

**Apple Pay**

REST API

Streamline

# Developer Guide



© 2021. Cybersource Corporation. All rights reserved.

Cybersource Corporation (Cybersource) furnishes this document and the software described in this document under the applicable agreement between the reader of this document (You) and Cybersource (Agreement). You may use this document and/or software only in accordance with the terms of the Agreement. Except as expressly set forth in the Agreement, the information contained in this document is subject to change without notice and therefore should not be interpreted in any way as a guarantee or warranty by Cybersource. Cybersource assumes no responsibility or liability for any errors that may appear in this document. The copyrighted software that accompanies this document is licensed to You for use only in strict accordance with the Agreement. You should read the Agreement carefully before using the software. Except as permitted by the Agreement, You may not reproduce any part of this document, store this document in a retrieval system, or transmit this document, in any form or by any means, electronic, mechanical, recording, or otherwise, without the prior written consent of Cybersource.

## **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 subparagraphs (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 Unpublished rights reserved under the copyright laws of the United States.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**

Authorize.Net, eCheck.Net, and The Power of Payment are registered trademarks of Cybersource Corporation. Cybersource, Cybersource Payment Manager, Cybersource Risk Manager, Cybersource Decision Manager, and Cybersource Connect are trademarks and/or service marks of Cybersource Corporation. Visa, Visa International, Cybersource, the Visa logo, and the Cybersource logo are the registered trademarks of Visa International in the United States and other countries. All other trademarks, service marks, registered marks, or registered service marks are the property of their respective owners.

Version: 21.05

# Contents

- Recent Revisions to This Document..... 4**
- About This Guide..... 5**
- Getting Started.....6**
  - Requirements for Using Apple Pay.....6
  - Supported Processors.....7
  - Enrolling in Apple Pay.....7
    - Generating a New CSR.....8
  - Transaction Request Report..... 8
- Apple Pay Integrations..... 9**
  - In-App Transactions Using the Cybersource API..... 9
    - Cybersource Decryption (In-App Transactions)..... 10
    - Merchant Decryption (In-App Transactions)..... 11
  - Web Transactions.....12
    - Cybersource Decryption (Web Transactions)..... 13
    - Merchant Decryption (Web Transactions)..... 14
    - Configuring Your Apple Pay Requirements..... 14
    - Setting Up Apple Pay JavaScript..... 16
- Requesting Services..... 18**
  - Requesting the Authorization Service..... 18
    - Using Our Decryption Method..... 18
    - Using the Merchant Decryption Method..... 26
  - Requesting Additional Services..... 34
- API Fields..... 35**

# Recent Revisions to This Document

## 21.05

Created a REST API version of this guide.

## 21.04

Changed the name of the **merchantInformation.merchant.url** field to **merchantInformation.merchantDomainName**. See the [API Field Reference for the REST API](#).

## 21.03

Added the **tokenInformation.networkTokenOption** field. See the [API Field Reference for the REST API](#).

## 21.02

Updated the **card\_type** field. See the [API Field Reference for the REST API](#).

## 21.01

Added the field. See the [API Field Reference for the REST API](#).

## 20.03

Updated information about recurring payments. See [Supported Processors](#).

## 20.02

This revision contains only editorial changes and no technical updates.

## 20.01

This revision contains only editorial changes and no technical updates.

# About This Guide


This section provides you with information about the structure and content within this guide.


## Audience and Purpose

This document is written for merchants who want to use Apple Pay in an iOS application and use information from Apple to process payments through Cybersource. This document provides an overview for integrating Apple and Cybersource services into an order management system.

## 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

For further technical documentation, visit the Cybersource Technical Documentation Portal:

<https://docs.cybersource.com/en/index.html>

## Customer Support

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

<http://www.cybersource.com/support>

# Getting Started

## Requirements for Using Apple Pay

In order to use the Cybersource platform to process Apple 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. See [Supported Processors](#).
- An *Admin* or *Team Agent* user of the [Apple Pay Developer](#) account.



### **Important:**

Apple Pay relies on authorizations with payment network tokens. You can sign up for Apple 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 Apple Pay:

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

### **Related information**

[Supported Processors](#)

# Supported Processors

Merchant-initiated transactions, multiple partial captures, and subsequent authorizations are described in the *Authorizations with Payment Network Tokens* guide. Recurring payments and split shipments are described in the *Credit Card Services* guide.


Processor	Card Types	Optional Features
Streamline	<ul style="list-style-type: none"><li>• Maestro (International)</li><li>• Maestro (UK Domestic)</li><li>• Mastercard</li><li>• Visa</li></ul>	<ul style="list-style-type: none"><li>• Multiple partial captures</li><li>• Recurring payments</li><li>• Subsequent authorizations</li></ul>

## Related information

[Authorizations with Payment Network Tokens Developer and Credit Card Services guides \(available on the Payment Services > Credit Card Services page\)](#)

## Enrolling in Apple Pay

1. Log in to the Business Center:
  - Test: <https://ebctest.cybersource.com/ebc2>
  - Production: <https://ebc.cybersource.com/ebc2>
2. On the left navigation panel, click the **Payment Configuration** icon.
3. Click **Digital Payment Solution**. The Digital Payments page appears.
4. Click **Configure**. The Apple Pay Registration panel opens.
5. Enter your Apple Merchant 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.
9. Complete the enrollment process by submitting your CSR to Apple.
10. For information about adding certificates to your Apple Merchant ID, refer to the Apple Pay PassKit: <https://developer.apple.com/documentation/passkit>.
11. Test your software by following the steps in [Requesting the Authorization Service](#).

 **Important:** If you are using a Cybersource test account, you must connect to the Apple developer system and not to the Apple production system.

After you complete your testing, you must create a new CSR for the Cybersource production system, and you must use that CSR for the Apple production system. Until you perform these steps, you cannot enable payments in your iOS application.

12. Repeat Steps 1 through 11 with your Cybersource production account and the Apple production account.

## Generating a New CSR

1. Log in to the Business Center:
  - Test: <https://ebctest.cybersource.com/ebc2>
  - Production: <https://ebc.cybersource.com/ebc2>
2. On the left navigation panel, click the **Payment Configuration** icon.
3. Click **Digital Payment Solution**. The Digital Payments page appears.
4. Click **Configure**. The Apple Pay Registration panel opens.
5. To download your CSR, click the **Download** icon next to the key.
6. Follow your browser's instructions to save and open the file.
7. To edit your Apple Merchant ID, click the **Edit** icon. The Edit CSR panel opens.
8. Modify your merchant ID as necessary, and click **Update**.

## Transaction Request Report

Through the Business Center, you can use the Transaction Request Report to obtain information about your transactions:

- Use the Transaction Search page to identify Apple transactions. You can search for transactions by date, application type, customer name, and other transaction identifiers.
- For information about the Transaction Request Report, see the [Business Center Reporting User Guide](#).



# Apple Pay Integrations

## In-App Transactions Using the Cybersource API

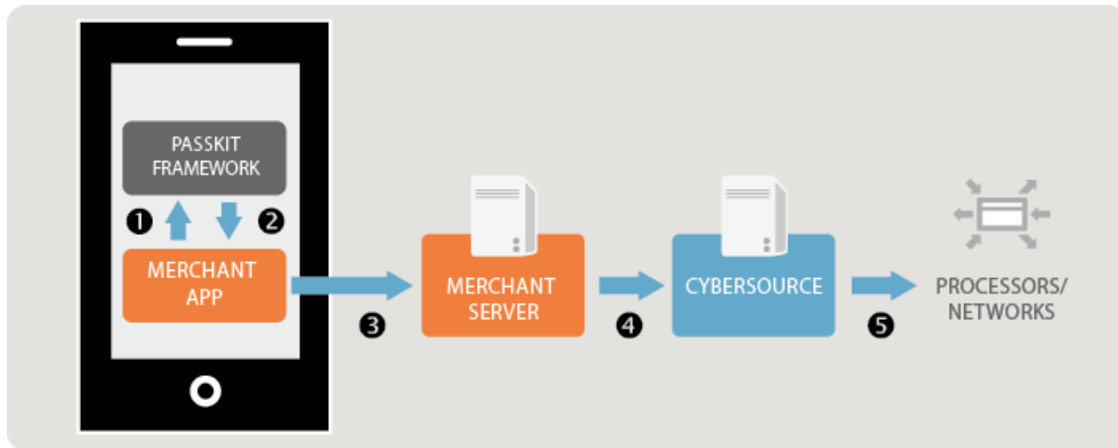
This section describes how in-app transactions are processed using the Cybersource API:

- [Cybersource Decryption \(In-App Transactions\)](#)
- [Merchant Decryption \(In-App Transactions\)](#)

For information about processing web transactions, refer to:

- [Cybersource Decryption \(Web Transactions\)](#)
- [Merchant Decryption \(Web Transactions\)](#)

## Cybersource Decryption (In-App Transactions)



1. When the customer chooses to pay with Apple Pay, you use the Apple PassKit Framework to request the encrypted payment data from Apple.
2. Apple uses the Secure Element to create a payment token (the **PKPaymentToken** structure) and encrypt the token's payment data (the **paymentData** field of the **PKPaymentToken** structure) before it sends your application.
3. You forward the encrypted payment data to your order management system.
4. Using the Cybersource API, you submit the authorization request. In the **paymentInformation.fluidData.value** field, include the Base64-encoded value obtained from the **paymentData** field of the **PKPaymentToken** structure.
5. Cybersource decrypts the payment data and forwards the information to the payment network, which includes your processor and the relevant payment card company.

**! Important:** You must use the Business Center or one of the Cybersource API services to capture, credit, or void the authorization. Refer to the *Credit Card Services* guide (see Related information below) for information.

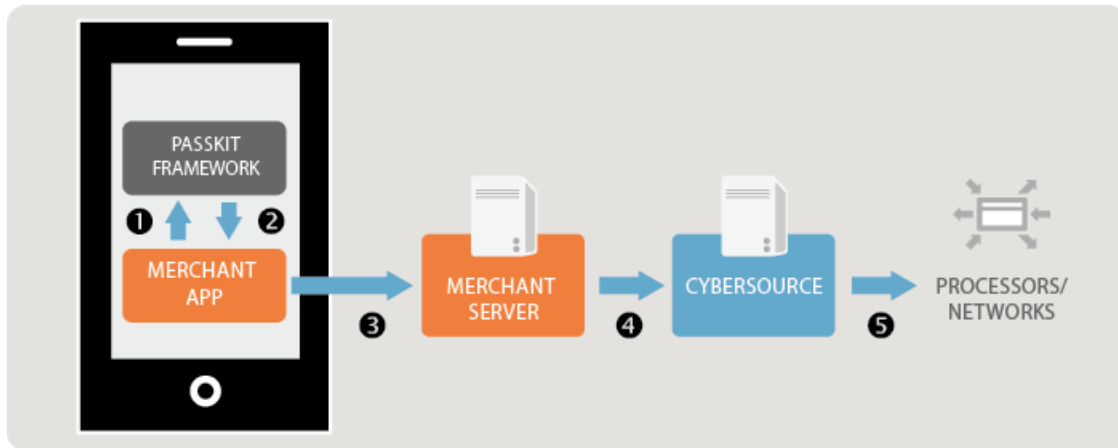
### Related information

[Requesting the Authorization Service for Mastercard Transactions \(Cybersource Decryption\)](#)

[Requesting the Authorization Service for Visa Transactions \(Cybersource Decryption\)](#)

[Credit Card Services guide on Payment Services > Credit Card Services page](#)

## Merchant Decryption (In-App Transactions)



1. When the customer chooses to pay with Apple Pay, you use the Apple PassKit Framework to request the encrypted payment data from Apple.
2. Apple uses the Secure Element to create a payment token (the **PKPaymentToken** structure) and encrypt the token's payment data (the **paymentData** field of the **PKPaymentToken** structure) before it sends your application.
3. You forward the encrypted payment data to your order management system to decrypt. For information on decryption, see: [https://developer.apple.com/library/archive/documentation/PassKit/Reference/PaymentTokenJSON/PaymentTokenJSON.html#//apple\\_ref/doc/uid/TP40014929-CH8-SW1](https://developer.apple.com/library/archive/documentation/PassKit/Reference/PaymentTokenJSON/PaymentTokenJSON.html#//apple_ref/doc/uid/TP40014929-CH8-SW1)
4. Using the Cybersource API, you submit the authorization request and include the decrypted payment data.
5. Cybersource forwards the information to the payment network, which includes your processor and the relevant payment card company.

**! Important:** You must use the Business Center or one of the Cybersource API services to capture, credit, or void the authorization. Refer to the *Credit Card Services* guide (see Related information below) for information.

### Related information

[Requesting the Authorization Service for Mastercard Transactions \(Merchant Decryption\)](#)

[Requesting the Authorization Service for Visa Transactions \(Merchant Decryption\)](#)

[Credit Card Services guide on Payment Services > Credit Card Services page](#)

# Web Transactions

This section describes how web transactions are processed using the Cybersource API:


- [Cybersource Decryption \(Web Transactions\)](#)
- [Merchant Decryption \(Web Transactions\)](#)

For information about processing in-app transactions, refer to:

- [Cybersource Decryption \(In-App Transactions\)](#)
- [Merchant Decryption \(In-App Transactions\)](#)

## Cybersource Decryption (Web Transactions)

1. When the customer chooses to pay with Apple Pay, you use the Apple Pay JavaScript to request the encrypted payment data from Apple.
2. Apple uses the Secure Element to create a payment token (the **PKPaymentToken** structure) and encrypt the token's payment data (the **paymentData** field of the **PKPaymentToken** structure) before it sends your application using the **onpaymentauthorized** callback function.
3. You forward the encrypted payment data to your order management system.
4. Using the Cybersource API, you submit the authorization request. In the **paymentInformation.fluidData.value** field, include the Base64-encoded value obtained from the **paymentData** field of the **PKPaymentToken** structure.
5. Cybersource decrypts the payment data and forwards the information to the payment network, which includes your processor and the relevant payment card company.


 **Important:** You must use the Business Center or one of the Cybersource API services to capture, credit, or void the authorization. Refer to the *Credit Card Services* guide (see Related information below) for information.

### Related information

[Credit Card Services guide on the Payment Services > Credit Card Services page](#)

## Merchant Decryption (Web Transactions)

1. When the customer chooses to pay with Apple Pay, you use the Apple Pay JavaScript to request the encrypted payment data from Apple.
2. Apple uses the Secure Element to create a payment token (the **PKPaymentToken** structure) and encrypt the token's payment data (the **paymentData** field of the **PKPaymentToken** structure) before it sends your application using the **onpaymentauthorized** callback function.
3. You forward the encrypted payment data to your order management system to decrypt. For information on decryption, see: [https://developer.apple.com/library/archive/documentation/PassKit/Reference/PaymentTokenJSON/PaymentTokenJSON.html#//apple\\_ref/doc/uid/TP40014929-CH8-SW1](https://developer.apple.com/library/archive/documentation/PassKit/Reference/PaymentTokenJSON/PaymentTokenJSON.html#//apple_ref/doc/uid/TP40014929-CH8-SW1)
4. Using the Cybersource API, you submit the authorization request and include the decrypted payment data.
5. Cybersource forwards the information to the payment network, which includes your processor and the relevant payment card company.

 **Important:** You must use the Business Center or one of the Cybersource API services to capture, credit, or void the authorization. Refer to the *Credit Card Services* guide (see Related information below) for information.

### Related information

[Credit Card Services guide on the Payment Services > Credit Card Services page](#)

## Configuring Your Apple Pay Requirements

Refer to <https://developer.apple.com/develop> for details about configuring your requirements.

1. Register your merchant ID.

If you are currently processing In-App transactions, you can use the same merchant ID for processing web transactions.

2. Create or upload a Certificate Signing Request (CSR), which is used to encrypt the payment information during the payment process.

If you are using the merchant decryption method, generate a new CSR.

If you are using the Cybersource decryption method, upload the CSR that you created in the Business Center when you enrolled in Apple Pay.

If you are currently processing In-App transactions, you can use the same CSR for processing web transactions.

3. Register your domain. Registration is required in order to use Apple Pay on your website.
4. Create a Merchant Identity Certificate. This certificate is required in order to connect to the Apple servers.

After configuring your Apple Pay requirements, you can set up any optional features (refer to the *Authorizations with Payment Network Tokens* guides).

**Related information**

[Authorizations with Payment Network Tokens guides \(available on the Payment Services > Credit Card Services page\)](#)

[Enrolling in Apple Pay](#)

[Generating a New CSR](#)

## Setting Up Apple Pay JavaScript

Preparing to use Apple Pay JavaScript to accept payments on your website involves several stages of development. This section is an overview of those stages. For detailed instructions, follow the references to the Apple developer site.

- **Enabling the Apple Pay JavaScript API:**

Before you can display an Apple Pay button on your website or create an Apple Pay session, you need to ensure that the Apple Pay JavaScript API is enabled on your device. See [Enabling the Apple Pay JavaScript API](#).

- **Displaying the Apple Pay button:**

Use the CSS templates provided by Apple to display the Apple Pay button on your website. See the Apple developer documentation on [Displaying Apple Pay Buttons Using CSS](#) for more information.

- **Creating the `ApplePaySession` class and object:**

The **`ApplePaySession`** class manages the payment process on your website. The **`ApplePaySession`** object is the entry point for Apple Pay on your website. See Apple's Developer article: For information about creating the **`ApplePaySession`** object, see the Apple developer reference on the [ApplePaySession](#) class.

To create the **`ApplePaySession`** object:

- Use the **Version number** and **Payment request** arguments. The API version is 1. See the Apple developer reference on the [ApplePayPaymentRequest](#) structure for information on how to display the payment form.
- After creating the **`ApplePaySession`**, call its **`begin`** method to display the payment form. You can call this method only when invoked by a user request. See the Apple developer reference on the [begin](#) method for details.

- **Getting merchant validation:**

When the payment form is displayed, the **`onvalidatemerchant`** callback function is called and provides a URL to pass to your server for validating the merchant session. See the Apple developer reference on [onvalidatemerchant](#). Also, refer to the Merchant Validation section of the Apple developer reference on the [ApplePaySession](#) class.

- **Handling payment confirmation:**

When the customer confirms the payment by clicking or tapping the Apple Pay button, the **`onpaymentauthorized`** callback function is invoked and provides the payment token. See the Apple developer reference on the [onpaymentauthorized](#) instance.

- **Handling encrypted payment data:**

The following two methods are available for handling encrypted payment data for Apple Pay transactions:

- **Merchant Decryption:**

This method forwards the encrypted payment data to your order management system to decrypt. For information on decryption, see the Apple [Payment Token Format Reference](#) for details. You use the Cybersource API to submit the authorization request and include the decrypted payment data.

- **Cybersource Decryption:**



This method forwards the encrypted payment data to your order management system. You use the Cybersource API to submit the authorization request and include the Base64-encoded value obtained from the **paymentData** object in the **paymentInformation.fluidData.value** field.

For example:

```
session.onpaymentauthorized = function (event) {  
    var paymentDataString =  
    JSON.stringify(event.payment.token.paymentData);  
    var paymentDataBase64 = btoa(paymentDataString);  
}
```

## Enabling the Apple Pay JavaScript API

1. Verify that the **window.ApplePaySession** class exists.
2. Calls one of the following methods:
  - **canMakePayments**—verifies that the device is enabled for Apple Pay.
  - **canMakePaymentsWithActiveCard**—verifies that the device is enabled for Apple Pay and the customer has a card stored on the device. You can call this method only if Apple Pay is the default payment method during your checkout flow, or if you want to add the Apple Pay button to your product detail page.

# Requesting Services

## Requesting the Authorization Service

You can request the authorization service using the Cybersource decryption method or the merchant decryption method.

### Related information

[Requesting the Authorization Service for Mastercard Transactions \(Cybersource Decryption\)](#)

[Requesting the Authorization Service for Visa Transactions \(Cybersource Decryption\)](#)

[Requesting the Authorization Service for Mastercard Transactions \(Merchant Decryption\)](#)

[Requesting the Authorization Service for Visa Transactions \(Merchant Decryption\)](#)

## Using Our Decryption Method

### Requesting the Authorization Service for Mastercard Transactions (Cybersource Decryption)

1. Set the **paymentInformation.fluidData.value** field to the Base64-encoded value obtained from the **paymentData** property of the **PKPaymentToken** object.
2. Set the **paymentInformation.fluidData.descriptor** field to [RK1EPUNPTU1PTi5BUFBMRS5JTKFQUC5QQVINRU5U](#).
3. Set the **processingInformation.paymentSolution** field to **001**.

### Related information

[API Field Reference for the REST API](#)

[Cybersource Decryption \(In-App Transactions\)](#)

[Cybersource Decryption \(Web Transactions\)](#)

[Relaxed Requirements for Address Data and Expiration Date](#)

## Example (Cybersource Decryption)

### Request (Mastercard Transactions)

```
{
  "clientReferenceInformation": {
    "code": "1234567890"
  },
  "processingInformation": {
    "paymentSolution": "001"
  },
  "paymentInformation": {
    "fluidData": {
      "value": "eyJkYXRhW5FINWZqVjfkak1NdVNSaE96dWF2ZGVyb2c9PSJ9",
      "descriptor": "RklePUNPTU1PTi5BUFBMRS5JTkFQUC5QQVlNRU5U",
      "encoding": "Base64",
    },
    "tokenizedCard": {
      "type": "002",
      "transactionType": "1",
    }
  },
  "orderInformation": {
    "amountDetails": {
      "totalAmount": "100.00",
      "currency": "USD"
    },
    "billTo": {
      "firstName": "Maya",
      "lastName": "Lee",
      "address1": "123 Main St",
      "locality": "SomeCity",
      "administrativeArea": "CA",
      "postalCode": "94404",
      "country": "US",
      "email": "maya.lee@email.world",
      "phoneNumber": "6504327113"
    }
  }
}
```

### Response (Mastercard Transactions)

```
{
  "_links": {
    "authReversal": {
      "method": "POST",
    }
  }
}
```

```

        "href": "/pts/v2/payments/6234236182176225003004/reversals"
    },
    "self": {
        "method": "GET",
        "href": "/pts/v2/payments/6234236182176225003004"
    },
    "capture": {
        "method": "POST",
        "href": "/pts/v2/payments/6234236182176225003004/captures"
    }
},
"clientReferenceInformation": {
    "code": "1234567890"
},
"id": "6234236182176225003004",
"orderInformation": {
    "amountDetails": {
        "authorizedAmount": "100.00",
        "currency": "USD"
    }
},
"paymentInformation": {
    "tokenizedCard": {
        "expirationYear": "2031",
        "prefix": "128945",
        "expirationMonth": "12",
        "suffix": "2398",
        "type": "002"
    },
    "card": {
        "type": "002"
    }
},
"pointOfSaleInformation": {
    "terminalId": "111111"
},
"processingInformation": {
    "paymentSolution": "001"
},
"processorInformation": {
    "approvalCode": "888888",
    "networkTransactionId": "123456789619999",
    "transactionId": "123456789619999",
    "responseCode": "100",
    "avs": {
        "code": "X",
        "codeRaw": "I1"
    }
},

```

```
"reconciliationId": "75729760OPN67ZFV",  
"status": "AUTHORIZED",  
"submitTimeUtc": "2021-06-11T15:00:18Z"  
}
```

## Requesting the Authorization Service for Visa Transactions (Cybersource Decryption)

1. Set the **paymentInformation.fluidData.value** field to the Base64-encoded value obtained from the **paymentData** property of the **PKPaymentToken** object.
2. Set the **paymentInformation.fluidData.descriptor** field to [RK1EPUNPTU1PT15BUFBMRS5JTKFQUC5QQVINRU5U](#).
3. Set the **processingInformation.paymentSolution** field to [001](#).

### Related information

[API Field Reference for the REST API](#)

[Cybersource Decryption \(In-App Transactions\)](#)

[Cybersource Decryption \(Web Transactions\)](#)

[Relaxed Requirements for Address Data and Expiration Date](#)

## Example (Cybersource Decryption)

### Request (Visa Transactions)

```
{
  "clientReferenceInformation": {
    "code": "1234567890"
  },
  "processingInformation": {
    "paymentSolution": "001"
  },
  "paymentInformation": {
    "fluidData": {
      "value": "eyJkYXRhW5FINWZqVjfkak1NdVNSaE96dWF2ZGVyb2c9PSJ9",
      "descriptor": "RklePUNPTU1PTi5BUFBMRS5JTkFQUC5QQVlNRU5U",
      "encoding": "Base64",
    },
    "tokenizedCard": {
      "type": "001",
      "transactionType": "1",
    }
  },
  "orderInformation": {
    "amountDetails": {
      "totalAmount": "100.00",
      "currency": "USD"
    },
    "billTo": {
      "firstName": "Maya",
      "lastName": "Lee",
      "address1": "123 Main St",
      "locality": "SomeCity",
      "administrativeArea": "CA",
      "postalCode": "94404",
      "country": "US",
      "email": "maya.lee@email.world",
      "phoneNumber": "6504327113"
    }
  }
}
```

### Response (Visa Transactions)

```
{
  "_links": {
    "authReversal": {
```

```

        "method": "POST",
        "href": "/pts/v2/payments/6234236182176225003004/reversals"
    },
    "self": {
        "method": "GET",
        "href": "/pts/v2/payments/6234236182176225003004"
    },
    "capture": {
        "method": "POST",
        "href": "/pts/v2/payments/6234236182176225003004/captures"
    }
},
"clientReferenceInformation": {
    "code": "1234567890"
},
"id": "6234236182176225003004",
"orderInformation": {
    "amountDetails": {
        "authorizedAmount": "100.00",
        "currency": "USD"
    }
},
"paymentInformation": {
    "tokenizedCard": {
        "expirationYear": "2031",
        "prefix": "411111",
        "expirationMonth": "12",
        "suffix": "1111",
        "type": "001"
    },
    "card": {
        "type": "001"
    }
},
"pointOfSaleInformation": {
    "terminalId": "111111"
},
"processingInformation": {
    "paymentSolution": "001"
},
"processorInformation": {
    "approvalCode": "888888",
    "networkTransactionId": "123456789619999",
    "transactionId": "123456789619999",
    "responseCode": "100",
    "avs": {
        "code": "X",
        "codeRaw": "I1"
    }
}

```



```
},  
  "reconciliationId": "75729760OPN67ZFY",  
  "status": "AUTHORIZED",  
  "submitTimeUtc": "2021-06-11T15:00:18Z"  
}
```

## Using the Merchant Decryption Method

### Requesting the Authorization Service for Mastercard Transactions (Merchant Decryption)

1. Set the **paymentInformation.tokenizedCard.number** field to the payment network token value.
2. Set the **paymentInformation.tokenizedCard.expirationMonth** and **paymentInformation.tokenizedCard.expirationYear** fields to the values from the payment network token expiration date.
3. Set the **consumerAuthenticationInformation.ucafAuthenticationData** field to the 3D Secure cryptogram of the payment network token.
4. Set the **paymentInformation.tokenizedCard.cryptogram** field to the network token cryptogram.
5. Set the **consumerAuthenticationInformation.ucafCollectionIndicator** field to [2](#).
6. Set the **paymentInformation.tokenizedCard.transactionType** field to [1](#).
7. Set the **consumerAuthenticationInformation.ecommerceIndicator** field to [spa](#).
8. Set the **processingInformation.paymentSolution** field to [001](#).

#### Related information

[API Field Reference for the REST API](#)

[Merchant Decryption \(In-App Transactions\)](#)

[Merchant Decryption \(Web Transactions\)](#)

[Relaxed Requirements for Address Data and Expiration Date](#)

# Example (Merchant Decryption)

## Authorization Request (Mastercard Transactions)

```
{
  "clientReferenceInformation": {
    "code": "1234567890"
  },
  "processingInformation": {
    "paymentSolution": "001",
    "commerceIndicator": "spa"
  },
  "paymentInformation": {
    "tokenizedCard": {
      "number": "5432543254325432",
      "expirationMonth": "12",
      "expirationYear": "2031",
      "cryptogram": "ABCDEFabcdefABCDEFabcdef0987654321234567",
      "transactionType": "1",
      "type": "002"
    }
  },
  "orderInformation": {
    "amountDetails": {
      "totalAmount": "100.00",
      "currency": "USD"
    },
    "billTo": {
      "firstName": "Maya",
      "lastName": "Lee",
      "address1": "123 Main St",
      "locality": "SomeCity",
      "administrativeArea": "CA",
      "postalCode": "94404",
      "country": "US",
      "email": "maya.lee@email.world",
      "phoneNumber": "6504327113"
    }
  },
  "consumerAuthenticationInformation": {
    "ucafAuthenticationData": "ABCDEFabcdefABCDEFabcdef0987654321234567",
    "ucafCollectionIndicator": "2"
  }
}
```

## Authorization Response (Mastercard Transactions)

```
{
  "_links": {
    "authReversal": {
      "method": "POST",
      "href": "/pts/v2/payments/6234236182176225003004/reversals"
    },
    "self": {
      "method": "GET",
      "href": "/pts/v2/payments/6234236182176225003004"
    },
    "capture": {
      "method": "POST",
      "href": "/pts/v2/payments/6234236182176225003004/captures"
    }
  },
  "clientReferenceInformation": {
    "code": "1234567890"
  },
  "id": "6234236182176225003004",
  "orderInformation": {
    "amountDetails": {
      "authorizedAmount": "100.00",
      "currency": "USD"
    }
  },
  "paymentInformation": {
    "tokenizedCard": {
      "expirationYear": "2031",
      "prefix": "543254",
      "expirationMonth": "12",
      "suffix": "5432",
      "type": "002"
    },
    "card": {
      "type": "002"
    }
  },
  "pointOfSaleInformation": {
    "terminalId": "111111"
  },
  "processingInformation": {
    "paymentSolution": "001"
  },
  "processorInformation": {
    "approvalCode": "888888",
    "networkTransactionId": "123456789619999",
  }
```

```
    "transactionId": "123456789619999",
    "responseCode": "100",
    "avs": {
      "code": "X",
      "codeRaw": "I1"
    }
  },
  "reconciliationId": "75729760OPN67ZFY",
  "status": "AUTHORIZED",
  "submitTimeUtc": "2021-06-11T15:00:18Z"
}
```

## Requesting the Authorization Service for Visa Transactions (Merchant Decryption)

1. Set the **paymentInformation.tokenizedCard.number** field to the payment network token value.
2. Set the **paymentInformation.tokenizedCard.expirationMonth** and **paymentInformation.tokenizedCard.expirationYear** fields to the values from the payment network token expiration date.
3. Set the **consumerAuthenticationInformation.cavv** field to the 3D Secure cryptogram of the payment network token.

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

4. Set the **consumerAuthenticationInformation.ecommerceIndicator** field to the ECI value contained in the Apple Pay response payload (**5**=vbv and **7**=internet).
5. Set the **paymentInformation.tokenizedCard.cryptogram** field to the network token cryptogram.
6. Set the **paymentInformation.tokenizedCard.transactionType** field to **1**.
7. Set the **processingInformation.paymentSolution** field to **001**.

### Related information

[API Field Reference for the REST API](#)

[Merchant Decryption \(In-App Transactions\)](#)

[Merchant Decryption \(Web Transactions\)](#)

[Relaxed Requirements for Address Data and Expiration Date](#)

## Example (Merchant Decryption)

### Authorization Request (Visa Transactions)

```
{
  "clientReferenceInformation": {
    "code": "1234567890"
  },
  "processingInformation": {
    "paymentSolution": "001",
    "commerceIndicator": "internet"
  },
  "paymentInformation": {
    "tokenizedCard": {
      "number": "4111111111111111",
      "expirationMonth": "12",
      "expirationYear": "2031",
      "cryptogram": "AceY+igABPs3jdwNaDg3MAACAAA=",
      "transactionType": "1",
      "type": "001"
    }
  },
  "orderInformation": {
    "amountDetails": {
      "totalAmount": "100.00",
      "currency": "USD"
    },
    "billTo": {
      "firstName": "Maya",
      "lastName": "Lee",
      "address1": "123 Main St",
      "locality": "SomeCity",
      "administrativeArea": "CA",
      "postalCode": "94404",
      "country": "US",
      "email": "maya.lee@email.world",
      "phoneNumber": "6504327113"
    }
  },
  "consumerAuthenticationInformation": {
    "cavv": "AceY+igABPs3jdwNaDg3MAACAAA="
  }
}
```

### Authorization Response (Visa Transactions)

```
{
```

```

    "_links": {
      "authReversal": {
        "method": "POST",
        "href": "/pts/v2/payments/6234236182176225003004/reversals"
      },
      "self": {
        "method": "GET",
        "href": "/pts/v2/payments/6234236182176225003004"
      },
      "capture": {
        "method": "POST",
        "href": "/pts/v2/payments/6234236182176225003004/captures"
      }
    },
    "clientReferenceInformation": {
      "code": "1234567890"
    },
    "id": "6234236182176225003004",
    "orderInformation": {
      "amountDetails": {
        "authorizedAmount": "100.00",
        "currency": "USD"
      }
    },
    "paymentInformation": {
      "tokenizedCard": {
        "expirationYear": "2031",
        "prefix": "411111",
        "expirationMonth": "12",
        "suffix": "1111",
        "type": "001"
      },
      "card": {
        "type": "001"
      }
    },
    "pointOfSaleInformation": {
      "terminalId": "111111"
    },
    "processingInformation": {
      "paymentSolution": "001"
    },
    "processorInformation": {
      "approvalCode": "888888",
      "networkTransactionId": "123456789619999",
      "transactionId": "123456789619999",
      "responseCode": "100",
      "avs": {
        "code": "X",

```



```
        "codeRaw": "I1"
    },
    "reconciliationId": "75729760OPN67ZfV",
    "status": "AUTHORIZED",
    "submitTimeUtc": "2021-06-11T15:00:18Z"
}
```

# Requesting Additional Services

To request the following additional services, refer to the *Credit Card Services* guide (see Related information below):

- **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.

## Related information

[Credit Card Services guide on the Payment Services > Credit Card Services page](#)  
[Payment Network Tokenization](#)

# API Fields

See the [API Field Reference guide](#) for your API type.