PIX by Iugu Integration

Simple Order API

Developer Guide

cybersource
A Visa Solution

Version: 23.01

Contents

Recent Revisions to This Document **4**

About This Guide **5**

Introduction to PIX by Iugu **6**

PIX Transaction Workflow 7

Sale **9**

Required Fields for a Sale 10

Optional Fields for a Sale 11

Line Items 12

Example: Processing a Sale Using the Simple Order API with XML 13

Example: Processing a Sale Using the Simple Order API with Name-Value Pairs 15

Check Status **17**

Required Fields for a Check Status 18

Example: Processing a Check Status with XML 19

Example: Processing a Check Status with Name-Value Pairs 20

Webhooks **21**

Required Fields for Creating an Alternative Payment Notification Webhook Using the REST API 22

Optional Fields for Creating an Alternative Payment Notification Webhook Using the REST API 23

Example: Creating Alternative Payment Notification Webhooks Using the REST API 24

Refund **26**

Required Fields for a Refund 27

Example: Processing a Refund with XML 28

Example: Processing a Refund with Name-Value Pairs 30

Reference Information **31**

Reason Codes for PIX for the Simple Order API 32

Reporting 36

Recent Revisions to This Document

23.02

Updated the transaction workflow. See PIX Transaction Workflow (on page 7).

23.01

Added the Reporting section. See Reporting (on page 36).

Updated optional line item fields section. See Line Items (on page 12).

22.01

Initial release.

About This Guide

This section describes the audience and purpose of this guide as well as conventions and related documentation.

Audience and Purpose

This guide is written for application developers who want to use the Simple Order API to integrate PIX by Iugu with their payment 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.

Related Documentation

Refer to these sites for technical documentation:

- Technical Documentation Portal: https://docs.cybersource.com/en/index.html
- Support Center: https://www.cybersource.com/en-us/support/technical-documentation.html

Customer Support

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

http://www.cybersource.com/support

Introduction to PIX by Iugu

PIX is a Brazilian alternative payment method created by the Central Bank of Brazil. PIX allows customers to make instant payments on your e-commerce website using their checking account or savings account.

Iugu is a platform that automates financial operations and online payments, approved by the Central Bank of Brazil to allow e-commerce websites to generate the QR code and the copy-and-paste string for PIX payments.

Supported Services

Cybersource supports sale, check status, and full refund payment services for PIX by Iugu transactions.

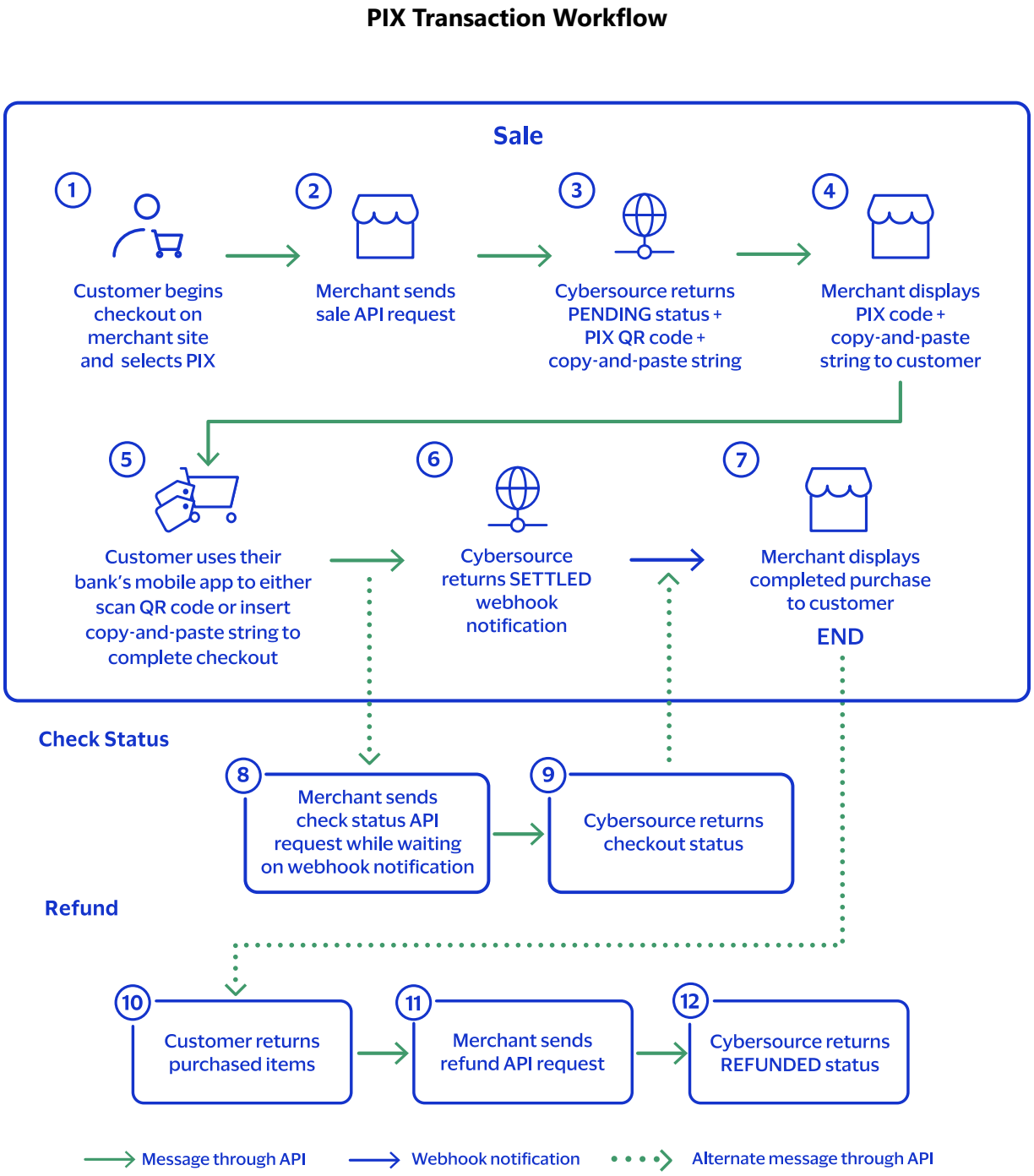
Requirements

You must obtain an Iugu account in order to begin processing PIX payments. Cybersource will send you an email requesting all the necessary documentation for your PIX account to be created.

Important: The account owner receives notification within 48 hours if the documentation provided was accepted and approved. If the documentation is rejected, new submissions are required until approval.

PIX Transaction Workflow

This workflow describes a successful PIX transaction.



1. Customer chooses the goods to be purchased from your website and selects PIX by Iugu as their payment method.
2. Merchant sends a sale API request. See Sale (on page 9).
3. Cybersource responds with a `PENDING` status, a PIX QR code, and a copy-and-paste string.
4. Merchant displays the PIX QR code and the copy-and-paste string to the customer on the merchant site.
5. Customer uses their bank's mobile app to either scan the PIX QR code or enter the copy-and-paste string to complete the checkout.
6. Cybersource responds with a `SETTLED` webhook notification.
7. Merchant displays confirmation of the completed payment to the customer.
8. Merchant sends a check status API request when they are waiting on a webhook notification and want to know if there is a status update for the payment. See Check Status (on page 17).
9. Cybersource responds with a either a `PENDING` or `SETTLED` status.
10. Customer returns a purchased item.
11. Merchant sends a refund API request. See Refund (on page 26).
12. Cybersource returns a `REFUNDED` status.

Sale

When the customer makes a purchase on your site, you send the sale service request. This returns a unique QR code and copy-and-paste string to the PIX transaction. The customer either scans the QR code using their banking app, or copies and pastes the copy-and-paste string into their banking app to confirm their payment.

Important: All sales must be at least $1 BRL.

Response Status

The sale service responds with one of these statuses:

- `FAILED`: the sale cannot be completed.

  To understand why a transaction has failed, check the `reasonCode` field in the check status response and see Reason Codes.

- `PENDING`: the payment is waiting for customer's approval. When you receive a pending status, wait and request the check status service.

The sale service also responds with a reason code as the **apAuthReply_reasonCode** field value. For more information on reason codes, see the Reason Codes for PIX for the Simple Order API (on page 32).

Endpoints

Set the **apSaleService_run** field to `true` and send the request to one of these endpoints:

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

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

Required Fields for a Sale

These fields are required in a sale request:

apPaymentType

Set the value of this field to `PIX`.

apSaleService_run

Set the value of this field to `true`.

billTo_email

merchantID

merchantReferenceCode

purchaseTotals_currency

Set the value of this field to `BRL`.

purchaseTotals_grandTotalAmount

Related information

API Field Reference for the Simple Order API

Optional Fields for a Sale

These fields are optional in a sale request:

ap_sale_transaction_timeout

billTo_street1

billTo_street2

billTo_city

billTo_country

billTo_district

billTo_state

billTo_postalCode

billTo_company

billTo_email

billTo_firstName

billTo_ipAddress

billTo_lastName

billTo_phoneNumber

item_#_productDescription

Replace the `#` character with the number `0` and increase in numerical order for any additional items.

item_#_quantity

Replace the `#` character with the number `0` and increase in numerical order for any additional items.

item_#_totalAmount

Replace the `#` character with the number `0` and increase in numerical order for any additional items.

item_#_unitPrice

Replace the `#` character with the number `0` and increase in numerical order for any additional items.

Related information

API Field Reference for the Simple Order API

Line Items

PIX allows optional line items when you send a sale request. *Line items* are used to include information about the goods that your customers purchase, such as product description, quantity, and price.

Line items are represented as the **item_#_** fields, starting with **item_0_**, and increasing in numerical order.

These fields are required for each line item that you use:

item_#_productDescription

Brief description of item.

item_#_quantity

Quantity of the product.

item_#_totalAmount

Total amount for the items.

item_#_unitPrice

Per-item price of the product.

Including Line Items in a Service Request

This example shows three valid line items.

```
<item id="0">
 <unitPrice>5.00</unitPrice>
 <quantity>1</quantity>
 <totalAmount>5.00</totalAmount>
 <productDescription>Blue hat</productDescription>
</item>
<item id="1">
 <unitPrice>7.50</unitPrice>
 <quantity>2</quantity>
 <totalAmount>15.00</totalAmount>
 <productDescription>Medium hat</productDescription>
</item>
<item id="2">
 <unitPrice>10.00</unitPrice>
 <quantity>3</quantity>
 <totalAmount>30.00</totalAmount>
 <productDescription>Large hat</productDescription>
</item>
```

Example: Processing a Sale Using the Simple Order API with XML

Request

Use one of these production endpoints:

- `https://ics2ws.ic3.com/commerce/1.x/transactionProcessor`
- `https://ics2wsa.ic3.com/commerce/1.x/transactionProcessor`

Important: Use XML schema version 1.203 or newer.

```
<requestMessage xmlns="urn:schemas-cybersource-com:transaction-data-1.203">
    <merchantID>pspops_altpay_tester</merchantID>
    <merchantReferenceCode>iugupixrefno1d</merchantReferenceCode>
    <billTo>
        <email>test@cybs.com</email>
    </billTo>
    <item id="0">
        <unitPrice>5.00</unitPrice>
        <quantity>1</quantity>
        <totalAmount>5.00</totalAmount>
        <productDescription>Blue hat</productDescription>
    </item>
    <item id="1">
        <unitPrice>7.50</unitPrice>
        <quantity>2</quantity>
        <totalAmount>15.00</totalAmount>
        <productDescription>Medium hat</productDescription>
    </item>
    <purchaseTotals>
        <currency>BRL</currency>
        <grandTotalAmount>20.00</grandTotalAmount>
    </purchaseTotals>
    <apPaymentType>PIX</apPaymentType>
    <apSaleService run="true">
    </apSaleService>
</requestMessage>
```

Response

```
<replyMessage xmlns:c="urn:schemas-cybersource-com:transaction-data-1.203">
    <merchantReferenceCode>iugupixrefno1d</merchantReferenceCode>
    <requestID>6637054069676672404007</requestID>
    <decision>ACCEPT</decision>
    <reasonCode>100</reasonCode>

  <requestToken>AxjnrwSTZ8ugx8BawconAL8ZYjWmdpw4jyoTBsojkVCMU32hf0CNeGTSTL0Yr+4sEmz5dBj4C1
g5ROAAyxSR</requestToken>
    <purchaseTotals>
        <currency>BRL</currency>
    </purchaseTotals>
    <apReply>
        <transactionExpirationDate>20221020</transactionExpirationDate>
    </apReply>
    <apSaleReply>
        <reasonCode>100</reasonCode>
        <paymentStatus>pending</paymentStatus>
        <responseCode>00001</responseCode>

  <merchantURL>https://faturas.iugu.com/iugu_pix/c23f73f4-6c70-4250-a270-53a3480a6739-9b
8d/test/qr_code</merchantURL>
        <processorTransactionID>C23F73F46C704250A27053A3480A6739</processorTransactionID>
        <reconciliationID>XFZ3Z88GJB06</reconciliationID>
        <amount>20.00</amount>
        <processorResponse>00001</processorResponse>
        <dateTime>2022-09-20T20:23:28Z</dateTime>

  <additionalData>http://faturas.iugu.com/iugu_pix/c23f73f4-6c70-4250-a270-53a3480a6739-9b
8d/test/pay</additionalData>
    </apSaleReply>
</replyMessage>
```

Related reference

Reason Codes for PIX for the Simple Order API *(on page 32)*

Example: Processing a Sale Using the Simple Order API with Name-Value Pairs

Request

Use one of these production endpoints:

- `https://ics2ws.ic3.com/commerce/1.x/transactionProcessor`
- `https://ics2wsa.ic3.com/commerce/1.x/transactionProcessor`

```
merchantID=jdoe
merchantReferenceCode=TIGUG-2
billTo_email=test@cybs.com
item_0_quantity=2
item_0_unitPrice=2.16
item_0_totalAmount=4.32
item_0_productDescription=female skirt in blue and green
item_1_quantity=2
item_1_unitPrice=5.00
item_1_totalAmount=10.00
item_1_productDescription=A small drone fly automatically
purchaseTotals_currency=BRL
purchaseTotals_grandTotalAmount=14.32
apPaymentType=PIX
apSaleService_run=true
```

Response

```
reasonCode=100
apSaleReply_reasonCode=100
apSaleReply_amount=14.32
apSaleReply_processorResponse=00001
apSaleReply_processorTransactionID=ECB946F1C317431389FA5E9C30B125B7
apSaleReply_reconciliationID=TIGUG265468Sale
merchantReferenceCode=TIGUG-2
apSaleReply_dateTime=2022-09-01T22:17:51Z
decision=ACCEPT
apReply_transactionExpirationDate=20221001
requestID=662070669440614610300 8
apSaleReply_additionalData=http://
faturas.iugu.com/iugu_pix/ecb946f1-c317-4313-89fa-5e9c30b125b7-6787/test/pay
apSaleReply_paymentStatus=pending
requestToken=0
```

```
purchaseTotals_currency=BRL
apSaleReply_merchantURL=https://
faturas.iugu.com/iugu_pix/ecb946f1-c317-4313-89fa-5e9c30b125b7-6787/test/qr_code
```

Related reference

Reason Codes for PIX for the Simple Order API *(on page 32)*

Check Status

The check status service returns the current status of a PIX transaction.

When requesting the check status service, you must use the value of the **requestID** field from the sale response as the value for the **apCheckStatusService_checkStatusRequestID** field.

Response Status

The check status service responds with one these statuses:

- `ABANDONED`: the customer did not complete the transaction within the time limit.
- `FAILED`: the sale cannot be completed. To understand why a transaction failed, check the **reasonCode** field in the check status response and see Reason Codes.
- `PENDING`: payment is waiting for authentication or approval. When you receive a pending status, wait and request the check status service again.
- `REFUNDED`: the refund is complete. The funds have been deposited to the customer's account.
- `SETTLED`: payment has been confirmed and the funds transferred from the customer to PIX. The customer can receive the purchased goods.

The check status service also responds with a reason code as the **apAuthReply_reasonCode** field value. For more information on reason codes, see the Reason Codes for PIX for the Simple Order API (on page 32).

Endpoints

Set the **apCheckStatusService_run** field to `true` and send the request to one of these endpoints:

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

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

Required Fields for a Check Status

These fields are required in a check status request:

apCheckStatusService_checkStatusRequestID

apCheckStatusService_run

Set the value of this field to `true`.

apPaymentType

Set the value of this field to `PIX`.

merchantID

merchantReferenceCode

Related information

API Field Reference for the Simple Order API

Example: Processing a Check Status with XML

Request

Use one of these production endpoints:

- `https://ics2ws.ic3.com/commerce/1.x/transactionProcessor`
- `https://ics2wsa.ic3.com/commerce/1.x/transactionProcessor`

Important: Use XML schema version 1.203 or newer.

```
<requestMessage xmlns="urn:schemas-cybersource-com:transaction-data-1.203">
    <merchantID>npr_iugu_occ</merchantID>
    <merchantReferenceCode>iugupixrefno123</merchantReferenceCode>
        <apPaymentType>PIX</apPaymentType>
    <apCheckStatusService run="true">
    <checkStatusRequestID>6637054069676672404007</checkStatusRequestID>
    </apCheckStatusService>
</requestMessage>
```

Response

```
<replyMessage xmlns:c="urn:schemas-cybersource-com:transaction-data-1.203">
    <merchantReferenceCode>iugupixrefno1d</merchantReferenceCode>
    <requestID>6637054588706739704009</requestID>
    <decision>ACCEPT</decision>
    <reasonCode>100</reasonCode>

  <requestToken>AxjnrwSTZ8uin851i9DJAL8ZYjWmrVw4jyLEWmojkVCNKL2hf0CNeGTSTL0Yr+4sEmz5dBj4C1
g5ROAAsxPw</requestToken>
    <apCheckStatusReply>
        <reasonCode>100</reasonCode>
        <reconciliationID>XFZ5588GHXES</reconciliationID>
        <paymentStatus>settled</paymentStatus>
        <processorResponse>00004</processorResponse>
    </apCheckStatusReply>
</replyMessage>
```

Related reference

Reason Codes for PIX for the Simple Order API *(on page 32)*

Example: Processing a Check Status with Name-Value Pairs

Request

Use one of these production endpoints:

- `https://ics2ws.ic3.com/commerce/1.x/transactionProcessor`
- `https://ics2wsa.ic3.com/commerce/1.x/transactionProcessor`

```
merchantID=npr_iugu_occ
merchantReferenceCode=TIGUG-2
apPaymentType=PIX
apCheckStatusService_run=true
apCheckStatusService_checkStatusRequestID=662747909717641740 3008
```

Response

```
apCheckStatusReply_reasonCode=100
apCheckStatusReply_processorResponse=00001
reasonCode=100
decision=ACCEPT
requestToken=AxjnrwSTZ0brka0buAAAAL8rUkx6sdk2atG0PRlx66fTD16c1EciPx3K+0LswRmQyaSZe
jFs59Ck2dGv6T+JuyAgAAAAPglA
requestID=6627491460836591403008
apCheckStatusReply_paymentStatus=pending
```

Related reference

Reason Codes for PIX for the Simple Order API *(on page 32)*

Webhooks

Webhooks are automated notifications generated by system events that occur in your organizations. You can enroll in a sale or refund event and designate a URL to receive notifications when the event updates.

Important: Webhooks are only supported using the REST API. Pix by Iugu payments are only supported using the Simple Order API. Consider the different APIs when integrating webhooks for Pix by Iugu.

Notifications that contain sensitive, personally identifiable information, such as account numbers, are sent using message-level encryption.

Transport Layer Security is required in order to ensure data integrity.

Response Status

Webhooks responds with one of these statuses:

- `SETTLED`
- `CANCELLED`

Webhooks API Developer Guide

For more information about webhooks and subscriptions, see **Managing Subscriptions** in the *Webhooks API Developer Guide*.

Create New Webhooks

For more information about how to create an alternative payment notification webhook, see the **Create a Webhook** in the *Cybersource REST API Reference*.

Required Fields for Creating an Alternative Payment Notification Webhook Using the REST API

Use these fields to create an alternative payment notification webhook.

healthCheckUrl

organizationId

products.eventTypes

Set the value of this field to `payments.payments.updated`.

products.productId

Set the value of this field to `alternativePaymentMethods`.

webhookUrl

Optional Fields for Creating an Alternative Payment Notification Webhook Using the REST API

Use these optional fields to create an alternative payment notification webhook.

description

name

notificationScope.scopeData

Required if you submit the **notificationScope.scope** field. All organizations subscribing to the webhook must be listed in the value, separated by commas.

notificationScope.scope

For more information, see Using the Notification Scope.

securityPolicy.securityType

securityPolicy.securityConfigurations.{propertyName}

Example: Creating Alternative Payment Notification Webhooks Using the REST API

Use these examples to create an alternative payment notification webhook.

Endpoints

- **Production**: `https://api.cybersource.com/notification-subscriptions/v2/webhooks`
- **Test**: `https://apitest.cybersource.com/notification-subscriptions/v1/webhooks`

Request

```
{
  "name": "My Custom Webhook",
  "description": "Sample Webhook from Developer Center",
  "organizationId": "organizationId",
  "productId": "alternativePaymentMethods",
  "eventTypes": [
    "payments.payments.updated"
  ],
  "webhookUrl": "https://MyWebhookServer.com:8443/simulateClient",
  "healthCheckUrl": "https://MyWebhookServer.com:8443/simulateClientHealthCheck",
  "notificationScope": "SELF",
  "retryPolicy": {
   "algorithm": "ARITHMETIC",
    "firstRetry": 1,
    "interval": 1,
    "numberOfRetries": 3,
    "deactivateFlag": "false",
    "repeatSequenceCount": 0,
    "repeatSequenceWaitTime": 0
  },
  "securityPolicy": {
   "securityType": "KEY",
   "proxyType": "external"
  }
}
```

Encrypted Response


```
{
  "eventType": "payments.payments.updated",
  "productId": "alternativePaymentMethods",
  "organizationId": "npr_iugu",
  "transactionTraceId": "6584251287300178228949",
  "payload":
 "{CBC}
nL1LK67IH1LQKad3qyhilPoeDGrHo/6ClIZ17HSC5YX9TQFIgCOhVPi139kmB/xiVCALGdU8YjqmOEwo3i
UXKy6oq517eDiTHv/Crq0pOiEzs0dtuTvYhriXrpvQCF+XxqfqTPPFyus/10/FlyEK7ZKIc66DrmA3Ifdl
YTpL7gICAOttQrMWNGODFu7JXaFyQahDarhFCPsvWMwyRK9yOLwihWj3lC1yPD/1xA7xpTGp9kX3ABKzzZ
xKjHoC5OYOhSLQm7MPnN4BND+Hs
+cU/mYzLYxfN7GSDSbqZwhbJgXaKapT8lMeXUTlPejyqGyCPagWNqZJFHx6q1OBJKB8XZ0pZIpT+v
+bMe5EnbQyv9Nj6BJX1BBgqhvKmXrSqD16",
      "metadata": {
          "notificationTypeID": "5",
          "notificationReason": "payment"
       }
}
```


Decrypted Response


```
{
  "id": "658425128730017822894 9",
  "reconcillationID": "85100194sale",
  "status": "settled",
  "submitTimeUTC": "2022-07-21T17:38:48Z",
  "clientReferenceInformation.code": "TC85100-1",
  "orderInformation.amountDetails.totalAmount": "1999.99",
  "orderInformation.amountDetails.currency": "BRL"

}
```

Refund

The refund service uses the request ID value returned by the sale service to link the refund to the payment.

> **Important:** The payment status must be `SETTLED` before you refund a payment and return funds to the customer's account.

When requesting the refund service, you must use the value in the **requestID** field from the sale response for the **apRefundService_refundRequestID** field value.

Response Status

The refund service responds with one of these statuses:

- `FAILED`: the refund cannot be completed. No funds have been returned to the customer. Send the refund service request again.
- `REFUNDED`: the refund is complete. The funds have been deposited to the customer’s account by PIX.

The refund service also responds with a reason code as the **apAuthReply_reasonCode** field value. For more information on reason codes, see the Reason Codes for PIX for the Simple Order API (on page 32).

Endpoints

Set the **apRefundService_run** field to `true` and send the request to one of these endpoints:

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

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

Required Fields for a Refund

These fields are required in a refund request:

apPaymentType

Set the value of this field to `PIX`.

apRefundService_refundRequestID

apRefundService_run

Set the value of this field to `true`.

purchaseTotals_currency

Set the value of this field to `BRL`.

purchaseTotals_grandTotalAmount

merchantID

merchantReferenceCode

Related information

API Field Reference for the Simple Order API

Example: Processing a Refund with XML

Request

Use one of these production endpoints:

- `https://ics2ws.ic3.com/commerce/1.x/transactionProcessor`
- `https://ics2wsa.ic3.com/commerce/1.x/transactionProcessor`

Important: Use XML schema version 1.203 or newer.

```
<requestMessage xmlns="urn:schemas-cybersource-com:transaction-data-1.203">
    <merchantID>npr_iugu_occ</merchantID>
    <merchantReferenceCode>iugupixrefno123</merchantReferenceCode>
    <purchaseTotals>
        <currency>BRL</currency>
        <grandTotalAmount>20.00</grandTotalAmount>
    </purchaseTotals>
    <apPaymentType>PIX</apPaymentType>
    <apRefundService run="true">
        <refundRequestID>6637054069676672404007</refundRequestID>
    </apRefundService>
</requestMessage>
```

Response

```
<replyMessage xmlns:c="urn:schemas-cybersource-com:transaction-data-1.203">
    <merchantReferenceCode>iugupixrefno1d</merchantReferenceCode>
    <requestID>6637055304926748604009</requestID>
    <decision>ACCEPT</decision>
    <reasonCode>100</reasonCode>

  <requestToken>AxjnrwSTZ8ulKzRswfZpAL8ZYjWmdpw4jyoTBsojkVCOBr2hf0CNeGTSTL0Yr+4sEmz5dBj4C1
g5ROAAnxRT</requestToken>
    <purchaseTotals>
        <currency>BRL</currency>
    </purchaseTotals>
    <apRefundReply>
        <reasonCode>100</reasonCode>
        <transactionID>C23F73F46C704250A27053A3480A6739</transactionID>
        <status>REFUNDED</status>
        <processorResponse>00006</processorResponse>
```

```
        <amount>20.00</amount>
        <dateTime>2022-09-20T20:25:51Z</dateTime>
        <reconciliationID>XFZ3Z88GJB06</reconciliationID>
        <returnRef>XFZ3Z88GJB0B</returnRef>
        <paymentStatus>refunded</paymentStatus>
        <responseCode>00006</responseCode>
    </apRefundReply>
</replyMessage>
```

Related reference

Reason Codes for PIX for the Simple Order API *(on page 32)*

Example: Processing a Refund with Name-Value Pairs

Request

Use one of these production endpoints:

- `https://ics2ws.ic3.com/commerce/1.x/transactionProcessor`
- `https://ics2wsa.ic3.com/commerce/1.x/transactionProcessor`

```
merchantID=npr_iugu_occ
merchantReferenceCode=TIGUG-2
purchaseTotals_currency=BRL
purchaseTotals_grandTotalAmount=14.32
apPaymentType=PIX
apRefundService_run=true
apRefundService_refundRequestID=662747909717641740 3008
```

Response

```
apRefundReply_dateTime=2022-09-09T19:11:03Z
decision=ACCEPT
apRefundReply_returnRef=HB2JG7TJUBPN
apRefundReply_amount=14.32
apRefundReply_reasonCode=100
apRefundReply_status=REFUNDED
apRefundReply_paymentStatus=refunded
requestID=662750661902689920 3008
purchaseTotals_currency=BRL
requestToken=AxjnrwSTZ0cha/hQc/PAAL8ZYjWmdlvSbRpcikojkR+k2f2hdmCMyGTSTL0YtnPoUmz
o1/SfxN2QEAAAaQuw
apRefundReply_responseCode=00006
reasonCode=100
apRefundReply_transactionID=CD9FC199000A40349C70013835E18941
merchantReferenceCode=TIGUG-2
```

Related reference
Reason Codes for PIX for the Simple Order API *(on page 32)*

Reference Information

This section contains reference information that is useful when integrating PIX by Iugu.

- Reason Codes for PIX for the Simple Order API (on page 32)
- Reporting (on page 36)

Reason Codes for PIX for the Simple Order API

Important: Response fields and reason codes can be added at any time. Therefore:

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

The following table describes the reason codes that are returned by the Simple Order API.

Reason Codes

Reason Code	Processor Response Code	Description
100	• 00000—status: completed. • 00001—status: pending. • 00002—status: abandoned. • 00003—status: authorized. • 00004—status: settled. • 00006—status: refunded. • 00008—status: reversed. • 00009—status: cancelled. • 00010—status: accepted. • 00011—status: chargeback. • 00012—status: settle_initiated. • 00013—status: settle_accepted. • 00014—status: refund_initiated. • 00015—status: active. • 00016—status: revoked. • 00017—status: expired. • 00020—status: refund_rejected.	Transaction was successful.
101		Request is missing one or more required fields. Examine the response fields **missingField_0** through **missingField_N** to identify which fields are missing. Resend the request with all the required fields.

Reason Codes (continued)

Reason Code	Processor Response Code	Description
102	• 10000—status: failed.	One or more fields in the request contain invalid data. Examine the response fields **invalidField_0** through **invalidField_N** to identify which fields are invalid. Resend the request with valid data.
150	• 20000—status: failed. • 20001—status: failed. • 20002—status: failed. • 30000—status: failed. • 30100—status: failed.	System error. You must design your transaction management system to include a way to correctly handle system errors. Depending on which payment processor is handling the transaction, the error might indicate a valid Cybersource system error, or it might indicate a processor rejection because of invalid data. In either case, do not design your system to endlessly try to resend a transaction when a system error occurs. See the documentation for the Cybersource client (SDK) that you are using for important information about how to handle system errors and retries.
151		Request was received but a server timeout occurred. This error does not include timeouts that occur between the client and the server. To avoid duplicating the transaction, do not resend the request until you have reviewed the transaction status in the Business Center. See the documentation for your Cybersource client for information about handling retries in the case of system errors.
152		Request was received, but a service did not finish running in time. To avoid duplicating the transaction, do not resend the request until you have reviewed the transaction status in the Business Center. See the documentation for your Cybersource client for information about handling retries in the case of system errors.
153		Your account is not enabled for the OCT service. Contact customer support to have your account enabled for this service.
202		Payment method is expired. You might also receive this value if the expiration date that you provided does not match the date that the issuing bank has on file. Request a different form of payment.

Reason Codes (continued)

Reason Code	Processor Response Code	Description
203	• 30200—status: failed. • 30400—status: failed. • 30500—status: failed.	Payment method was declined. No other information was provided by the issuing bank. Request a different form of payment.
204	• 30350—status: failed.	Account does not contain sufficient funds. Request a different form of payment.
223	• 30600—status: failed. • 30700—status: failed.	Processor declined the transaction due to tax errors or government compliance errors.
233	• 30600—status: failed. • 30700—status: failed.	Processor declined the payment method. For more information about the decline, search for the transaction in the Business Center and view the transaction details. Request a different form of payment.

Related information

Getting Started with Cybersource Advanced for the Simple Order API

Reporting

You can generate various types of reports for your financial and reconciliation data. For more information on how to generate these reports, see the *Reporting User Guide*.

The *Reporting User Guide* contains these relevant topics:

- How and When Reports Are Generated
- Downloading Available Reports
- Subscribing to Standard Reports

Additional Resources

For additional information about how to use the Business Center, see these helpful resources.

Business Center Overview

For an overview of the various resources available in the Business Center, see this YouTube video: `https://www.youtube.com/watch?v=UDmAWGHPbWs`

Business Center Navigation

For a step-by-step demonstration of how to navigate in the Business Center, see this YouTube video:

`https://www.youtube.com/watch?v=2qi_g2DParI`

Managing Report Subscriptions

For an overview of how to manage report subscriptions in the Downloadable Reports section in the Business Center, see this YouTube video:

`https://www.youtube.com/watch?v=tFlmkXtvxWE`

Downloading Reports

For an overview of how to download available reports in the Reports section in the Business Center, see this YouTube video:

`https://www.youtube.com/watch?v=E0slUYjJvmw`