Samsung Pay

Using the Simple Order API

May 2019

CyberSource®

the power of payment

CyberSource Corporation HQ | P.O. Box 8999 | San Francisco, CA 94128-8999 | Phone: 800-530-9095

CyberSource Contact Information

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

Copyright

Restricted Rights Legends

Trademarks

Contents

Recent Revisions to This Document **5**

About This Guide **7**

Audience and Purpose **7**

Conventions **7**

  Notes and Important Statements **7**

  Text and Command Conventions **8**

Related Documents **8**

Customer Support **8**

Chapter 1 **Introduction** **9**

Requirements **9**

Supported Processors, Card Types, and Optional Features **10**

Transaction Endpoints **12**

Chapter 2 **Registration** **13**

Registering with Samsung **13**

Registering with CyberSource **14**

Chapter 3 **Integrating the Samsung SDK** **16**

Creating a Project **16**

Integrating the Samsung Pay SDK **17**

Using the API Key **17**

Chapter 4 **Using the Samsung Pay SDK** 18
Eligibility 18
Payment Request 19
  Initiating a Payment 19
  Requesting a Payment 21

Chapter 5 **Authorizing a Payment** 23
Merchant Decryption 23
  Visa Transaction 23
  Mastercard Transaction 25
  American Express Transaction 27
  JCB Transaction 29
CyberSource Decryption 32
  Visa Transaction 32
  Mastercard Transaction 34
  American Express Transaction 37
  JCB Transaction 40
Additional CyberSource Services 43

Appendix A **API Fields** 44
Data Type Definitions 44
Numbered Elements 44
Relaxed Requirements for Address Data 45
API Request Fields 46
API Reply Fields 54

Recent Revisions to This Document

Release	Changes
May 2019	Removed the following request fields that were erroneously added in the April 2019 release: ■ ccSaleService_directoryServerTransactionID ■ ccSaleService_networkTokenCryptogram ■ ccSaleService_paSpecificationVersion Removed the following reply fields that were erroneously added in the April 2019 release: ■ payerAuthEnrollReply_directoryServerTransactionID ■ payerAuthValidateReply_directoryServerTransactionID
April 2019	Added support for tokenized transactions using a network token with 3D Secure or SecureCode. See "Merchant Decryption," page 23. Added the following request fields that support tokenized transactions using a network token with 3D Secure or SecureCode (see "API Request Fields," page 46): ■ ccAuthService_directoryServerTransactionID ■ ccAuthService_networkTokenCryptogram ■ ccAuthService_paSpecificationVersion ■ ccSaleService_directoryServerTransactionID ■ ccSaleService_networkTokenCryptogram ■ ccSaleService_paSpecificationVersion Added the following reply fields that support tokenized transactions using a network token with 3D Secure or SecureCode (see "API Reply Fields," page 54): ■ payerAuthEnrollReply_directoryServerTransactionID ■ payerAuthValidateReply_directoryServerTransactionID Added support for the processor *Elavon Americas*. See "Supported Processors, Card Types, and Optional Features," page 10. Added support for the following optional features by Elavon Americas (see "Supported Processors, Card Types, and Optional Features," page 10): ■ Merchant-Initiated transactions ■ Multiple partial captures ■ Recurring payments

Release	Changes
February 2019	Updated URLs for Samsung Pay Partner Portal. See "Registering with Samsung," page 13.
January 2019	Updated "JCB Transaction," page 29 and "JCB Transaction," page 29 to correct an erroneous Product Service code, Authorization Service fieldname, and payment descriptor. Updated "Supported Processors, Card Types, and Optional Features," page 10 to remove erroneous content regarding Vantiv. Updated URLs for the following: ▪ Samsung Pay Partner Portal (see "Related Documents," page 8) ▪ Transaction Endpoints (see "Transaction Endpoints," page 12) ▪ Samsung Pay registration (see "Registering with Samsung," page 13) ▪ Decrypting payment credentials (see "Encrypted Payment Credential," page 22)
July 2018	All processors: updated information about optional features. See "Supported Processors, Card Types, and Optional Features," page 10.
December 2017	Added the following to the list of supported processors. See "Supported Processors, Card Types, and Optional Features," page 10: ▪ JCN Gateway ▪ Barclays ▪ GPN ▪ OmniPay Direct ▪ Streamline

About This 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 of integrating the Samsung Pay SDK and describes how to request the CyberSource API to process an authorization.

This document describes the Samsung Pay SDK and the CyberSource API. See "Using the Samsung Pay SDK," page 18, and "Authorizing a Payment," page 23. 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

Notes and Important Statements

Note

A *Note* contains helpful suggestions or references to material not contained in the document.

Important

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

Text and Command Conventions

Convention	Usage
bold	▪ Field and service names in text; for example: Include the **card_accountNumber** field. ▪ Items that you are instructed to act upon; for example: Click **Save**.
`Screen text`	Code examples and samples.

Related Documents

CyberSource Documents:

- *Getting Started with CyberSource Advanced for the Simple Order API* (PDF | HTML)
- Simple Order API and SOAP Toolkit API Documentation and Downloads page
- *Credit Card Services Using the Simple Order API* (PDF | HTML)
- *Payment Network Tokenization Using the Simple Order API* (PDF | HTML)

Samsung Pay documents:

Samsung Pay Partner Portal

Refer to the Support Center for complete CyberSource technical documentation:

http://www.cybersource.com/support_center/support_documentation

Customer Support

For support information about any CyberSource service, visit the Support Center:

http://www.cybersource.com/support

CHAPTER 1

Introduction

Requirements

Important

ProductName relies on payment network tokenization. You can sign up for ProductName only if both of the following statements are true:

- Your processor supports payment network tokenization.
- CyberSource supports payment network tokenization 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 ProductName:

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

You must create:

- A CyberSource account. If you do not already have a CyberSource account, contact your local CyberSource sales representative:

  http://www.cybersource.com/locations/

- A merchant account with a supported processor. See "Supported Processors, Card Types, and Optional Features," page 10.

- A profile on the Samsung Pay Partner Portal, and you must obtain a Partner ID. See "Registration," page 13.

Note

All optional features are described in *Payment Network Tokenization Using the Simple Order API.*

Supported Processors, Card Types, and Optional Features

Note All optional features, except split shipments, are described in *Payment Network Tokenization Using the Simple Order API* (PDF | HTML) Split shipments are described in *Credit Card Services Using the Simple Order API* (PDF | HTML).

Table 1 Supported Processors, Card Types, and Optional Features

Processors	Card Types	Optional
American Express Direct	American Express	▪ Multiple partial captures ▪ Recurring Payments
Barclays	Visa, Mastercard, JCB, Maestro (International), Maestro (UK Domestic) If you support Maestro (UK Domestic), you must also support Maestro (International), and you must support Mastercard SecureCode for both card types.	▪ Multiple partial captures ▪ Recurring Payments
Chase Paymentech Solutions	Visa, Mastercard, American Express, Discover, Diners Club, JCB, Carte Blanche, Maestro (International)	▪ Multiple partial captures ▪ Recurring Payments
CyberSource through VisaNet. The supported acquirer is: ▪ Vantiv CyberSource through VisaNet is a single processor with multiple acquirers.	Visa, Mastercard, American Express, Discover, JCB, Diners Club	▪ Split shipments. ▪ Recurring Payments.
Elavon Americas	Visa, Mastercard, American Express, JCB, Diners Club, Discover, China UnionPay	▪ Merchant-Initiated transactions ▪ Multiple partial captures ▪ Recurring payments
FDC Compass	Visa, Mastercard, American Express, Discover, Diners Club, JCB	▪ Multiple partial captures ▪ Recurring Payments

Table 1 Supported Processors, Card Types, and Optional Features (Continued)

Processors	Card Types	Optional
FDC Nashville Global	Visa, Mastercard, American Express, Discover, Diners Club, JCB, China UnionPay	▪ Multiple partial captures. ▪ Recurring Payments
GPN	Visa, Mastercard, American Express, Discover, Diners Club, JCB	▪ Split shipments ▪ Recurring Payments
JCN Gateway	Visa, Mastercard, American Express, Diners Club, JCB, NICOS house card, ORICO house card	Multiple partial captures
OmniPay Direct ▪ Bank of America Merchant Services ▪ First Data Merchant Solutions (Europe) ▪ Global Payments International Acquiring	Visa, Mastercard, Discover, Diners Club, Maestro (UK Domestic), Maestro (International)	▪ Multiple partial captures ▪ Recurring Payments
Streamline	Visa, Mastercard	▪ Multiple partial captures ▪ Recurring Payments ▪ Subsequent authorizations
TSYS Acquiring Solutions	Visa, Mastercard, American Express	▪ Multiple partial captures ▪ Recurring Payments

Transaction Endpoints

CAS (test transactions):

- Akamai endpoints:

  https://ics2wstesta.ic3.com/commerce/1.x/transactionProcessor

- Non-Akamai endpoints:

  https://ics2wstest.ic3.com/commerce/1.x/transactionProcessor

Production (live transactions):

- Akamai endpoints:

  https://ics2wsa.ic3.com/commerce/1.x/transactionProcessor

- Non-Akamai endpoints

  https://ics2ws.ic3.com/commerce/1.x/transactionProcessor

CHAPTER 2

Registration

Registering with Samsung

To register with Samsung:

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

Note Samsung will contact you if clarifications are required.

Step 2 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 the Certificate Signing Request (CSR) file in using the CyberSource Business Center. See "Registering with CyberSource," page 14. Samsung requires the CSR file in order to encrypt sensitive payment data; it contains an identifier and public key.

Step 3 Using the Samsung Pay Partner Portal, upload the CSR file.

Step 4 Enter an application name and a package name.

Step 5 When you associate the CSR file with the application, Samsung generates a product ID.

Step 6 Create login details for application developers on the Samsung Pay Partner Portal.

Step 7 Download and integrate the Samsung Pay SDK into your application. See "Using the Samsung Pay SDK," page 18.

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

Step 8 Register a Samsung account ID and request a *debug-api-key* file using the Samsung Pay Partner Portal. The *debug-api-key* file is valid for three months. See "Using the API Key," page 17.

Note 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 purposes.

Step 9 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.

Registering with CyberSource

To register with CyberSource:

Step 1 Log in to the Business Center:

- Create a CSR file for live transactions: https://ebc.cybersource.com
- Create a CSR file for test transactions: https://ebctest.cybersource.com

Step 2 Under Account Management in the left navigation panel, click Digital Payment Solutions.

Step 3 Click **Sign Up**. Follow the steps to verify your account information and accept the ProductName Merchant Services Agreement.

Step 4 Register with CyberSource:

a. Enter your Samsung partner ID that you obtained in Step 2.

b. Click **Generate CSR** to generate a Certificate Signing Request (CSR) file that is associated with your Samsung partner ID.

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.

c. Submit the CSR file to Samsung.

CHAPTER 3

Integrating the Samsung SDK

Creating a Project

To create a new project using Android Studio:

Step 1 Download Android Studio.

Step 2 Open Android Studio and click **Start a new Android Studio project**.

Step 3 In the New Project settings, enter the following:

- The name of your application.
- The company domain.
- 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.

Step 4 Click **Next**.

Step 5 In the Target Android Devices settings, choose the required API levels.

Step 6 Click **Next**.

Step 7 Choose the required activity and click **Finish**.

Integrating the Samsung Pay SDK

To integrate the Samsung Pay SDK:

Step 1 Add the *samsungpay.jar* and *sdk-v1.0.0.jar* files to the *libs* folder of your Android project.

Step 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')
}
```

Step 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 (Step 8) and include it in the *manifest* file.

To use the API key:

Step 1 Include the API key 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 meta-data.

Example 1 Debug Mode

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

Example 2 Release Mode

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

CHAPTER

Using the Samsung Pay SDK

4

Eligibility

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 a `SsdkUnsupportedException` or `NullPointerException` error.

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

Example 3 Samsung Pay Class

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

Payment Request

Initiating a Payment

To initiate a payment:

Step 1 Include the following 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. This field is required.
- Amount—this field is required.
- Payment Protocol—3D Secure. This field is required.
- Permitted Card Brands—specify the card brands that are supported such as Visa, Mastercard, or American Express. This field is required.
- 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 4 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

To request a payment:

Step 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.

Step 2 Request the `startSamsungPay()` API method and include the following data:

- `PaymentInfo`—the paymentInfo structure, which contains payment information.
- `PID`—the product ID created in the Samsung Pay Partner Portal. See "Registration," page 13.
- `StatusListener`—the result of the payment request is delivered to `StatusListener`. This listener should be registered before calling the `startSamsungPay()` API method.

When you request the `startSamsungPay()` API method, the Samsung Pay online payment sheet is displayed on the screen of 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. See Table 2, "Encrypted Payment Credential," on page 22.
- `onFailure()`—this event is requested when the transaction fails. It returns an error code and error message.

Example 5 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 4, "Transaction Request Structure,"
    on page 20.
    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 ");
    };
```

Table 2 Encrypted Payment Credential

Payment Credential	Description
method	Payment protocol: 3D 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.

For information on how to decrypt the encrypted payment credential, see:

https://pay.samsung.com/developers

Authorizing a Payment

CHAPTER 5

Note

Your payment processor can include API reply fields that are not documented in this guide. See *Credit Card Services Using the Simple Order API* (PDF | HTML) for detailed descriptions of additional API reply fields.

Merchant Decryption

Visa Transaction

To request an authorization for a Visa transaction:

Note

See "API Request Fields," page 46, and "API Reply Fields," page 54, for detailed field descriptions.

Step 1 Set the **card_accountNumber** field to the payment network token value.

Step 2 Set the **card_expirationMonth** and **card_expirationYear** values to the payment network token expiration date fields.

Step 3 Set the **ccAuthService_cavv** field to the 3D Secure cryptogram of the payment network token.

Step 4 Set the **ccAuthService_networkTokenCryptogram** field to the network token cryptogram.

Step 5 Set the **paymentNetworkToken_transactionType** field to `1`.

Step 6 Set the **ccAuthService_commerceIndicator** field to `internet`.

Step 7 Set the **paymentSolution** field to `008`.

Example 6 Merchant Decryption Authorization Request (Visa)

```
<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>
      <xid>1234567890987654321ABCDEFabcdefABCDEF123</xid>
   </ccAuthService>
   <paymentNetworkToken>
      <transactionType>1</transactionType>
   </paymentNetworkToken>
   <paymentSolution>008</paymentSolution>
</requestMessage>
```

Example 7 Merchant Decryption Authorization Reply (Visa)

```
<c:replyMessage>
   <c:merchantReferenceCode>demorefnum</c:merchantReferenceCode>
   <c:requestID>4465840340765000001541</c:requestID>
   <c:decision>ACCEPT</c:decision>
   <c:reasonCode>100</c:reasonCode>
   <c:requestToken>Ahj/7wSR5C/4Icd2fdAKakGLadfg5535r/ghx3Z90AoBj3u</c:requestToken>
   <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>
```

Mastercard Transaction

To request an authorization for a Mastercard transaction:

Note See "API Request Fields," page 46, and "API Reply Fields," page 54, for detailed field descriptions.

Step 1 Set the **card_accountNumber** field to the payment network token value.

Step 2 Set the **card_expirationMonth** and **card_expirationYear** values to the payment network token expiration date fields.

Step 3 Set the **ucaf_authenticationData** field to the 3D Secure cryptogram of the payment network token.

Step 4 Set the **ccAuthService_networkTokenCryptogram** field to the network token cryptogram.

Step 5 Set the **ucaf_collectionIndicator** field to 2.

Step 6 Set the **paymentNetworkToken_transactionType** field to 1.

Step 7 Set the **ccAuthService_commerceIndicator** field to `spa`.

Step 8 Set the **paymentSolution** field to `008`.

Example 8 Merchant Decryption Authorization Request (Mastercard)

```
<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>
```

Example 9 Merchant Decryption Authorization Reply (Mastercard)

```
<c:replyMessage>
    <c:merchantReferenceCode>demorefnum</c:merchantReferenceCode>
    <c:requestID>4465840340765000001541</c:requestID>
    <c:decision>ACCEPT</c:decision>
    <c:reasonCode>100</c:reasonCode>
    <c:requestToken>Ahj/7wSR5C/4Icd2fdAKakGLadfg5535r/ghx3Z90AoBj3u</c:requestToken>
    <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>
```

American Express Transaction

To request an authorization for an American Express transaction:

Note See "API Request Fields," page 46, and "API Reply Fields," page 54, for detailed field descriptions.

Step 1 Set the **card_accountNumber** field to the payment network token value.

Step 2 Set the **card_expirationMonth** and **card_expirationYear** values to the payment network token expiration date fields.

Step 3 Set the **ccAuthService_cavv** field to the 3D Secure cryptogram of the payment network token.

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

Step 4 Set the **ccAuthService_networkTokenCryptogram** field to the network token cryptogram.

Step 5 Set the **paymentNetworkToken_transactionType** field to `1`.

Step 6 Set the **ccAuthService_commerceIndicator** field to `aesk`.

Step 7 Set the **paymentSolution** field to `008`.

Example 10 Merchant Decryption Authorization Request (American Express)

```
<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>xxxx8224631xxxx</accountNumber>
      <expirationMonth>12</expirationMonth>
      <expirationYear>2020</expirationYear>
   </card>
   <ccAuthService run="true">
      <cavv>ABCDEFabcdefABCDEFabcdef0987654321234567</cavv>
      <commerceIndicator>aesk</commerceIndicator>
      <xid>1234567890987654321ABCDEFabcdefABCDEF123</xid>
   </ccAuthService>
   <paymentNetworkToken>
      <transactionType>1</transactionType>
   </paymentNetworkToken>
   <paymentSolution>008</paymentSolution>
</requestMessage>
```

Example 11 Merchant Decryption Authorization Reply (American Express)

```
<c:replyMessage>
   <c:merchantReferenceCode>demorefnum</c:merchantReferenceCode>
   <c:requestID>4465840340765000001541</c:requestID>
   <c:decision>ACCEPT</c:decision>
   <c:reasonCode>100</c:reasonCode>
   <c:requestToken>Ahj/7wSR5C/4Icd2fdAKakGLadfg5535r/ghx3Z90AoBj3u</c:requestToken>
   <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:replyMessage>
```

JCB Transaction

To request an authorization for a JCB transaction:

Note See "API Request Fields," page 46, and "API Reply Fields," page 54, for detailed field descriptions.

Step 1 Set the **card_accountNumber** field to the payment network token value.

Step 2 Set the **card_expirationMonth** and **card_expirationYear** fields to the payment network token expiration date values.

Step 3 Set the **ccAuthService_cavv** field to the 3D Secure cryptogram of the payment network token.

Step 4 Set the **ccAuthService_networkTokenCryptogram** field to the network token cryptogram.

Step 5 Set the **paymentNetworkToken_transactionType** field to `1`.

Step 6 Set the **ccAuthService_eciRaw** field to the ECI value contained in the Samsung Pay response payload.

Step 7 Set the **PaymentSolution** field to `008`.

Example 12 Merchant Decryption Authorization Request (JCB)

```
<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>001</cardType>
   </card>
   <ccAuthService run="true">
      <cavv>ABCDEFabcdefABCDEFabcdef0987654321234567</cavv>
      <eciRaw>5</eciRaw>
   </ccAuthService>
   <paymentNetworkToken>
      <transactionType>1</transactionType>
   </paymentNetworkToken>
   <paymentSolution>008</paymentSolution>
</requestMessage>
```

Example 13 Merchant Decryption Authorization Reply (JCB)

```
<c:replyMessage>
   <c:merchantReferenceCode>demorefnum</c:merchantReferenceCode>
   <c:requestID>4465840340765000001541</c:requestID>
   <c:decision>ACCEPT</c:decision>
   <c:reasonCode>100</c:reasonCode>
   <c:requestToken>
      Ahj/7wSR5C/4Icd2fdAKakGLadfg5535r/ghx3Z90AoBj3u
   </c:requestToken>
   <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>
```

Example 14 Merchant Decryption NVP Request (JCB)

```
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=001
ccAuthService_cavv=ABCDEFabcdefABCDEFabcdef0987654321234567
ccAuthService_eciRaw=5
paymentNetworkToken_transactionType=1
paymentSolution=008
```

Example 15 Merchant Decryption NVP Reply (JCB)

```
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
```

CyberSource Decryption

Visa Transaction

To request an authorization for a Visa transaction:

Note: See "API Request Fields," page 46, and "API Reply Fields," page 54, for detailed field descriptions.

Step 1 Set the **encryptedPayment_data** field to the value that was returned from Samsung Pay in the *3ds.data* block.

a Retrieve the payment data from Samsung Pay in JSON Web Encryption (JWE) format, which includes the key reference ID (KID) value of your public key hash in the JWE header.

b Encode it in Base64.

c Set the values in this structure:

```
{
    "publicKeyHash": "enter the encrypted KID value here",
    "version": "100",
    "data": "enter the encoded data from step b here"
}
```

   d. Encode the structure in Base64.

   e. Add the value to the **encryptedPayment_data** field.

Step 2 Set the **encryptedPayment_descriptor** field to `RklEPUNPTU1PTi5TQU1TVU5HLklOQVBQLlBBWU1FTlQ=`.

Step 3 Set the **paymentNetworkToken_transactionType** to `1`.

Step 4 Set the **ccAuthService_commerceIndicator** field to `internet`.

Step 5 Set the **paymentSolution** field to `008`.

Example 16 CyberSource Decryption Authorization Request (Visa)

```
<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>
```

Example 17 CyberSource Decryption Authorization Reply (Visa)

```
<c:replyMessage>
    <c:merchantReferenceCode>demorefnum</c:merchantReferenceCode>
    <c:requestID>4465840340765000001541</c:requestID>
    <c:decision>ACCEPT</c:decision>
    <c:reasonCode>100</c:reasonCode>
    <c:requestToken>Ahj/7wSR5C/4Icd2fdAKakGLadfg5535r/ghx3Z90AoBj3u</c:requestToken>
    <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>
```

Mastercard Transaction

To request an authorization for a Mastercard transaction:

Note See "API Request Fields," page 46, and "API Reply Fields," page 54, for detailed field descriptions.

Step 1 Set the **encryptedPayment_data** field to the value that was returned from Samsung Pay in the *3ds.data* block.

a Retrieve the payment data from Samsung Pay in JSON Web Encryption (JWE) format, which includes the key reference ID (KID) value of your public key hash in the JWE header.

b Encode it in Base64.

c Set the values in this structure:

```
{
   "publicKeyHash": "enter the encrypted KID value here",
   "version": "100",
   "data": "enter the encoded data from step b here"
}
```

d Encode the structure in Base64.

e Add the value to the **encryptedPayment_data** field.

Step 2 Set the **encryptedPayment_descriptor** field to `RklEPUNPTU1PTi5TQU1TVU5HLklOQVBQLlBBWU1FTlQ=`.

Step 3 Set the **ccAuthService_commerceIndicator** field to `spa`.

Step 4 Set the **paymentNetworkToken_transactionType** field to `1`.

Step 5 Set the **paymentSolution** field to `008`.

Example 18 CyberSource Decryption Authorization Request (Mastercard)

```
<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>
```

Example 19 CyberSource Decryption Authorization Reply (Mastercard)

```
<c:replyMessage>
   <c:merchantReferenceCode>demorefnum</c:merchantReferenceCode>
   <c:requestID>4465840340765000001541</c:requestID>
   <c:decision>ACCEPT</c:decision>
   <c:reasonCode>100</c:reasonCode>
   <c:requestToken>Ahj/7wSR5C/4Icd2fdAKakGLadfg5535r/ghx3Z90AoBj3u</c:requestToken>
   <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>
```

American Express Transaction

To request an authorization for an American Express transaction:

Note See "API Request Fields," page 46, and "API Reply Fields," page 54, for detailed field descriptions.

Step 1 Set the **encryptedPayment_data** field to the value that was returned from Samsung Pay in the *3ds.data* block.

a Retrieve the payment data from Samsung Pay in JSON Web Encryption (JWE) format, which includes the key reference ID (KID) value of your public key hash in the JWE header.

b Encode it in Base64.

c Set the values in this structure:

```
{
   "publicKeyHash": "enter the encrypted KID value here",
   "version": "100",
   "data": "enter the encoded data from step b here"
}
```

d Encode the structure in Base64.

e Add the value to the **encryptedPayment_data** field.

Step 2 Set the **encryptedPayment_descriptor** field to `RklEPUNPTU1PTi5TQU1TVU5HLklOQVBQLlBBWU1FTlQ=`.

Step 3 Set the **ccAuthService_commerceIndicator** field to `spa`.

Step 4 Set the **paymentSolution** field to `008`.

Example 20 CyberSource Decryption Authorization Request (American Express)

```
<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>aesk</commerceIndicator>
   </ccAuthService>
   <encryptedPayment>
      <data>ABCDEFabcdefABCDEFabcdef0987654321234567</data>
      <descriptor>RklEPUNPTU1PTi5TQU1TVU5HLklOQVBQLlBBWU1FTlQ=</descriptor>
   </encryptedPayment>
   <paymentSolution>008</paymentSolution>
   <paymentNetworkToken>
      <transactionType>1</transactionType>
   </paymentNetworkToken>
</requestMessage>
```

Example 21 CyberSource Decryption Authorization Reply (American Express)

```
<c:replyMessage>
   <c:merchantReferenceCode>demorefnum</c:merchantReferenceCode>
   <c:requestID>4465840340765000001541</c:requestID>
   <c:decision>ACCEPT</c:decision>
   <c:reasonCode>100</c:reasonCode>
   <c:requestToken>Ahj/7wSR5C/4Icd2fdAKakGLadfg5535r/ghx3Z90AoBj3u</c:requestToken>
   <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: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>
```

JCB Transaction

To request an authorization for a JCB transaction:

Note

See "API Request Fields," page 46, and "API Reply Fields," page 54, for detailed field descriptions.

Step 1 Set the **encryptedPayment_data** field to the Base64 encoded value obtained from the **paymentData** property of the **PKPaymentToken** object.

Step 2 Set the **encryptedPaymentdescriptor** field to `RklEPUNPTU1PTi5TQU1TVU5HLklOQVBQLlBBWU1FTlQ=`.

Step 3 Set the **paymentSolution** field to `008`.

Example 22 CyberSource Decryption Authorization Request (JCB)

```
<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>001</cardType>
   </card>
   <ccAuthService run="true"/>
      <paymentSolution>008</paymentSolution>
</requestMessage>
```

Example 23 CyberSource Decryption Authorization Reply (JCB)

```
<c:replyMessage>
   <c:merchantReferenceCode>demorefnum</c:merchantReferenceCode>
   <c:requestID>4465840340765000001541</c:requestID>
   <c:decision>ACCEPT</c:decision>
   <c:reasonCode>100</c:reasonCode>
   <c:requestToken>Ahj/7wSR5C/4Icd2fdAKakGLadfg5535r/ghx3Z90AoBj3u</
   c:requestToken>
   <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>
```

Additional CyberSource Services

Refer to *Credit Card Services Using the Simple Order API* (PDF | HTML) for information on how to request these follow-on services.

Table 3 CyberSource Services

CyberSource Service	Description
Capture	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.
Sale	A sale is a bundled authorization and capture. Request the authorization and capture services at the same time. CyberSource processes the capture immediately.
Authorization Reversal	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.

API Fields

Data Type Definitions

For more information about these data types, see the World Wide Web Consortium (W3C) XML Schema Part 2: Datatypes Second Edition.

Table 4 Data Type Definitions

Data Type	Description
Integer	Whole number {..., -3, -2, -1, 0, 1, 2, 3, ...}
String	Sequence of letters, numbers, spaces, and special characters

Numbered Elements

The CyberSource XML schema includes several numbered elements. You can include these complex elements more than once in a request. For example, when a customer order includes more than one item, you must include multiple `<item>` elements in your request. Each item is numbered, starting with `0`. The XML schema uses an `id` attribute in the item's opening tag to indicate the number. For example:

```
<item id="0">
```

As a name-value pair field name, this tag is represented as **item_0**. In this portion of the field name, the underscore before the number does not indicate hierarchy in the XML schema. The item fields are generically referred to as **item_#_<element name>** in the documentation.

Below is an example of the numbered `<item>` element and the corresponding name-value pair field names. If you are using SOAP, the client contains a corresponding `Item` class.

Example 24 Numbered XML Schema Element Names and Name-Value Pair Field Names

XML Schema Element Names	Corresponding Name-Value Pair Field Names

`<item id="0">`
`  <unitPrice>`
`  <quantity>`
`</item>`
	item_0_unitPrice
	item_0_quantity

`<item id="1">`
`  <unitPrice>`
`  <quantity>`
`</item>`
	item_1_unitPrice
	item_1_quantity

Important

When a request is in XML format and includes an `<item>` element, the element must include an `id` attribute. For example: `<item id="0">`.

Relaxed Requirements for Address Data

To enable relaxed requirements for address data and expiration date, contact CyberSource Customer Support to have your account configured for this feature. For details about relaxed requirements, see the Relaxed Requirements for Address Data and Expiration Date page.

API Request Fields

Note Unless otherwise noted, all field names are case sensitive and all fields accept special characters such as `@`, `#`, and `%`.

Table 5 API Request Fields

Field	Description	Used By: Required (R) or Optional (O)	Data Type (Length)
billTo_city	City of the billing address. **Important** It is your responsibility to determine whether a field is required for the transaction you are requesting. See "Relaxed Requirements for Address Data," page 45.	ccAuthService (See description)	String (50)
billTo_country	Country of the billing address. Use the two-character *ISO Standard Country Codes*. **Important** It is your responsibility to determine whether a field is required for the transaction you are requesting. See "Relaxed Requirements for Address Data," page 45.	ccAuthService (See description)	String (2)
billTo_email	Customer's email address. **Important** It is your responsibility to determine whether a field is required for the transaction you are requesting. See "Relaxed Requirements for Address Data," page 45.	ccAuthService (See description)	String (255)
billTo_firstName	Customer's first name. For a credit card transaction, this name must match the name on the card. **Important** It is your responsibility to determine whether a field is required for the transaction you are requesting. See "Relaxed Requirements for Address Data," page 45.	ccAuthService (See description)	String (60)
billTo_ipAddress	Customer's IP address.	ccAuthService (O)	String (15)

1 The TC 33 Capture file contains information about the purchases and refunds that a merchant submits to CyberSource. CyberSource through VisaNet creates the TC 33 Capture file at the end of the day and sends it to the merchant's acquirer, who uses this information to facilitate end-of-day clearing processing with payment card companies.

Table 5 API Request Fields (Continued)

Field	Description	Used By: Required (R) or Optional (O)	Data Type (Length)
billTo_lastName	Customer's last name. For a credit card transaction, this name must match the name on the card. **Important** It is your responsibility to determine whether a field is required for the transaction you are requesting. See "Relaxed Requirements for Address Data," page 45.	ccAuthService (See description)	String (60)
billTo_phoneNumber	Customer's phone number. CyberSource recommends that you include the country code when the order is from outside the U.S.	ccAuthService (O)	String (15)
billTo_postalCode	Postal code for the billing address. The postal code must consist of 5 to 9 digits. When the billing country is the U.S., the 9-digit postal code must follow this format: [5 digits][dash][4 digits] **Example** 12345-6789 When the billing country is Canada, the 6-digit postal code must follow this format: [alpha][numeric][alpha][space] [numeric][alpha][numeric] **Example** A1B 2C3 **Important** It is your responsibility to determine whether a field is required for the transaction you are requesting. See "Relaxed Requirements for Address Data," page 45.	ccAuthService (See description)	String (9)
billTo_state	State or province of the billing address. For an address in the U.S. or Canada, use the *State, Province, and Territory Codes for the United States and Canada*. **Important** It is your responsibility to determine whether a field is required for the transaction you are requesting. See "Relaxed Requirements for Address Data," page 45.	ccAuthService (See description)	String (2)

1 The TC 33 Capture file contains information about the purchases and refunds that a merchant submits to CyberSource. CyberSource through VisaNet creates the TC 33 Capture file at the end of the day and sends it to the merchant's acquirer, who uses this information to facilitate end-of-day clearing processing with payment card companies.

Table 5 API Request Fields (Continued)

Field	Description	Used By: Required (R) or Optional (O)	Data Type (Length)
billTo_street1	First line of the billing street address. **Important** It is your responsibility to determine whether a field is required for the transaction you are requesting. See "Relaxed Requirements for Address Data," page 45.	ccAuthService (See description)	String (60)
billTo_street2	Additional address information. **Example** Attention: Accounts Payable	ccAuthService (R)	String (60)
card_accountNumber	Payment network token value. This value is obtained by decrypting the customer's encrypted payment data.	ccAuthService (R)	Nonnegative integer (20)
card_expirationMonth	Two-digit month in which the payment network token expires. Format: MM. Possible values: `01` through `12`.	ccAuthService (R)	String (2)
card_expirationYear	Four-digit year in which the payment network token expires. Format: YYYY.	ccAuthService (R)	Nonnegative integer (4)
ccAuthService_cavv	***Visa*** Cryptogram for payment network tokenization transactions. The value for this field must be 28-character Base64 or 40-character hex binary. All cryptograms use one of these formats. ***American Express*** For a 20-byte cryptogram, set this field to the cryptogram for payment network tokenization transactions. For a 40-byte cryptogram, set this field to block A of the cryptogram for payment network tokenization transactions. The value for this field must be 28-character Base64 or 40-character hex binary. All cryptograms use one of these formats.	ccAuthService (R)	String (40)

1 The TC 33 Capture file contains information about the purchases and refunds that a merchant submits to CyberSource. CyberSource through VisaNet creates the TC 33 Capture file at the end of the day and sends it to the merchant's acquirer, who uses this information to facilitate end-of-day clearing processing with payment card companies.

Table 5 API Request Fields (Continued)

Field	Description	Used By: Required (R) or Optional (O)	Data Type (Length)
ccAuthService_ commerceIndicator	For a payment network tokenization transaction. Possible values: ▪ `aesk` for the American Express card type ▪ `spa` for the Mastercard card type ▪ `internet` for the Visa card type	ccAuthService (R)	String (20)
ccAuthService_ directoryServerTrans actionID	Identifier generated during the authentication transaction by the Mastercard Directory Server and passed back with the authentication results.	ccAuthService (O)	String (36)
ccAuthService_eciRaw	Raw electronic commerce indicator (ECI).	ccAuthService	String (2)
ccAuthService_ networkTokenCryptogram	Token authentication verification value cryptogram. For token-based transactions with 3D Secure or SecureCode, you must submit both types of cryptograms: network token and 3D Secure/SecureCode. The value for this field must be 28-character Base64 or 40-character hex binary. All cryptograms use one of these formats.	ccAuthService (O)	String (40)
ccAuthService_ paSpecificationVersion	The 3D Secure version that you used for Secured Consumer Authentication (SCA); for example, 3D Secure 1.0.2 or 2.0.0.	ccAuthService (O)	String (20)
ccAuthService_run	Whether to include **ccAuthService** in your request. Possible values: ▪ `true`: Include the service in your request. ▪ `false` (default): Do not include the service in your request.	ccAuthService (R)	String (5)

1 The TC 33 Capture file contains information about the purchases and refunds that a merchant submits to CyberSource. CyberSource through VisaNet creates the TC 33 Capture file at the end of the day and sends it to the merchant's acquirer, who uses this information to facilitate end-of-day clearing processing with payment card companies.

Table 5 API Request Fields (Continued)

Field	Description	Used By: Required (R) or Optional (O)	Data Type (Length)
ccAuthService_xid	***Visa*** Cryptogram for payment network tokenization transactions. The value for this field must be 28-character Base64 or 40-character hex binary. All cryptograms use one of these formats. ***American Express*** For a 20-byte cryptogram, set this field to the cryptogram for payment network tokenization transactions. For a 40-byte cryptogram, set this field to block A of the cryptogram for payment network tokenization transactions (see "Merchant Decryption," page 23). The value for this field must be 28-character Base64 or 40-character hex binary. All cryptograms use one of these formats.	ccAuthService (R)	String (40)
encryptedPayment_data	Encrypted payment data value. If you are using the CyberSource Decryption option, populate this field with the encrypted payment data value returned from Samsung Pay in the *3ds.data* block.	ccAuthService (R)	
encryptedPayment_ descriptor	Format of the encrypted payment data. The value for Samsung Pay is `RklEPUNPTU1PTi5TQU1TVU5HLklOQVBQLlBBWU1FTlQ=`	ccAuthService (R)	
item_#_productCode	Type of product. This value is used to determine the product category: electronic, handling, physical, service, or shipping. The default is `default`. See "Numbered Elements," page 44.	ccAuthService (O)	String (255)
item_#_productName	Name of the product. This field is required when the **item_#_ productCode** value is not `default` or one of the values related to shipping and/or handling. See "Numbered Elements," page 44.	ccAuthService (See description)	String (255)

1 The TC 33 Capture file contains information about the purchases and refunds that a merchant submits to CyberSource. CyberSource through VisaNet creates the TC 33 Capture file at the end of the day and sends it to the merchant's acquirer, who uses this information to facilitate end-of-day clearing processing with payment card companies.

Table 5 API Request Fields (Continued)

Field	Description	Used By: Required (R) or Optional (O)	Data Type (Length)
item_#_productSKU	Identification code for the product. This field is required when the **item_#_productCode** value is not `default` or one of the values related to shipping and/or handling. See "Numbered Elements," page 44.	ccAuthService (See description)	String (255)
item_#_quantity	Default is 1. This field is required when the **item_#_productCode** value is not `default` or one of the values related to shipping and/or handling. See "Numbered Elements," page 44.	ccAuthService (See description)	Integer (10)
item_#_taxAmount	Total tax to apply to the product. This value cannot be negative. See "Numbered Elements," page 44.	ccAuthService (See description)	String (15)
item_#_unitPrice	Per-item price of the product. This value cannot be negative. You can include a decimal point (.), but you cannot include any other special characters. See "Numbered Elements," page 44.	ccAuthService (See description)	String (15)
merchantID	Your CyberSource merchant ID. Use the same merchant ID for evaluation, testing, and production.	ccAuthService (R)	String (30)
merchantReferenceCode	Merchant-generated order reference or tracking number. CyberSource recommends that you send a unique value for each transaction so that you can perform meaningful searches for the transaction. For information about tracking orders, see *Getting Started with CyberSource Advanced for the Simple Order API* (PDF | HTML).	ccAuthService (R)	String (50)
paymentNetworkToken_assuranceLevel	Confidence level of the tokenization. This value is assigned by the token service provider. **Note** This field is supported only for FDC Nashville Global.	ccAuthService (O)	String (2)

1 The TC 33 Capture file contains information about the purchases and refunds that a merchant submits to CyberSource. CyberSource through VisaNet creates the TC 33 Capture file at the end of the day and sends it to the merchant's acquirer, who uses this information to facilitate end-of-day clearing processing with payment card companies.

Table 5 API Request Fields (Continued)

Field	Description	Used By: Required (R) or Optional (O)	Data Type (Length)
paymentNetworkToken_ deviceTechType	Type of technology used in the device to store token data. Possible value: `002`: Host card emulation (HCE) Emulation of a smart card by using software to create a virtual and exact representation of the card. Sensitive data is stored in a database that is hosted in the cloud. For storing payment credentials, a database must meet very stringent security requirements that exceed PCI DSS. **Note** This field is supported only for FDC Compass.	ccAuthService (O)	Integer (3)
paymentNetworkToken_ requestorID	Value that identifies your business and indicates that the cardholder's account number is tokenized. This value is assigned by the token service provider and is unique within the token service provider's database. **Note** This field is supported only for FDC Nashville Global and Chase Paymentech Solutions.	ccAuthService (O)	String (11)
paymentNetworkToken_ transactionType	Type of transaction that provided the token data. This value does not specify the token service provider; it specifies the entity that provided you with information about the token. Set the value for this field to `1`.	ccAuthService (R)	String (1)
paymentSolution	Identifies Samsung Pay as the payment solution that is being used for the transaction: Set the value for this field to `008`. **Note** This unique ID differentiates digital solution transactions within the CyberSource platform for reporting purposes.	ccAuthService (R)	String (3)
purchaseTotals_currency	Currency used for the order: USD	ccAuthService (R)	String (5)

1 The TC 33 Capture file contains information about the purchases and refunds that a merchant submits to CyberSource. CyberSource through VisaNet creates the TC 33 Capture file at the end of the day and sends it to the merchant's acquirer, who uses this information to facilitate end-of-day clearing processing with payment card companies.

Table 5 API Request Fields (Continued)

Field	Description	Used By: Required (R) or Optional (O)	Data Type (Length)
purchaseTotals_ grandTotalAmount	Grand total for the order. This value cannot be negative. You can include a decimal point (.), but you cannot include any other special characters. CyberSource truncates the amount to the correct number of decimal places.	ccAuthService (R)	String (15)
surchargeAmount	amount is included in the total transaction amount but is passed in a separate field to the issuer and acquirer for tracking. The issuer can provide information about the surcharge amount to the customer. This field is supported only for CyberSource through VisaNet.	ccAuthService (O)	String (15)
ucaf_authenticationData	Cryptogram for payment network tokenization transactions with Mastercard.	ccAuthService (R)	String (32)
ucaf_collectionIndicator	Required field for payment network tokenization transactions with Mastercard. Set the value for this field to 2.	ccAuthService (R)	String with numbers only (1)

1 The TC 33 Capture file contains information about the purchases and refunds that a merchant submits to CyberSource. CyberSource through VisaNet creates the TC 33 Capture file at the end of the day and sends it to the merchant's acquirer, who uses this information to facilitate end-of-day clearing processing with payment card companies.

API Reply Fields

Important

Because CyberSource can add reply fields and reason codes at any time:

- You must parse the reply data according to the names of the fields instead of the field order in the reply. For more information about parsing reply fields, see the documentation for your client.
- Your error handler should be able to process new reason codes without problems.
- Your error handler should use the **decision** field to determine the result if it receives a reply flag that it does not recognize.

Note

Your payment processor can include additional API reply fields that are not documented in this guide. See *Credit Card Services Using the Simple Order API* (PDF | HTML) for detailed descriptions of additional API reply fields.

Table 6 API Reply Fields

Field	Description	Returned By	Data Type & Length
card_suffix	Last four digits of the cardholder's account number. This field is returned only for tokenized transactions. You can use this value on the receipt that you give to the cardholder. This field is returned only for FDC Nashville Global.	ccAuthReply	String (4)
ccAuthReply_amount	Amount that was authorized.	ccAuthReply	String (15)
ccAuthReply_ authorizationCode	Authorization code. Returned only when the processor returns this value.	ccAuthReply	String (7)
ccAuthReply_ authorizedDateTime	Time of authorization. Format: YYYY-MM-DDThh:mm:ssZ Example: 2016-08-11T22:47:57Z equals August 11, 2016, at 22:47:57 (10:47:57 p.m.).	ccAuthReply	String (20)
ccAuthReply_avsCode	AVS results. See *Credit Card Services Using the Simple Order API* (PDF | HTML) for a detailed list of AVS codes.	ccAuthReply	String (1)
ccAuthReply_ avsCodeRaw	AVS result code sent directly from the processor. Returned only when the processor returns this value.	ccAuthReply	String (10)
ccAuthReply_cvCode	CVN result code. See *Credit Card Services Using the Simple Order API* (PDF | HTML) for a detailed list of CVN codes.	ccAuthReply	String (1)

Table 6 API Reply Fields (Continued)

Field	Description	Returned By	Data Type & Length
ccAuthReply_cvCodeRaw	CVN result code sent directly from the processor. Returned only when the processor returns this value.	ccAuthReply	String (10)
ccAuthReply_ processorResponse	For most processors, this is the error message sent directly from the bank. Returned only when the processor returns this value.	ccAuthReply	String (10)
ccAuthReply_reasonCode	Numeric value corresponding to the result of the credit card authorization request. See *Credit Card Services Using the Simple Order API* (PDF | HTML) for a detailed list of reason codes.	ccAuthReply	Integer (5)
ccAuthReply_ reconciliationID	Reference number for the transaction. This value is not returned for all processors.	ccAuthReply	String (60)
decision	Summarizes the result of the overall request. Possible values: ■ `ACCEPT` ■ `ERROR` ■ `REJECT` ■ `REVIEW`: Returned only when you use CyberSource Decision Manager.	ccAuthReply	String (6)
invalidField_0 through invalidField_N	Fields in the request that contained invalid data. For information about missing or invalid fields, see *Getting Started with CyberSource Advanced for the Simple Order API* (PDF | HTML).	ccAuthReply	String (100)
merchantReferenceCode	Order reference or tracking number that you provided in the request. If you included multi-byte characters in this field in the request, the returned value might include corrupted characters.	ccAuthReply	String (50)

Table 6 API Reply Fields (Continued)

Field	Description	Returned By	Data Type & Length
missingField_0 through missingField_N	Required fields that were missing from the request. For information about missing or invalid fields, see *Getting Started with CyberSource Advanced for the Simple Order API* (PDF | HTML).	ccAuthReply	String (100)
paymentNetworkToken_ assuranceLevel	Confidence level of the tokenization. This value is assigned by the token service provider. **Note** This field is returned only for FDC Nashville Global.	ccAuthReply	String (2)
purchaseTotals_currency	Currency used for the order. For the possible values, see the *ISO Standard Currency Codes*.	ccAuthReply	String (5)
reasonCode	Numeric value corresponding to the result of the overall request. See *Credit Card Services Using the Simple Order API* (PDF | HTML) for a detailed list of reason codes.	ccAuthReply	Integer (5)
requestID	Identifier for the request generated by the client.	ccAuthReply	String (26)
requestToken	Request token data created by CyberSource for each reply. The field is an encoded string that contains no confidential information such as an account or card verification number. The string can contain a maximum of 256 characters.	ccAuthReply	String (256)
token_expirationMonth	Month in which the token expires. CyberSource includes this field in the reply message when it decrypts the payment blob for the tokenized transaction. Format: MM. Possible values: `01` through `12`.	ccAuthReply	String (2)
token_expirationYear	Year in which the token expires. CyberSource includes this field in the reply message when it decrypts the payment blob for the tokenized transaction. Format: YYYY.	ccAuthReply	String (4)

Table 6 API Reply Fields (Continued)

Field	Description	Returned By	Data Type & Length
token_prefix	First six digits of token. CyberSource includes this field in the reply message when it decrypts the payment blob for the tokenized transaction.	ccAuthReply	String (6)
token_suffix	Last four digits of token. CyberSource includes this field in the reply message when it decrypts the payment blob for the tokenized transaction.	ccAuthReply	String (4)