName>
      <street1>1295 Charleston Road</street1>
      <city>Test City</city>
      <state>CA</state>
      <postalCode>99999</postalCode>
      <country>US</country>
      <email>demo@example.com</email>
    </billTo>
    <purchaseTotals>
      <currency>USD</currency>
      <grandTotalAmount>5.00</grandTotalAmount>
    </purchaseTotals>
    <card>
      <accountNumber>xxxx10000000xxxx</accountNumber>
      <expirationMonth>12</expirationMonth>
      <expirationYear>2020</expirationYear>
    </card>
    <ccAuthService run="true">
      <cavv>ABCDEFabcdefABCDEFabcdef0987654321234567</cavv>
      <commerceIndicator>internet</commerceIndicator>
    </ccAuthService>
    <paymentNetworkToken>
      <transactionType>1</transactionType>
    </paymentNetworkToken>
    <paymentSolution>008</paymentSolution>
</requestMessage>
```

Authorization Response

```
<c:replyMessage>
    <c:merchantReferenceCode>demorefnum</c:merchantReferenceCode>
    <c:requestID>4465840340765000001541</c:requestID>
    <c:decision>ACCEPT</c:decision>
    <c:reasonCode>100</c:reasonCode>
    <c:purchaseTotals>
      <c:currency>USD</c:currency>
    </c:purchaseTotals>
    <c:ccAuthReply>
      <c:reasonCode>100</c:reasonCode>
      <c:amount>5.00</c:amount>
      <c:authorizationCode>888888</c:authorizationCode>
      <c:avsCode>X</c:avsCode>
      <c:avsCodeRaw>I1</c:avsCodeRaw>
      <c:authorizedDateTime>2015-11-03T20:53:54Z</c:authorizedDateTime>
      <c:processorResponse>100</c:processorResponse>
      <c:reconciliationID>11267051CGJSMQDC</c:reconciliationID>
    </c:ccAuthReply>
</c:replyMessage>
```

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
FDC Compass	Card types supported for full authorization reversals: Visa, Mastercard, American Express, Discover, Diners Club, and JCB. A full authorization reversal must occur within three days of the authorization.

Related concepts
- Required Fields for Reversing an Authorization on page 31

Related tasks
- Reversing an Authorization on page 32

Required Fields for Reversing an Authorization

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

ccAuthReversalService_authRequestID	Set to the request ID that was included in the authorization reply message.
ccAuthReversalService_run	Set to `true`.
merchantID	
merchantReferenceCode	
paymentSolution	Set to `008`.
purchaseTotals_currency	
purchaseTotals_grandTotalAmount	Either **purchaseTotals_grandTotalAmount** or **item_#_unitPrice** must be included in the request.

Related Information

Simple Order API Field Reference Guide

Reversing an Authorization

1. Request the service. Set the **ccAuthReversalService_run** field to `true`, and send the request to one of these endpoints:
   - `https://ics2ws.ic3.com/commerce/1.x/transactionProcessor`
   - `https://ics2wsa.ic3.com/commerce/1.x/transactionProcessor`
2. Check the response message to make sure that the request was successful. A value of `ACCEPT` for the **decision** field indicates success. For information about reason codes, see *Reason Codes for the Simple Order API*.

XML Example: Basic Credit Card Authorization Reversal Using the Simple Order API

Authorization Reversal Request

```
<requestMessage xmlns="urn:schemas-cybersource-com:transaction-data-1.121">
  <merchantID>retail_910</merchantID>
  <merchantReferenceCode>MS299131501003</merchantReferenceCode>
  <purchaseTotals>
    <currency>USD</currency>
    <grandTotalAmount>99.49</grandTotalAmount>
  </purchaseTotals>
  <ccAuthReversalService run="true">
    <authRequestID>6152173358406291304007</authRequestID>
  </ccAuthReversalService>
  <paymentSolution>008</paymentSolution>
</requestMessage>
```

Authorization Reversal Response

```
<c:replyMessage xmlns="urn:schemas-cybersource-com:transaction-data-1.121">
  <c:merchantReferenceCode>MS299131501003</c:merchantReferenceCode>
  <c:requestID>1019827520348290570293</c:requestID>
  <c:purchaseTotals>
    <c:currency>USD</c:currency>
  </c:purchaseTotals>
  <c:ccAuthReversalReply>
    <c:amount>99.49</c:amount>
    <c:authorizationCode>1</c:authorizationCode>
  </c:ccAuthReversalReply>
</c:replyMessage>
```

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

Related concepts
- Required Fields for Capturing a Payment on page 33

Related tasks
- Capturing a Payment on page 33

Required Fields for Capturing a Payment

The following fields are required when creating a capture request:

ccCaptureService_authRequestID	Set to the request ID that was included in the authorization reply message. Optional when **ccAuthService** and **ccCaptureService** are in the same request.
ccCaptureService_run	Set to `true`.
merchantID	
merchantReferenceCode	
paymentSolution	Set to `008`.
purchaseTotals_currency	
purchaseTotals_grandTotalAmount	Either **purchaseTotals_grandTotalAmount** or **item_#_unitPrice** must be included in the request.

Related Information

Simple Order API Field Reference Guide

Capturing a Payment

1. Request the service. Set the **ics_applications** field to `ics_bill`, and send the request to one of these endpoints:
   - `https://ics2ws.ic3.com/commerce/1.x/transactionProcessor`
   - `https://ics2wsa.ic3.com/commerce/1.x/transactionProcessor`
2. Check the response message to make sure that the request was successful. A value of `ACCEPT` for the **decision** field indicates success. For information about reason codes, see *Reason Codes for the Simple Order API*.

XML Example: Basic Credit Card Capture Using the Simple Order API

Capture Request

```
<requestMessage xmlns="urn:schemas-cybersource-com:transaction-data-1.37">
  <merchantID>Napa Valley Vacations</merchantID>
  <merchantReferenceCode>482046C3A7E94F5BD1FE3C66C</merchantReferenceCode>
```

```
  <item id="0">
    <unitPrice>49.95</unitPrice>
    <quantity>1</quantity>
  </item>
  <purchaseTotals>
    <currency>USD</currency>
  </purchaseTotals>
  <ccCaptureService run="true">
    <authRequestID>0305782650000167905080</authRequestID>
  </ccCaptureService>
  <paymentSolution>008</paymentSolution>
</requestMessage>
```

Capture Response

```
<c:replyMessage xmlns:c="urn:schemas-cybersource-com:transaction-data-1.37">
  <c:merchantReferenceCode>482046C3A7E94F5BD1FE3C66C</c:merchantReferenceCode>
  <c:requestID>5019827520348290570293</c:requestID>
  <c:decision>ACCEPT</c:decision>
  <c:reasonCode>100</c:reasonCode>
  <c:purchaseTotals>
    <c:currency>USD</c:currency>
  </c:purchaseTotals>
  <c:ccCaptureReply>
    <c:reasonCode>100</c:reasonCode>
    <c:amount>49.95</c:amount>
    <c:reconciliationID>1094820975023470</c:reconciliationID>
  </c:ccCaptureReply>
</c:replyMessage>
```

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:

ccCaptureService_run	Set this field to `true`.
Fields required for requesting the authorization service	Use the same values that are set for requesting the *Authorization Service* on page 17.

Related Information

Simple Order 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. Request the service. Set the **ics_applications** field to `ics_auth,ics_bill`, and send the request to one of these internet endpoints:
   - `https://ics2ws.ic3.com/commerce/1.x/transactionProcessor`
   - `https://ics2wsa.ic3.com/commerce/1.x/transactionProcessor`
2. Check the response message to make sure that the request was successful. A value of `ACCEPT` for the **decision** field indicates success. For information about reason codes, see *Reason Codes for the Simple Order API*.

XML Example: Basic Credit Card Sale Using the Simple Order API

Authorization and Capture (Sale) Request

```
<requestMessage xmlns="urn:schemas-cybersource-com:transaction-data-1.121">
  <merchantID>demomerchant</merchantID>
  <merchantReferenceCode>demorefnum</merchantReferenceCode>
  <billTo>
    <firstName>James</firstName>
    <lastName>Smith</lastName>
    <street1>1295 Charleston Road</street1>
    <city>Test City</city>
    <state>CA</state>
    <postalCode>99999</postalCode>
    <country>US</country>
    <email>demo@example.com</email>
  </billTo>
  <purchaseTotals>
    <currency>USD</currency>
    <grandTotalAmount>5.00</grandTotalAmount>
  </purchaseTotals>
  <ccCaptureService run="true">
    <commerceIndicator>aesk</commerceIndicator>
  </ccCaptureService>
  <encryptedPayment>
    <data>ABCDEFabcdefABCDEFabcdef0987654321234567</data>
    <descriptor>RklEPUNPTU1PTi5TQU1TVU5HLklOQVBQLlBBWU1FTlQ=</descriptor>
  </encryptedPayment>
  <paymentSolution>008</paymentSolution>
  <paymentNetworkToken>
    <transactionType>1</transactionType>
  </paymentNetworkToken>
</requestMessage>
```

Authorization and Capture (Sale) Response

```
<c:replyMessage>
  <c:merchantReferenceCode>demorefnum</c:merchantReferenceCode>
  <c:requestID>4465840340765000001541</c:requestID>
  <c:decision>ACCEPT</c:decision>
```

```
  <c:reasonCode>100</c:reasonCode>
  <c:purchaseTotals>
    <c:currency>USD</c:currency>
  </c:purchaseTotals>
  <c:ccAuthReply>
    <c:reasonCode>100</c:reasonCode>
    <c:amount>5.00</c:amount>
    <c:authorizationCode>888888</c:authorizationCode>
    <c:avsCode>V</c:avsCode>
    <c:avsCodeRaw>I1</c:avsCodeRaw>
    <c:authorizedDateTime>2015-11-03T20:53:54Z</c:authorizedDateTime>
    <c:processorResponse>100</c:processorResponse>
    <c:reconciliationID>11267051CGJSMQDC</c:reconciliationID>
  </c:ccAuthReply>
  <c:ccCaptureReply>
    <c:reconciliationID>02850840187309570</c:reconciliationID>
    <c:amount>100.00</c:amount>
  </c:ccCaptureReply>
  <c:token>
    <c:prefix>593056</c:prefix>
    <c:suffix>0842</c:suffix>
    <c:expirationMonth>08</c:expirationMonth>
    <c:expirationYear>2021</c:expirationYear>
  </c:token>
</c:replyMessage>
```