Payments

REST API
Lloyds TSB Cardnet

cybersource
A Visa Solution

Developer Guide

Cybersource Contact Information

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

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

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

Copyright

Restricted Rights Legends

Trademarks

Confidentiality Notice

Revision

25.05.01

Contents

Payments Developer Guide **8**
- Recent Revisions to This Document 9

Introduction to Payments **11**
- Financial Institutions and Payment Networks 11
  - Merchant Financial Institutions (Acquirers) 11
  - Customer Financial Institutions (Issuers) 12
  - Payment Networks 12
  - Payment Processors 12
- Card Types 13
  - Credit Cards 13
  - Debit Cards 13
- Transaction Types 13
  - Card-Not-Present Transactions 13
  - Authorizations with Card Verification Numbers 14
  - International Transactions 15
  - Token Management Service 15
- Payment Services 16
  - Authorizations 17
  - Sales 19
  - Authorization Reversals 19
  - Captures 20
  - Credits 21
  - Voids 22
- Payment Features 22
  - Debit and Prepaid Card Payments 23
  - Payer Authentication 23
  - Introduction to Credentialed Transactions 24
  - Token Management Service 25
- Testing the Payment Services 26
  - Requirements for Testing 27
  - Test Card Numbers 27
  - Using Amounts to Simulate Errors 28

Test American Express Card Verification....28

Standard Payment Processing....29

Basic Authorizations....29

Declined Authorizations....29

Required Fields for Processing a Basic Authorization....30

REST Interactive Example: Processing a Basic Authorization....30

REST Example: Processing a Basic Authorization....30

Authorizations with Line Items....33

Required Fields for Processing an Authorization with Line Items....34

REST Example: Processing an Authorization with Line Items....35

Authorizations with Payment Network Tokens....37

Required Fields for Authorizations with Payment Network Tokens....37

Optional Fields for Authorizations with Payment Network Tokens....38

REST Example: Authorizations with Payment Network Tokens....39

Authorizations with a Card Verification Number....40

Required Fields for Processing an Authorization with a Card Verification Number....42

Optional Fields for Processing an Authorization with a Card Verification Number....43

REST Example: Processing an Authorization with a Card Verification Number....43

Authorizations with Strong Customer Authentication Exemption....45

Required Fields for Processing an Authorization with an SCA Exemption....47

REST Example: Processing an Authorization with an SCA Exemption for Low-Value Transactions....47

Zero Amount Authorizations....49

Required Fields for Processing a Zero Amount Authorization....49

REST Example: Processing a Zero Amount Authorization....50

Pre-Authorizations....52

Required Fields for a Pre-Authorization....52

REST Example: Processing a Pre-Authorization....53

Final Authorization Indicator....55

Requirements for Final Authorizations....56

Pre-Authorizations....56

Unmarked Authorizations....56

Undefined Authorizations....57

Required Fields for Final Authorizations....57

REST Example: Final Authorizations....58

Authorization Reversal....59

Required Fields for Processing an Authorization Reversal....60

REST Example: Processing an Authorization Reversal....60

Sales....61

Required Fields for Processing a Sale....61

REST Example: Processing a Sale....62

Sales with Payment Network Tokens....63

Required Fields for Sales with Payment Network Tokens....64

Optional Fields for Sales with Payment Network Tokens 64
REST Example: Sales with Payment Network Tokens 65
Captures 67
Required Fields for Capturing an Authorization 67
REST Example: Capturing an Authorization 67
Multiple Partial Captures 68
Required Fields for Processing Multiple Partial Captures 69
REST Example: Processing Multiple Partial Captures 69
Refunds 70
Required Fields for Processing a Refund 70
REST Interactive Example: Processing a Refund 71
REST Example: Processing a Refund 71
Credits 73
Required Fields for Processing a Credit 73
REST Interactive Example: Processing a Credit 74
REST Example: Processing a Credit 74
Voids for a Capture or Credit 75
Required Fields for Voiding a Capture or Credit 76
REST Example: Voiding a Capture or Credit 76
Debit and Prepaid Card Processing 77
Additional Resources for Debit and Prepaid Payments 77
Processing Debit and Prepaid Authorizations 77
Required Fields for Processing Debit and Prepaid Authorizations 77
Optional Field for Processing Debit and Prepaid Authorizations 78
REST Example: Processing Debit and Prepaid Authorizations 78
Enabling Debit and Prepaid Partial Authorizations 80
Required Fields for Enabling Debit and Prepaid Partial Authorizations 81
Optional Field for Enabling Debit and Prepaid Partial Authorizations 81
REST Example: Enabling Debit and Prepaid Partial Authorizations 82
Disabling Debit and Prepaid Partial Authorizations 83
Required Field for Disabling Debit and Prepaid Partial Authorizations 84
Optional Field for Disabling Debit and Prepaid Partial Authorizations 84
REST Example: Disabling Debit and Prepaid Partial Authorizations 85
Payer Authentication Processing 87
Additional Resources for Payer Authentication 87
Providing Payer Authentication Information for Authorization 87
Mastercard Identity Check 89
Required Fields for Processing an Authorization Using Mastercard Identity Check 90
REST Example: Processing an Authorization Using Mastercard Identity Check 91
Visa Secure 93
Required Fields for Processing an Authorization Using Visa Secure 93
REST Example: Validating and Authorizing a Transaction 94
Processing Payments Using Credentials 97
Additional Resources for Credentialed Transactions 97

Customer-Initiated Transactions with Credentials on File........ 97
Storing Customer Credentials with a CIT and PAN........ 98
Storing Customer Credentials with a CIT and TMS........ 100
Retrieving Stored Customer Credentials During a CIT........ 105
Delayed Transaction........ 107
Merchant-Initiated Delayed Transaction with PAN........ 107
Merchant-Initiated Delayed Transaction with TMS........ 110
No-Show Transactions........ 118
Merchant-Initiated No-Show Transactions with PAN........ 119
Merchant-Initiated No-Show Transaction with TMS........ 122
Reauthorization Transaction........ 130
Merchant-Initiated Reauthorization Transactions with PAN........130
Merchant-Initiated Reauthorization Transactions with TMS........133
Recurring Payments........ 141
Customer-Initiated Recurring Payment with PAN........ 142
Customer-Initiated Recurring Payment with TMS........145
Customer-Initiated Recurring Payment with Enrollable Network Tokens........150
Merchant-Initiated Recurring Payments with PAN........155
Merchant-Initiated Recurring Payments with TMS........ 159
Mastercard Standing Order Payments........167
Mastercard Initial CIT Standing Order Payment........167
Mastercard Initial CIT Standing Order Payment with TMS........ 169
Mastercard Subscription Payments........173
Mastercard CIT Initial Subscription Payment........ 173
Mastercard CIT Initial Subscription Payment with TMS........ 174
Unscheduled COF Payments........179
Customer-Initiated Unscheduled COF Payment with PAN........179
Customer-Initiated Unscheduled COF Payments with TMS........ 182
Customer-Initiated Unscheduled COF Payment with Enrollable Network Tokens........186
Merchant-Initiated Unscheduled COF Payments with PAN........190
Merchant-Initiated Unscheduled COF Payments with TMS........193
Token Management Service Processing........202
Additional Resources for TMS........ 202
Authorizing a Payment with a Customer Token........202
Required Fields for Authorizing a Payment with a Customer Token........ 203
REST Example: Authorizing a Payment with a Customer Token........ 203
REST Example: Authorizing a Payment Using a Customer Token Linked to a Network Token........205
Authorizing a Payment with a Non-Default Shipping Address........ 207
Required Fields for Authorizing a Payment with a Non-Default Shipping Address........207
REST Example: Authorizing a Payment with a Non-Default Shipping Address........207
Authorizing a Payment with a Non-Default Payment Instrument........ 209

- Required Fields for Authorizing a Payment with a Non-Default Payment Instrument ... 209
- Optional Fields for Authorizing a Payment with a Non-Default Payment Instrument ... 210
- REST Example: Authorizing a Payment with a Non-Default Payment Instrument ... 210

Authorizing a Payment with a Payment Instrument ... 212
- Required Fields for Authorizing a Payment with a Payment Instrument ... 212
- Optional Fields for Authorizing a Payment with a Payment Instrument ... 212
- REST Example: Authorizing a Payment with a Payment Instrument ... 213

Authorize a Payment with an Instrument Identifier ... 215
- Required Fields for Authorizing a Payment with an Instrument Identifier ... 215
- REST Example: Authorizing a Payment with an Instrument Identifier ... 215
- REST Example: Authorizing a Payment with an Instrument Identifier While Creating TMS Tokens ... 217

Authorize a Payment While Ignoring Network Token ... 219
- Required Fields for Authorizing a Payment While Ignoring Network Token ... 219
- REST Example: Authorizing a Payment While Ignoring Network Token ... 220

Authorizing a Payment with a Legacy Token ... 222
- Required Fields for Authorizing a Payment with a Legacy Token ... 222
- REST Example: Authorizing a Payment with a Legacy Token ... 222

Making a Credit with a Customer Token ... 224
- Required Fields for Making a Credit with a Customer Token ... 224
- REST Example: Making a Credit with a Customer Token ... 224

Making a Credit with a Non-Default Payment Instrument ... 226
- Required Fields for Making a Credit with a Non-Default Payment Instrument ... 226
- Optional Fields for Making a Credit with a Non-Default Payment Instrument ... 226
- REST Example: Making a Credit with a Non-Default Payment Instrument ... 227

Making a Credit with a Payment Instrument ... 228
- Required Fields for Making a Credit with a Payment Instrument ... 228
- REST Example: Making a Credit with a Payment Instrument ... 229

Making a Credit with an Instrument Identifier ... 230
- Required Fields for Making a Credit with an Instrument Identifier ... 230
- REST Example: Making a Credit with an Instrument Identifier ... 231

Making a Credit with a Legacy Token ... 232
- Required Fields for Making a Credit with a Legacy Token ... 232
- REST Example: Making a Credit with a Legacy Token ... 232

Payments Developer Guide

This section describes how to use this guide and where to find further information.

Audience and Purpose

This guide is written for application developers who want to use the REST API to integrate payment card processing into an order management system.

Implementing the Cybersource payment services requires software development skills. You must write code that uses the API request and response fields to integrate the credit card services into your existing order management system.

Conventions

These statements appear 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

Visit the *Cybersource documentation hub* to find additional processor-specific

	versions of this guide and additional technical documentation.
Customer Support	**For support information about any service, visit the Support Center:** *http://support.visaacceptance.com*

Recent Revisions to This Document

25.05.01

International Transaction Compliance	**Added a section about international transaction compliance. See** *Compliance* **on page 15.**

25.04.01

This revision contains only editorial changes and no technical updates.

25.03

This revision contains only editorial changes and no technical updates.

25.02

This revision contains only editorial changes and no technical updates.

25.01

Added a testing section. See *Testing the Payment Services* on page 26.

Credentialed Transactions	**Removed Mastercard required field for retrieving customer credentials during a CIT request. See** *Card-Specific Required Field for Retrieving Customer Credentials During a CIT* **on page 105.**

24.14

This revision contains only editorial changes and no technical updates.

24.13

This revision contains only editorial changes and no technical updates.

24.12

This revision contains only editorial changes and no technical updates.

24.11

This revision contains only editorial changes and no technical updates.

24.10

This revision contains only editorial changes and no technical updates.

24.09

This revision contains only editorial changes and no technical updates.

24.08

This revision contains only editorial changes and no technical updates.

24.07

Pre-Authorizations	**Added pre-authorization processing. See** *Pre-Authorizations* **on page 52.**

Introduction to Payments

This introduction provides the basic information that you will need to successfully process payment transactions. It also provides an overview of the payments industry and provides workflows for each process.

With Cybersource payment services, you can process payment cards (tokenized or non-tokenized), digital payments such as Apple Pay and Google Pay, and customer ID transactions. You can process payments across the globe and across multiple channels with scalability and security. Cybersource supports a large number of payment cards and offers a wide choice of gateways and financial institutions, all through one connection.

Visit the *Cybersource documentation hub* to find additional processor-specific versions of this guide and additional technical documentation.

Financial Institutions and Payment Networks

Financial institutions and payment networks enable payment services. These entities work together to complete the full payment cycle.

Merchant Financial Institutions (Acquirers)

A merchant financial institution, also known as an acquirer, offers accounts to businesses that accept payment cards. Before you can accept payments, you must have a merchant account from an acquirer. Your merchant account must be configured to process card-not-present, card-present, or mail-order/telephone-order (MOTO) transactions.

Each acquirer has connections to a limited number of payment processors. You must choose a payment processor that your acquirer supports.

You can expect your acquirer to charge these fees:

- Discount rates: your acquirer charges a fee and collects a percentage of every transaction. The combination of the fee and the percentage is called the discount rate. These charges can be bundled (combined into a single charge) or unbundled (charged separately).

- Interchange fees: payment networks, such as Visa or Mastercard, each have a base fee, called the interchange fee, for each type of transaction. Your acquirer and processor can show you ways to reduce this fee.
- Chargebacks: when cardholders dispute charges, you can incur chargebacks. A chargeback occurs when a charge on a customer's account is reversed. Your acquirer removes the money from your account and could charge you a fee for processing the chargeback.

Take these precautions to prevent chargebacks:

- Use accurate merchant descriptors so that customers can recognize the transactions on their statements.
- Provide good customer support.
- Ensure rapid problem resolution.
- Maintain a high level of customer satisfaction.
- Minimize fraudulent transactions.

If excessive chargebacks or fraudulant changes occur, these actions might be taken:

- You might be required to change your business processes to reduce the number chargebacks, fraud, or both.
- Your acquiring institution might increase your discount rate.
- Your acquiring institution might revoke your merchant account.

Contact your sales representative for information about products that can help prevent fraud.

Customer Financial Institutions (Issuers)

A customer financial institution, also known as an issuer, provides payment cards to and underwrites lines of credit for their customers. The issuer provides monthly statements and collects payments. The issuer must follow the rules of the payment card companies to which they belong.

Payment Networks

Payment networks manage communications between acquiring financial institutions and issuing financial institutions. They also develop industry standards, support their brands, and establish fees for acquiring institutions.
Some payment networks, such as Visa, Mastercard, and UnionPay International, are trade associations that do not issue cards. Issuers are members of these associations, and they issue cards under license from the association.
Other networks, such as Discover and American Express, issue their own cards. Before you process cards from these companies, you must sign agreements with them.

Payment Processors

Payment processors connect with acquirers. Before you can accept payments, you must register with a payment processor. An acquirer might require you to use a payment processor with an existing relationship with the acquirer.

Your payment processor assigns one or more merchant IDs (MIDs) to your business. These unique codes identify your business during payment transactions.
This table lists the processors and corresponding card types that are supported for payment services.

Important

Only the card types explicitly listed here are supported.

Payment Processors and Supported Card Types

Payment Processor	Supported Card Types	Notes
LloydsTSB Cardnet	Visa, Mastercard, Maestro (UK Domestic)	

Card Types

You can process payments with these kinds of cards:

- Credit cards
- Debit cards

Credit Cards

Cardholders use credit cards to borrow money from issuing banks to pay for goods and services offered by merchants that accept credit cards.

Debit Cards

A debit card is linked to a cardholder's checking account. A merchant who accepts the debit card can deduct funds directly from the account.

Transaction Types

This topic provides information about transaction types that are supported by your processor, such as card-present, card-not-present, and international transactions.

Card-Not-Present Transactions

When a customer provides a card number, but the card and the customer are not physically present at the merchant's location, the purchase is known as a card-not-present transaction. Typical card-not-present transactions are internet and phone transactions. Card-not-present transactions pose an additional level of risk to your business because the customer's identification cannot be verified. You can reduce that risk by using features such as the Address Verification System (AVS) and Card Verification Numbers (CVNs). The AVS and CVNs provide additional protection from fraud by verifying the validity of the customer's information and notifying you when discrepancies occur.

Authorizations with Card Verification Numbers

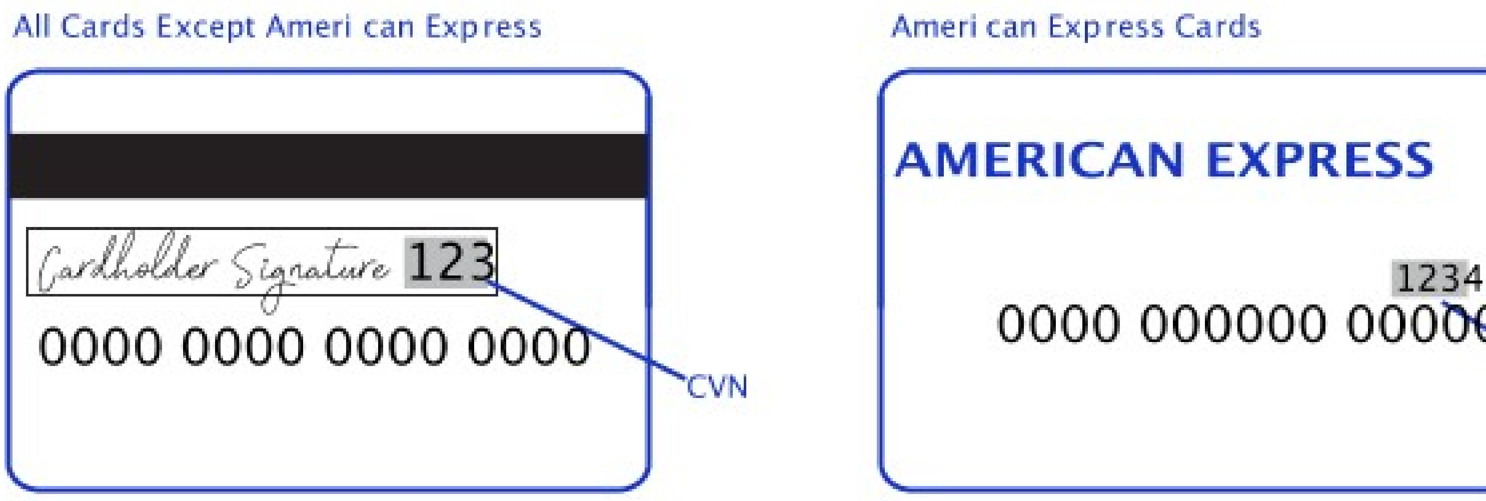
Card verification numbers (CVNs) are a required feature for the authorization service. The CVN is printed on a payment card, and only the cardholder can access it. The CVN is used in card-not-present transactions as a verification feature. Using the CVN helps reduce the risk of fraud.
CVNs are not included in payment card track data and cannot be obtained from a card swipe, tap, or dip.
CVNs must not be stored after authorization.

Important

In Europe, Visa mandates that you not include a CVN for mail-order transactions and not record a CVN on any physical format such as a mail-order form.

CVN Locations and Terminology

For most cards, the CVN is a three-digit number printed on the back of the card, to the right of the signature field. For American Express, the CVN is a four-digit number printed on the front of the card above the card number.



CVN Locations

Each payment card company has its own name for the CVN value:

- American Express and Discover call it the Card Identification Number (CID).

- JCB calls it the Card Authentication Value (CAV2).
- Mastercard calls it the Card Validation Code (CVC2).
- Visa calls it the Card Verification Value (CVV2).

International Transactions

Consider compliance and merchant remittance funding when processing international transactions.

Compliance

Accepting payments from a country other than your own requires that you observe the processing rules and practices of the payment systems in that country. The following list describes areas of compliance that are especially important.

- Merchant descriptor requirements—A merchant descriptor communicates merchant information to customers to remind them of the circumstances that triggered a payment. Merchant descriptors reduce the possibility of a chargeback. Accordingly, the merchant descriptor displayed on a customer's statement should be a close match to the name on your website. It is not good practice to consolidate multiple websites into a single merchant account and use a generic descriptor that more-or-less covers all offerings.
- Excessive chargebacks—To prevent an excessive number of chargebacks, you must maintain good customer support, rapid problem resolution, a high level of customer satisfaction, and transaction management processes that minimize fraudulent transactions. When payment card chargebacks become excessive, you must change business processes to reduce chargebacks. If chargebacks are not reduced to a satisfactory level, your account can be terminated.

Merchant Remittance Funding

You can request that the transaction proceeds be converted to another currency. Currency conversion uses a foreign exchange rate to calculate the conversion to the requested currency. The foreign exchange rate might be explicitly stated as a rate or implicitly stated as a transaction amount. The funded amount and can vary from day to day. The foreign exchange rate might also include an increase for the foreign exchange risk, sales commissions, and handling costs.

Token Management Service

The Token Management Service (TMS) tokenizes, securely stores, and manages customer and payment data. TMS enables you to:

- Securely store a customer's payment details and their billing and shipping addresses.
- Create a network token of a customer's payment card.

TMS simplifies your PCI DSS compliance. TMS passes back to you tokens that represent this data. You then store these tokens in your environment and databases instead of customer payment details.
TMS Token Types

- Customer — Stores the buyer's email address and the merchant's account ID for that buyer plus any other custom fields.
- Shipping Address — Stores a shipping address for a specific customer.
- Instrument Identifier — Stores either a payment card number or a bank account number and routing number
  This resource creates either:
  - An Instrument Identifier token using details of a payment card or an ACH bank account.
  - A payment network token using the details of a payment card; also uses the card expiration date and billing address, which are pass-through only fields.
- Payment Instrument — Stores a Payment Instrument using an Instrument Identifier token. It does not store the card number and cannot exist without an associated Instrument Identifier. It stores:
  - Card expiration date
  - Billing address

  You can also choose to store this information yourself instead and store only the card number or bank account and routing number in an Instrument Identifier object.
- Customer Payment Instrument — Creates and stores a payment instrument for a specific customer ID and an Instrument Identifier token.

TMS Features

- Create, retrieve, update, and delete tokens.
- Set a default payment instrument and shipping address for a customer.
- Process follow-on payment transactions with token IDs.
- Create and update tokens through bundled payment transactions.

Important

Due to mandates from the Reserve Bank of India, Indian merchants cannot store personal account numbers (PAN). Use network tokens instead. For more information on network tokens, see the Network Tokenization section of the *Token Management Service Guide.*

Payment Services

This section describes various services for processing payments.
These services enable customers to purchase goods and services. They also enable merchants to receive payments from customer accounts, to provide refunds, and to void transactions.

Authorizations

An authorization confirms that a payment card account holds enough funds to pay for a purchase. Authorizations can be made online or offline.

Micropayment Authorizations

Micropayments are payments for less than one unit in the transaction’s currency.
For LloydsTSB Cardnet, Cybersource supports micropayment authorizations for Mastercard and Visa payment cards.

Online Authorizations

Online authorizations provide immediate confirmation of funds availability. The customer's financial institution also reduces the amount of credit available in the customer's account, setting aside the authorized funds for the merchant to capture at a later time. Authorizations for most payment cards are processed online. Typically, it is safe to start fulfilling the order when you receive an authorization confirmation.
An online authorization confirmation and the subsequent hold on funds expire after a specific length of time. Therefore it is important to capture funds in a timely manner. The issuing bank sets the expiration time interval, but most authorizations expire within 5 to 7 days.
The issuing bank does not inform Cybersource when an authorization confirmation expires. By default, the authorization information for each transaction remains in the Cybersource database for 180 days after the authorization date. To capture an authorization that expired with the issuing bank, you can resubmit the authorization request.

Offline Authorizations

Online transactions require an internet connection. In situations where the internet is not available, for example, due to an outage, merchants can continue to take credit card payments using offline transactions. An offline authorization is an authorization request for which you do not receive an immediate confirmation about the availability of funds. Offline authorizations have a higher level of risk than online transactions because they do not confirm funds availability or set aside the funds for later capture. Further, it can take up to 5 days to receive payment confirmations for offline transactions. To mitigate this risk, merchants may choose to fulfill orders only after receiving payment confirmation.

Incremental Authorizations

Incremental authorizations are useful when a customer adds products and services to a purchase. After a successful initial authorization, you can request subsequent authorizations and request one capture for the initial authorization and the incremental authorizations.
The incremental authorization service is not the same as the incremental authorization scenario for a merchant-initiated transaction.

Scenario for the Incremental Authorization Service

This sequence is an example of how incremental authorizations work:

1. The customer reserves a hotel room for two nights at a cost of 200.00 per night. You request an authorization for 400.00. The authorization request is approved.
2. The customer orders dinner through room service the first night. You request an incremental authorization of 50.00 for the dinner.
3. The customer decides to stay an extra night. You request an incremental authorization of 200.00 for the additional night.
4. The customer uses items from the mini-bar. The cost of the mini-bar items is 50.00. You request an incremental authorization of 50.00.
5. When the customer checks out, they sign a receipt for 700.00, which is the total of all costs incurred.
6. You request a capture for 700.00.

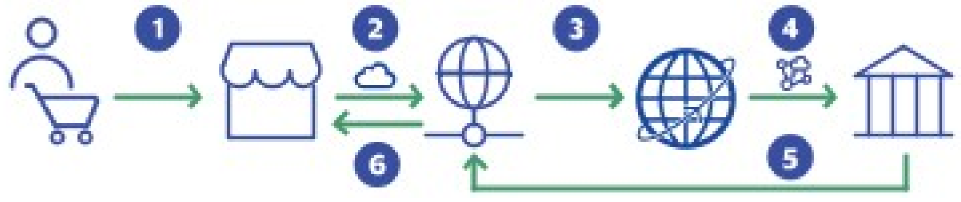
Pre-Authorizations

A pre-authorization enables you to authorize a payment when the final amount is unknown. It is typically used for lodging, auto rental, e-commerce, and restaurant transactions.
For a pre-authorization:

- The authorization amount must be greater than zero.
- The authorization must be submitted for capture within 30 calendar days of its request.
- When you do not capture the authorization, you must reverse it.
  In the U.S., Canada, Latin America, and Asia Pacific, Mastercard charges an additional fee for a pre-authorization that is not captured and not reversed.
  In Europe, Russia, Middle East, and Africa, Mastercard charges fees for all pre-authorizations.
- Chargeback protection is in effect for 30 days after the authorization.

Authorization Workflow

This image and description show the authorization workflow:



1. The customer purchases goods or services from the merchant using a payment card.
2. You send an authorization request over secure internet connection to Cybersource. When the customer buys a digitally delivered product or service, you can request both the authorization and the capture at the same time. When the customer buys a physically fulfilled product, do not request the capture until you ship the product.
3. Cybersource validates the order information then contacts your payment processor and requests authorization.
4. The processor sends the transaction to the payment card company, which routes it to the issuing bank for the customer's payment card. Some card companies, including Discover and American Express, act as their own issuing banks.

5. The issuing bank approves or declines the request.
    - If funds are available, the issuing bank reserves the amount of the authorization request and returns an authorization approval to Cybersource.
    - If the issuing bank denies the request, it returns an authorization denial to Cybersource.
6. Cybersource runs its own tests then tells you whether the authorization succeeded.

Sales

A sale is a bundled authorization and capture. Some processors and acquirers require a sale transaction instead of using separate authorization and capture requests. For other processors and acquirers, you can request a sale instead of a separate authorization and capture when you provide the goods or services immediately after taking an order.
There are two types of sale processing: dual-message processing and single-message processing.

Dual-Message Processing

Dual-message processing is a two-step process. The authorization is processed first. If the authorization is successful, the capture is processed immediately afterward. The response includes the authorization and the capture information. If the authorization is declined, the capture is not processed, and the response message includes only the authorization information.

Partial Authorizations

All debit and prepaid card processors as well as a limited number of credit card processors support partial authorizations when dual-message processing is in place.
When partial authorization is enabled, the issuing financial institution can approve a partial amount when the balance on the card is less than the requested amount. When a partial amount is authorized, the capture is not processed. The merchant can then use a second card to cover the balance, adjust the total cost, or void the transaction.

Single-Message Processing

Single-message processing treats the authorization and capture as a single transaction. There are important differences between dual-message processing and single-message processing:

- Single-message processing treats the request as a full-financial transaction, and with a successful transaction, funds are immediately transferred from the customer account to the merchant account.
- Authorization and capture amounts must be the same.
- Some features cannot be used with single-message processing.

Authorization Reversals

The authorization reversal service releases the hold that an authorization placed on a customer’s payment card funds.

Each card-issuing financial institution has its own rules for deciding whether an authorization reversal succeeds or fails. When a reversal fails, contact the card-issuing financial institution to learn whether there is a different way to reverse the authorization. If your processor supports authorization reversal after void (ARAV), you can reverse an authorization after you void the associated capture. If your processor does not support ARAV, you can use the authorization reversal service only for an authorization that has not been captured and settled.

An authorization reversal is a follow-on transaction that uses the request ID returned from an authorization. The main purpose of a follow-on transaction is to link two transactions. The request ID links the follow-on transaction to the original transaction. The authorization request ID is used to look up the customer’s billing and account information in the Cybersource database. You are not required to include those fields in the full authorization reversal request. The original transaction and follow-on transaction are linked in the database and in the Business Center.

For processors that support debit cards and prepaid cards, the full authorization reversal service works for debit cards and prepaid cards in addition to credit cards.

Important

You cannot perform an authorization reversal if a transaction is in a review state, which can occur if you use a fraud management service. You must reject the transaction prior to authorization reversal. For more information, see the fraud management documentation in the Business Center.

Captures

A capture is a follow-on transaction to an authorization. It is used to transfer the authorized funds from the customer's account to the merchant account. To link the authorization transaction to the capture transaction, you include a request ID in your capture request. This request ID is returned to you in the authorization response.

Captures are typically not performed in real time. They are placed in a batch file and sent to the processor, and the processor settles all of the captures at one time. In most cases, these batch files are sent and processed outside of the merchant's business hours. It usually takes 2 to 4 days for the acquiring financial institution to deposit the funds into the merchant account.

When fulfilling only part of a customer’s order, do not capture the full amount of the authorization. Capture only the cost of the delivered items. When you deliver the remaining items, request a new authorization, and then capture the new authorization.

Important

It is not possible to perform a capture if a transaction is in a review state, which can occur if you use a fraud management service. You must accept the transaction prior to capture. For more information, see the fraud management documentation in the Business Center.

Capture Workflow

The capture workflow begins when you send a request for a capture.

1. The merchant sends a request for a capture to Cybersource.
2. For online captures, Cybersource validates the order information then sends an online capture to the payment processor. For offline captures, Cybersource stores the capture request in a batch file and sends the batch file to the payment processor after midnight.
3. The processor validates the request and forwards it to the issuing bank.
4. The issuing bank transfers funds to the acquiring bank.

Important

The payment processor does not notify Cybersource that the money has been transferred. To ensure that all captures are processed correctly, you should reconcile your capture requests with the capture reports from your processor.

Credits

Credits are payment refunds from a merchant to the cardholder after a cardholder pays for a product or service and that payment is captured by the merchant. When a credit request is successful, the issuer transfers funds from the merchant bank (acquirer) account to the customer's account. It typically takes 2 to 4 days for the acquirer to transfer funds from your merchant account.

Warning

You should carefully control access to the credit service. Do not request this service directly from your customer interface. Instead, incorporate this service as part of your customer service process. This process reduces the potential for fraudulent transactions.

There are two basic types of credits: refunds and stand-alone credits.

Refunds

Refunds, also known as follow-on credits, use the capture request ID to link the refund to a specific transaction. This request ID is returned during the capture request (also known as a settlement) and is used in all subsequent refunds associated with the original capture. The request ID links the transaction to the customer’s billing and account information, so you are not required to include those fields in the credit request. However, when you combine a request for a refund with a request for another service, such as the tax calculation service, you must provide the customer’s billing and account information. Unless otherwise specified, refunds must be requested within 180 days of a settlement. You can request multiple refunds against a single capture. To perform multiple refunds, use the same request ID in each request.

Stand-Alone Credits

Stand-alone credits are not tied to an original transaction. Stand-alone credits do not have a time restriction, and they can be used to issue refunds more than 180 days after a transaction settlement.

Credit Workflow

The credit workflow begins when you send a request for a credit.
A credit does not happen in real time. All of the credit requests for a day are typically placed in a file and sent to the processor as a single batch transaction. In most cases, the batch transaction is settled overnight.

1. The merchant sends a request for a credit to Cybersource.
2. For online credits, Cybersource validates the order information then sends an online credit to the payment processor. For offline credits, Cybersource stores the credit request in a batch file and sends the batch file to the payment processor after midnight.
3. The processor validates the request and forwards it to the acquiring bank.
4. The acquiring bank transfers funds to the issuing bank.

Voids

A void cancels a capture or credit request that was submitted but not yet processed by the processor.
Capture and credit requests are usually submitted once a day. A void request is declined when the capture or credit request has already been sent to the processor.
After a void is processed, you cannot credit or capture the funds. You must perform a new transaction to capture or credit the funds. Further, when you void a capture, a hold remains on the authorized funds. If you are not going to re-capture the authorization, and if your processor supports authorization reversal after void (ARAV), you should request an authorization reversal to release the hold on the unused funds.
A void uses the capture or credit request ID to link the transactions. The authorization request ID is used to look up the customer's billing and account information, so there is no need to include those fields in the void request. You cannot perform a follow-on credit against a capture that has been voided.

Payment Features

You can apply features to different payment services to enhance the customer payment processing experience. This section includes an overview of these features:

- *Debit and Prepaid Card Payments* on page 23
- *Payer Authentication* on page 23
- *Processing Payments Using Credentials* on page 97
- *Token Management Service* on page 25

Debit and Prepaid Card Payments

Debit cards are linked to a cardholder's checking account. A merchant who accepts the debit card can deduct funds directly from the linked cardholder's account.
You can process debit cards using these services:

- Credit card services
- PIN debit services

Related Information

- See *Standard Payment Processing* on page 29 for information that shows you how to use credit card services.
- See *Debit and Prepaid Card Processing* on page 77 for information that shows you how to process authorizations that use a debit or prepaid card.

Payer Authentication

Payer authentication is run before a transaction is submitted for authorization. Most of the time payer authentication is bundled with authorization so that after payer authentication happens, the transaction is automatically submitted for authorization. Payer authentication and authorization can be configured to occur as separate operations. This section shows you how to run payer authentication as a separate process and pass the payer authentication data when seeking authorization for a transaction.
Payer authentication consists of a two-step verification process that adds an extra layer of fraud protection during the payment process. During transactions, the transaction device, location, past purchasing habits, and other factors are analyzed for indications of fraud. This process collects customer data during the transaction from at least two of these three categories:

- Something you have: A payment card or a payment card number
- Something you know: A password or pin
- Something you are: Facial recognition or fingerprint

Each of these payment card companies has its own payer authentication product:

- Discover: ProtectBuy
- JCB: J/Secure
- Mastercard: Identity Check
- Visa: Visa Secure

Payer authentication can be used to satisfy the Strong Customer Authentication (SCA) requirement of the Payment Services Directive (PSD2). SCA applies to the European Economic Area (EEA) and the United Kingdom. SCA requires banks to perform additional checks when customers make payments to confirm their identity.

Related Information

- See the *Payer Authentication Developer Guide* for more information about payer authentication.
- See *Payer Authentication Processing* on page 87 for information about how to process payments with payer authentication.

Introduction to Credentialed Transactions

Credentialed transactions are transactions that involve either storing a customer's payment credentials for future transactions or using a customer's already stored payment credentials. When processing a credentialed transaction, you must indicate the type of credentialed transaction and the reason for the transaction. Credentialed transactions are also known as credential-on-file (COF) transactions.
There are several types of credentialed transactions:

- Customer-Initiated Transactions (CITs): Any transaction a customer is actively participating in such as making a card-present payment, completing an online checkout, or by using a stored credential. CIT transactions can store the customer's credentials in your system for future CITs or merchant-initiated transactions.
- Merchant-Initiated Transactions (MITs): Any transaction a merchant initiates without the customer's participation such as an industry practice transaction or a standing instruction transaction.
    - Industry Practice Transactions: MITs that are performed as subsequent transactions to a CIT because the initial transaction could not be completed in one transaction. Not every industry practice transaction involves a stored credential. If a stored credential is used only for one transaction, that transaction is not considered a credentialed transaction.
    - Standing Instruction Transactions: MITs that are performed to follow agreed-upon instructions from the customer for the provision of goods and services.

Supported Services

These are the supported merchant-initiated services:

- Delayed Authorization
- Mastercard Standing Order Transactions
- Mastercard Subscription Transactions
- No-Show Transactions
- Reauthorization
- Recurring Transactions
- Unscheduled Credentials-on-File Transactions

The service determines the reason for the credentialed transaction.

Token Management Service

The Token Management Service (TMS) enables you to replace personally identifiable information (PII), such as the primary account numbers (PANs), with unique tokens. These tokens do not include the PII data, but act as a placeholder for the personal information that would otherwise need to be shared. By using tokens, businesses can provide a secure payment experience, reduce the risk of fraud, and comply with industry consumer security regulations such as PCI-DSS.
TMS links tokens across service providers, payment types, and channels for sellers, acquirers, and technology partners. TMS tokenizes, securely stores, and manages the primary account number (PAN), the payment card expiration date, electronic check details, and customer data. TMS also enables you to create a network token of a customer's payment card.

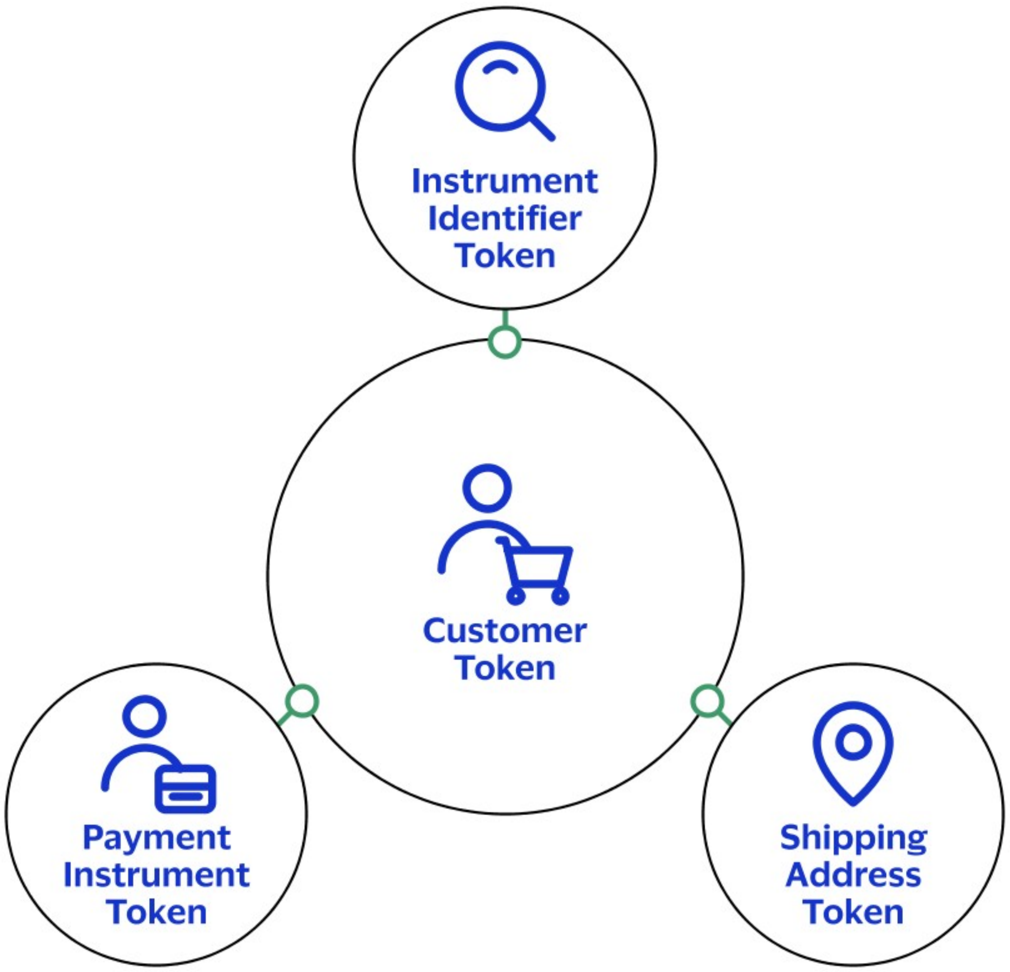
Important

Due to mandates from the Reserve Bank of India, Indian merchants cannot store PANs. Use network tokenization instead.

You can manage sensitive data securely by creating, retrieving, updating, and deleting tokens through the *TMS API*.
TMS simplifies your PCI DSS compliance. TMS passes tokens back to you that represent this data. You then store these tokens in your environment and databases instead of storing customer payment details.
TMS protects sensitive payment information through tokenization and secures and manages customer data using these token types:

- Customer tokens
- Instrument identifier tokens
- Payment instrument tokens
- Shipping address tokens

These TMS tokens can be used individually, or they can be associated with one customer token:



TMS Token Types

Related Information

- See the *Token Management Service Developer Guide* for more information about the TMS.
- See *Token Management Service Processing* on page 202 for information that shows you how to process payments using the TMS.

Testing the Payment Services

To ensure that requests are processed correctly, you must test the basic success and error conditions for each service you plan to use.

Requirements for Testing

Important

Before you can test, you must contact customer support to activate the credit card services and configure your account for testing. You must also contact your processor to set up your processor account.

Important

When building your connection to the Cybersource payment gateway, ensure that you have implemented controls to prevent card testing or card enumeration attacks on your platform. For more information, see the *best practices guide*. When we detect suspicious transaction activity associated with your merchant ID, including a card testing or card enumeration attack, Cybersource reserves the right to enable fraud management tools on your behalf in order to mitigate the attack. The fraud team might also implement internal controls to mitigate attack activity. These controls block traffic that is perceived as fraudulent. Additionally, if you are using one of our fraud tools and experience a significant attack, our internal team might modify or add rules to your configuration to help prevent the attack and minimize the threat to our infrastructure. However, any actions taken by Cybersource would not replace the need for you to follow industry standard best practices to protect your systems, servers, and platforms.

Follow these requirements when you test your system:

- Use your regular merchant ID.
- Use a real combination for the city, state, and postal code.
- Use a real combination for the area code and telephone number.
- Use a nonexistent account and domain name for the customer's email address.
- REST API test endpoint: `POST https://apitest.cybersource.com/pts/v2/payments`

Test Card Numbers

Use these payment card numbers to test the authorization, capture, and credit services. Remove the spaces from the test card numbers when sending them to the test system. Do not use real payment card numbers. To test card types that are not included in the list, use an account number that is in the card's BIN range. For best results, try each test with a different service request and with different test payment card numbers.

- American Express—3782 8224 6310 005
- Discover—6011 1111 1111 1117
- JCB—3566 1111 1111 1113
- Maestro (International)
  - 5033 9619 8909 17
  - 5868 2416 0825 5333 38

- Maestro (UK Domestic)—the issue number is not required for Maestro (UK Domestic) transactions.
    - 6759 4111 0000 0008
    - 6759 5600 4500 5727 054
    - 5641 8211 1116 6669
- Mastercard
    - 2222 4200 0000 1113
    - 2222 6300 0000 1125
    - 5555 5555 5555 4444
- UATP—1354 1234 5678 911
- Visa—4111 1111 1111 1111

Using Amounts to Simulate Errors

You can simulate error messages by requesting authorization, capture, or credit services with specific amounts that trigger the error messages. These triggers work only on the test server, not on the production server.
Each payment processor uses its own error messages. For more information, see: *REST API Testing Guide* .

Test American Express Card Verification

Before using CVN with American Express, it is strongly recommended that you follow these steps:

1. Contact customer support to have your account configured for CVN. Until you do this, you will receive a `1` in the **processorInformation.cardVerification.resultCode** response field.
2. Test your system in production using a small currency amount, such as one currency unit. Instead of using the test account numbers, use a real payment card account number, and send an incorrect CVN in the request for authorization. The card should be refused and the request declined.

Standard Payment Processing

This section shows you how to process various authorization, capture, credit, and sales transactions.

Basic Authorizations

This section provides the information you need in order to process a basic authorization.

Endpoint

Production: `POST https://api.cybersource.com/pts/v2/payments`
Test: `POST https://apitest.cybersource.com/pts/v2/payments`

Declined Authorizations

If an authorization is declined, you can use response categories to help you decide whether to retry or block a declined transaction. These response fields provide additional information:

- **paymentInsightsInformation.responseInsights.category**
- **paymentInsightsInformation.responseInsights.categoryCode**

Category codes have possible values (such as `01`) each of which corresponds to a category that contains a description.
You cannot retry this category code and category:

- `01 ISSUER_WILL_NEVER_APPROVE`

For these values, you can retry the transaction a maximum of 15 times over a period of 30 days:

- `02 ISSUER_CANNOT_APPROVE_AT_THIS_TIME`

- `03 ISSUER_CANNOT_APPROVE_WITH_THESE_DETAILS`: Data quality issue. Revalidate data prior to retrying the transaction.
- `04 GENERIC_ERROR`
- `97 PAYMENT_INSIGHTS_INTERNAL_ERROR`
- `98 OTHERS`
- `99 PAYMENT_INSIGHTS_RESPONSE_CATEGORY_MATCH_NOT_FOUND`

Required Fields for Processing a Basic Authorization

Use these required fields for processing a basic authorization.

orderInformation.amountDetails.currency

orderInformation.amountDetails.totalAmount

orderInformation.billTo.address1

orderInformation.billTo.administrativeArea

orderInformation.billTo.country

orderInformation.billTo.email

orderInformation.billTo.firstName

orderInformation.billTo.lastName

orderInformation.billTo.locality

orderInformation.billTo.postalCode

paymentInformation.card.expirationMonth

paymentInformation.card.expirationYear

paymentInformation.card.number

Related Information

- *API field reference guide for the REST API*

REST Interactive Example: Processing a Basic Authorization

```
Simple Authorization(Internet)
```

Live Console URL: *https://developer.cybersource.com/api-reference-assets/index.html#payments_payments_process-a-payment*

REST Example: Processing a Basic Authorization

Request

```
{
  "orderInformation": {
    "billTo": {
            "country": "US",
            "lastName": "Kim",
            "address1": "201 S. Division St.",
            "postalCode": "48104-2201",
            "locality": "Ann Arbor",
            "administrativeArea": "MI",
            "firstName": "Kyong-Jin",
            "email": "test@cybs.com"
        },
        "amountDetails": {
            "totalAmount": "100.00",
            "currency": "usd"
        }
    },
    "paymentInformation": {
        "card": {
            "expirationYear": "2031",
            "number": "4111111111111111",
            "expirationMonth": "12",
            "type": "001"
        }
    }
}
```

Response to a Successful Request

```
{
  "_links" : {
    "authReversal" : {
      "method" : "POST",
      "href" : "/pts/v2/payments/6461731521426399003473/reversals"
    },
    "self" : {
      "method" : "GET",
      "href" : "/pts/v2/payments/6461731521426399003473"
    },
    "capture" : {
      "method" : "POST",
      "href" : "/pts/v2/payments/6461731521426399003473/captures"
    }
  },
  "clientReferenceInformation" : {
    "code" : "1646173152047"
  },
  "id" : "6461731521426399003473",
  "orderInformation" : {
    "amountDetails" : {
      "authorizedAmount" : "100.00",
      "currency" : "usd"
    }
  },
  "paymentAccountInformation" : {
    "card" : {
      "type" : "001"
    }
  },
  "paymentInformation" : {
    "tokenizedCard" : {
      "type" : "001"
    },
    "card" : {
      "type" : "001"
    }
  },
 "paymentInsightsInformation" : {
    "responseInsights" : {
      "categoryCode" : "01"
    }
  },
  "processorInformation" : {
    "systemTraceAuditNumber" : "862481",
    "approvalCode" : "831000",
    "merchantAdvice" : {
      "code" : "01",
      "codeRaw" : "M001"
    },
    "responseDetails" : "ABC",
    "networkTransactionId" : "016153570198200",
    "consumerAuthenticationResponse" : {
      "code" : "2",
      "codeRaw" : "2"
    },
    "transactionId" : "016153570198200",
    "responseCode" : "00",
    "avs" : {
      "code" : "Y",
      "codeRaw" : "Y"
    }
  },
  "reconciliationId" : "64617315214263990003473",
  "status" : "AUTHORIZED",
  "submitTimeUtc" : "2022-03-01T22:19:12Z"
}
```

Response to a Declined Request

```
{
  "clientReferenceInformation": {
    "code": "TC50171_3"
  },
  "errorInformation": {
    "reason": "PROCESSOR_ERROR",
    "message": "Invalid account"
  },
  "id": "6583553837826789303954",
  "paymentInsightsInformation": {
    "responseInsights": {
      "categoryCode": "01",
      "category": "ISSUER_WILL_NEVER_APPROVE"
    }
  },
  "pointOfSaleInformation": {
    "amexCapnData": "1009S0600100"
  },
  "processorInformation": {
    "systemTraceAuditNumber": "004544",
    "merchantNumber": "1231231222",
    "networkTransactionId": "431736869536459",
    "transactionId": "431736869536459",
   "responseCode": "111",
    "avs": {
      "code": "Y",
      "codeRaw": "Y"
    }
  },
  "status": "DECLINED"
}
```

Authorizations with Line Items

This section shows you how to process an authorization with line items.

The main difference between a basic authorization and an authorization that includes line items is that the **orderInformation.amountDetails.totalAmount** field, which is included in a basic authorization, is substituted with one or more line items that are included in a **lineItem[]** array.

Fields Specific to this Use Case

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

orderInformation.lineItems[].unitPrice	
orderInformation.lineItems[].quantity	
orderInformation.lineItems[].productCode	
orderInformation.lineItems[].productSku	Optional when **item_#_productCode** is set to `default`, `shipping_only`, `handling_only`, or `shipping_and_handling`
orderInformation.lineItems[].productName	Optional when **item_#_productCode** is set to `default`, `shipping_only`, `handling_only`, or `shipping_and_handling`

At a minimum, you must include the **orderInformation.lineItems[].unitPrice** field in order to include a line item in an authorization. When this field is the only field included in the authorization, the system sets:

- **orderInformation.lineItems[].productCode**: `default`
- **orderInformation.lineItems[].quantity**: `1`

For example, these three line items are valid.

```
"orderInformation": {
 "lineItems": [
  {
   "unitPrice": "10.00"
  },
  {
   "unitPrice": "5.99",
   "quantity": "3",
   "productCode": "shipping_only"
  },
  {
   "unitPrice": "29.99",
   "quantity": "3",
   "productCode": "electronic_good",
   "productSku": "12384569",
   "productName": "receiver"
  }
 ]
}
```

Endpoint

Production: `POST https://api.cybersource.com/pts/v2/payments`
Test: `POST https://apitest.cybersource.com/pts/v2/payments`

Required Fields for Processing an Authorization with Line Items

Use these required fields for processing an authorization that includes line items.

orderInformation.amountDetails.currency

orderInformation.billTo.address1

orderInformation.billTo.administrativeArea

orderInformation.billTo.country

orderInformation.billTo.email

orderInformation.billTo.firstName

orderInformation.billTo.lastName

orderInformation.billTo.locality

orderInformation.billTo.postalCode

paymentInformation.card.expirationMonth

paymentInformation.card.expirationYear

paymentInformation.card.number

Related Information

- *API field reference guide for the REST API*

REST Example: Processing an Authorization with Line Items

Request

```
{
  "currencyConversion": {
    "indicator": "Y"
  },
  "paymentInformation": {
    "card": {
      "number": "4111111111111111",
      "expirationMonth": "12",
      "expirationYear": "2031"
    }
  },
  "orderInformation": {
    "amountDetails": {
      "currency": "USD",
      "exchangeRate": ".91",
      "originalAmount": "107.33",
      "originalCurrency": "eur"
    },
    "billTo": {
      "firstName": "John",
      "lastName": "Doe",
      "address1": "1 Market St",
      "locality": "san francisco",
      "administrativeArea": "CA",
      "postalCode": "94105",
      "country": "US",
      "email": "test@cybs.com"
    },
    "lineItems": [
      {
        "unitPrice": "10.00"
      },
      {
        "unitPrice": "5.99",
        "quantity": "3",
        "productCode": "shipping_only"
      },
      {
        "unitPrice": "29.99",
        "quantity": "3",
        "productCode": "electronic_good",
        "productSku": "12384569",
        "productName": "receiver"
      }
    ]
  }
}
```

Response to a Successful Request

```
{
 "_links": {
  "authReversal": {
   "method": "POST",
   "href": "/pts/v2/payments/6482385519226028804003/reversals"
  },
  "self": {
   "method": "GET",
   "href": "/pts/v2/payments/6482385519226028804003"
  },
  "capture": {
   "method": "POST",
   "href": "/pts/v2/payments/6482385519226028804003/captures"
  }
 },
 "clientReferenceInformation": {
  "code": "1648238551902"
 },
 "id": "6482385519226028804003",
 "orderInformation": {
  "amountDetails": {
   "authorizedAmount": "117.94",
   "currency": "USD"
  }
 },
 "paymentAccountInformation": {
  "card": {
   "type": "001"
  }
 },
 "paymentInformation": {
  "tokenizedCard": {
   "type": "001"
  },
  "card": {
   "type": "001"
  }
 },
 "processorInformation": {
  "systemTraceAuditNumber": "191521",
  "approvalCode": "831000",
  "merchantAdvice": {
   "code": "01",
   "codeRaw": "M001"
  },
  "responseDetails": "ABC",
  "networkTransactionId": "016153570198200",
  "consumerAuthenticationResponse": {
   "code": "2",
   "codeRaw": "2"
  },
  "transactionId": "016153570198200",
  "responseCode": "00",
  "avs": {
    "code": "Y",
    "codeRaw": "Y"
   }
  },
  "reconciliationId": "6482385519226028804003",
  "status": "AUTHORIZED",
  "submitTimeUtc": "2022-03-25T20:02:32Z"
}
```

Authorizations with Payment Network Tokens

This section shows you how to successfully process an authorization with payment network tokens.

Important

Due to mandates from the Reserve Bank of India, Indian merchants cannot store personal account numbers (PAN). Use network tokens instead. For more information on network tokens, see *Network Tokenization* in the Token Management Service Developer Guide.

Endpoint

Production: `POST https://api.cybersource.com/pts/v2/payments`
Test: `POST https://apitest.cybersource.com/pts/v2/payments`

Required Fields for Authorizations with Payment Network Tokens

Use these required fields for processing an authorization with payment network tokens.

orderInformation.amountDetails.currency

orderInformation.amountDetails.totalAmount

orderInformation.billTo.address1

orderInformation.billTo.email

orderInformation.billTo.firstName

orderInformation.billTo.lastName

paymentinformation.tokenizedCard.cryptogram

paymentinformation.tokenizedCard.expirationMonth

paymentinformation.tokenizedCard.expirationYear

Related Information

- *API field reference guide for the REST API*

Optional Fields for Authorizations with Payment Network Tokens

You can use these optional fields to include additional information when processing an authorization with a payment network token.

clientReferenceInformation.code	
consumerAuthenticationInformation.cavv	**For 3-D Secure in-app transactions for Visa and JCB, set this field to the 3-D Secure cryptogram. Otherwise, set to the network token cryptogram.**
consumerAuthenticationInformation.ucafAuthenticationData	**For Mastercard requests using 3-D Secure, set this field to the Identity Check cryptogram.**
consumerAuthenticationInformation.ucafCollectionIndicator	**For Mastercard requests using 3-D Secure, set the value to 2.**
orderInformation.amountDetails.currency	
orderInformation.amountDetails.totalAmount	
orderInformation.billTo.address1	
orderInformation.billTo.country	
orderInformation.billTo.email	
orderInformation.billTo.firstName	
orderInformation.billTo.lastName	
orderInformation.billTo.locality	
orderInformation.billTo.postalCode	**Required only for transactions in the US and Canada.**
orderInformation.billTo.administrativeArea	**Required only for transactions in the US and Canada.**
processingInformation.commerceIndicator	
paymentInformation.tokenizedCard.cardType	**It is strongly recommended that you send the card type even if it is optional for your processor. Omitting the card type can cause the transaction to be processed with the wrong card type.**
paymentInformation.tokenizedCard.cryptogram	
paymentInformation.tokenizedCard.expirationMonth	**Set to the token expiration month that you received from the token service provider.**
paymentInformation.tokenizedCard.expirationYear	**Set to the token expiration year that you received from the token service provider.**

paymentInformation.tokenizedCard.number	**Set to the token value that you received from the token service provider.**
paymentInformation.tokenizedCard.requestorId	
paymentInformation.tokenizedCard.transactionType	

Related Information

- *API field reference guide for the REST API*

REST Example: Authorizations with Payment Network Tokens

Request

```
{
  "orderInformation" : {
   "billTo": {
      "country": "US",
      "lastName": "Kim",
      "address1": "201 S. Division St.",
      "postalCode": "48104-2201",
      "locality": "Ann Arbor",
      "administrativeArea": "MI",
      "firstName": "Kyong-Jin",
      "email": "test@cybs.com"
      },
   "amountDetails" : {
     "totalAmount" : "100",
     "currency" : "USD"
   }
  },
   "paymentInformation" : {
   "tokenizedCard" : {
     "expirationYear" : "2031",
     "number" : "4111111111111111",
     "expirationMonth" : "12",
     "transactionType" : "1",
     "cryptogram" : "qE5juRwDzAUFBAkEHuWW9PiBkWv="
   }
  }
}
```

Response to a Successful Request

```
{
   "_links": {
      "authReversal": {
         "method": "POST",
         "href": "/pts/v2/payments/6838294805206235603954/reversals"
      },
      "self": {
         "method": "GET",
         "href": "/pts/v2/payments/6838294805206235603954"
      },
      "capture": {
         "method": "POST",
         "href": "/pts/v2/payments/6838294805206235603954/captures"
      }
   },
   "clientReferenceInformation": {
      "code": "1683829480593"
   },
   "id": "6838294805206235603954",
   "orderInformation": {
      "amountDetails": {
         "authorizedAmount": "100.00",
         "currency": "USD"
      }
   },
   "paymentAccountInformation": {
      "card": {
         "type": "001"
      }
   },
   "paymentInformation": {
      "tokenizedCard": {
         "type": "001"
      },
      "card": {
         "type": "001"
      }
   },
   "pointOfSaleInformation": {
      "terminalId": "111111"
   },
   "processorInformation": {
      "approvalCode": "888888",
      "networkTransactionId": "123456789619999",
      "transactionId": "123456789619999",
      "responseCode": "100",
      "avs": {
         "code": "1"
      }
   },
   "reconciliationId": "60332034UHI9PRJ0",
   "status": "AUTHORIZED",
   "submitTimeUtc": "2023-05-11T18:24:40Z"
}
```

Authorizations with a Card Verification Number

This section shows you how to process an authorization with a Card Verification Number (CVN).

CVN Results

The response includes a raw response code and a mapped response code:

- The raw response code is the value returned by the processor. This value is returned in the **processorInformation.cardVerification.resultCodeRaw** field. Use this value only for debugging purposes; do not use it to determine the card verification response.
- The mapped response code is the pre-defined value that corresponds to the raw response code. This value is returned in the **processorInformation.cardVerification.resultCode** field.

Even when the CVN does not match the expected value, the issuing bank might still authorize the transaction. You will receive a CVN decline, but you can still capture the transaction because it has been authorized by the bank. However, you must review the order to ensure that it is legitimate.
Settling authorizations that fail the CVN check might have an impact on the fees charged by your bank. Contact your bank for details about how card verification management might affect your discount rate.
When a CVN decline is received for the authorization in a sale request, the capture request is not processed unless you set the **processingInformation.authorizationOptions.ignoreCvResult** field to `true`.

CVN Results for American Express	A value of `1` in the **processorInformation.cardVerification.resultCode** field indicates that your account is not configured to use card verification. Contact customer support to have your account enabled for this feature.
CVN Results for Discover	
CVN Results for Visa and Mastercard	A CVN code of `D` or `N` causes the request to be declined with a reason code value of `230`. You can still capture the transaction, but you must review the order to ensure that it is legitimate. Cybersource, not the issuer, assigns the CVN decline to the authorization. You can capture any authorization that has a valid authorization code from the issuer, even when the request receives a CVN decline. When the issuer does not authorize the transaction and the CVN does not match, the request is declined because the card is refused. You cannot capture the transaction.

Fields Specific to this Use Case

Include this field with a standard authorization request when processing an authorization with a CVN:

- **paymentInformation.card.securityCode**

Endpoint

Production: POST https://api.cybersource.com/pts/v2/payments
Test: POST https://apitest.cybersource.com/pts/v2/payments

Required Fields for Processing an Authorization with a Card Verification Number

Use these required fields for processing an authorization that includes a Card Verification Number (CVN).

orderInformation.amountDetails.currency

orderInformation.amountDetails.totalAmount

orderInformation.billTo.address1

orderInformation.billTo.administrativeArea

orderInformation.billTo.country

orderInformation.billTo.email

orderInformation.billTo.firstName

orderInformation.billTo.lastName

orderInformation.billTo.locality

orderInformation.billTo.postalCode

paymentInformation.card.expirationMonth

paymentInformation.card.expirationYear

paymentInformation.card.number

paymentInformation.card.securityCode

paymentInformation.card.type

paymentInformation.card.securityCode

Related Information

- *API field reference guide for the REST API*

Optional Fields for Processing an Authorization with a Card Verification Number

You can use these optional fields to include additional information when processing an authorization with a card verification number.

paymentInformation.card.securityCodeIndicator

processingInformation.authorizationOptions.ignoreCvResult

REST Example: Processing an Authorization with a Card Verification Number

Request

```
{
   "paymentInformation": {
      "card": {
      "number": "4111111111111111",
      "expirationMonth": "12",
      "expirationYear": "2031",
      "type": "001",
      "securityCode": "999"
      }
   },
   "orderInformation": {
      "amountDetails": {
      "totalAmount": "49.95",
      "currency": "USD"
   },
    "billTo": {
      "firstName": "John",
      "lastName": "Doe",
      "address1": "1295 Charleston Rd.",
      "locality": "Mountain View",
      "administrativeArea": "CA",
      "postalCode": "94043",
      "country": "US",
      "email": "jdoe@example.com",
      "phoneNumber": "650-965-6000"
      }
   }
}
```

Response to a Successful Request

```
{
   "_links": {
      "authReversal": {
         "method": "POST",
         "href": "/pts/v2/payments/6554147587216874903954/reversals"
      },
      "self": {
         "method": "GET",
            "href": "/pts/v2/payments/6554147587216874903954"
        },
        "capture": {
            "method": "POST",
            "href": "/pts/v2/payments/6554147587216874903954/captures"
        }
    },
    "clientReferenceInformation": {
        "code": "1655414758839"
    },
    "id": "6554147587216874903954",
    "orderInformation": {
        "amountDetails": {
            "authorizedAmount": "49.95",
            "currency": "USD"
        }
    },
    "paymentAccountInformation": {
        "card": {
            "type": "001"
        }
    },
    "paymentInformation": {
        "tokenizedCard": {
            "type": "001"
        },
        "card": {
            "type": "001"
        }
    },
    "pointOfSaleInformation": {
        "terminalId": "111111"
    },
    "processorInformation": {
        "approvalCode": "888888",
        "networkTransactionId": "123456789619999",
        "transactionId": "123456789619999",
        "responseCode": "100",
        "avs": {
            "code": "X",
            "codeRaw": "I1"
        }
    },
    "reconciliationId": "67546603C43Z6JWN",
    "status": "AUTHORIZED",
    "submitTimeUtc": "2022-06-16T21:25:58Z"
}
```

Authorizations with Strong Customer Authentication Exemption

This section shows you how to process an authorization with a strong customer authentication (SCA) exemption.
You can use SCA exemptions to streamline the payment process. SCA exemptions are part of the European second Payment Services Directive (PSD2) and allow certain types of low-risk transactions to bypass additional authentication steps while still remaining compliant with PSD2. You can choose which exemption can be applied to a transaction, but the card-issuing bank actually grants an SCA exemption during card authentication.
You can process an authorization with two types of SCA exemptions:

- Exemption on Authorization: Send an authorization without payer authentication and request an SCA exemption on the authorization. If it is not approved, you may be required to request further authentication upon retry.
- Exemption on Authentication: Request an SCA exemption during payer authentication and if successful, send an authorization including the SCA exemption details.

Depending on your processor, use one of these exemption fields:

Important

If you send more than one SCA exemption field with a single authentication, the transaction is denied.

- Authentication Outage: Payer authentication is not available for this transaction due to a system outage.
- B2B Corporate Card: Payment cards specifically for business-to-business transactions are exempt.
- Delegated Authentication: Payer authentication was performed outside of the authorization workflow.
- Follow-On Installment Payment: Installment payments of a fixed amount are exempt after the first transaction.
- Follow-On Recurring Payment: Recurring payments of a fixed amount are exempt after the first transaction.
- Low Risk: The average fraud levels associated with this transaction are considered low.
- Low Value: The transaction value does not warrant SCA.
- Merchant Initiated Transactions: As follow-on transactions, merchant-initiated transactions are exempt.
- Stored Credential Transaction: Credentials are authenticated before storing, so stored credential transactions are exempt.
- Trusted Merchant: Merchants registered as trusted beneficiaries.

Exemption Fields Specific to the Strong Customer Authentication Use Case

Use one of these fields to request an SCA exemption:

Types of SCA Exemptions

Exemption Type	Field	Value
Authentication outage	consumerAuthenticationInformation. strongAuthentication. authenticationOutageExemptionIndicator	1
Follow-on recurring payment	processingInformation.commerceIndicator	recurring

Processor Support for SCA Exemptions

You can send an authorization without payer authentication and request an SCA exemption on the authorization. If it is not approved, you may be required to request further authentication upon retry. Use this table to determine which processors support SCA exemptions on authorization:

Processor Support for SCA Exemption on Authorization

	Authentication Outage	Follow-On Recurring Payment	Low Value	Transaction Risk Analysis
LloydsTSB Cardnet				

You can request an SCA exemption during payer authentication and if successful, send an authorization including the SCA exemption details. Use this table to determine which processors support SCA exemptions on authentication:

Processor Support for SCA Exemption on Authentication

	B2B Corporate Card	Delegated Authentication	Low Risk	Low Value	Trusted Merchant
LloydsTSB Cardnet					

For more information, see the *Exemption Test Cases* section of the Payer Authentication Developer Guide.

Endpoint

Production: `POST https://api.cybersource.com/pts/v2/payments`
Test: `POST https://apitest.cybersource.com/pts/v2/payments`

Required Fields for Processing an Authorization with an SCA Exemption

Use these required fields for processing an authorization that includes an SCA exemption.

orderInformation.amountDetails.currency

orderInformation.amountDetails.totalAmount

orderInformation.billTo.address1

orderInformation.billTo.administrativeArea

orderInformation.billTo.country

orderInformation.billTo.email

orderInformation.billTo.firstName

orderInformation.billTo.lastName

orderInformation.billTo.locality

orderInformation.billTo.postalCode

paymentInformation.card.expirationMonth

paymentInformation.card.expirationYear

paymentInformation.card.number

paymentInformation.card.type

Related Information

- *API field reference guide for the REST API*

REST Example: Processing an Authorization with an SCA Exemption for Low-Value Transactions

Request

```
{
  "consumerAutenticationInformation" : {
   "strongAuthentication" : {
     "lowValueExemptionIndicator" : "1"
   }
  },
  "orderInformation" : {
    "billTo" : {
      "country" : "US",
      "lastName" : "Kim",
      "address1" : "201 S. Division St.",
      "postalCode" : "48104-2201",
      "locality" : "Ann Arbor",
      "administrativeArea" : "MI",
      "firstName" : "Kyong-Jin",
      "email" : "test@cybs.com"
    },
    "amountDetails" : {
      "totalAmount" : "100.00",
      "currency" : "eur"
    }
  },
  "paymentInformation" : {
    "card" : {
      "expirationYear" : "2031",
      "number" : "4111111111111111",
      "expirationMonth" : "12"
    }
  }
}
```

Response to a Successful Request

```
{
    "_links": {
        "authReversal": {
            "method": "POST",
            "href": "/pts/v2/payments/6709780221406171803955/reversals"
        },
        "self": {
            "method": "GET",
            "href": "/pts/v2/payments/6709780221406171803955"
        },
        "capture": {
            "method": "POST",
            "href": "/pts/v2/payments/6709780221406171803955/captures"
        }
    },
    "clientReferenceInformation": {
        "code": "1670978022258"
    },
    "id": "6709780221406171803955",
    "orderInformation": {
        "amountDetails": {
            "authorizedAmount": "100.00",
            "currency": "eur"
        }
    },
    "paymentAccountInformation": {
        "card": {
            "type": "001"
        }
    },
    "paymentInformation": {
        "tokenizedCard": {
            "type": "001"
        },
        "card": {
            "type": "001"
        }
    },
  "pointOfSaleInformation": {
    "terminalId": "123456"
  },
  "processorInformation": {
    "approvalCode": "888888",
    "networkTransactionId": "123456789619999",
    "transactionId": "123456789619999",
    "responseCode": "100",
    "avs": {
      "code": "X",
      "codeRaw": "I1"
    }
  },
  "reconciliationId": "62859554PBDEMI43",
  "status": "AUTHORIZED",
  "submitTimeUtc": "2022-12-14T00:33:42Z"
}
```

Zero Amount Authorizations

This section provides the information that you need in order to process a zero amount authorization.
Authorizing a payment for a zero amount shows whether a payment card account is valid and whether the card is lost or stolen. You cannot capture a zero amount authorization.

Processor-Specific Information

LloydsTSB Cardnet	**AVS and CVN are supported.** **Card types: Mastercard, Visa** **The commerce indicator must be** `internet` **or** `moto`**.**

Endpoint

Production: `POST https://api.cybersource.com/pts/v2/payments`
Test: `POST https://apitest.cybersource.com/pts/v2/payments`

Required Fields for Processing a Zero Amount Authorization

Use these required fields for processing a zero amount authorization.

orderInformation.amountDetails.currency	
orderInformation.amountDetails.totalAmount	**Set this value to** `0`**.**
orderInformation.billTo.address1	
orderInformation.billTo.administrativeArea	
orderInformation.billTo.country	

orderInformation.billTo.email

orderInformation.billTo.firstName

orderInformation.billTo.lastName

orderInformation.billTo.locality

orderInformation.billTo.postalCode

paymentInformation.card.expirationMonth

paymentInformation.card.expirationYear

paymentInformation.card.number

Related Information

- *API field reference guide for the REST API*

REST Example: Processing a Zero Amount Authorization

Request

```
{
  "orderInformation" : {
    "billTo" : {
      "country" : "US",
      "lastName" : "Kim",
      "address1" : "201 S. Division St.",
      "postalCode" : "48104-2201",
      "locality" : "Ann Arbor",
      "administrativeArea" : "MI",
      "firstName" : "Kyong-Jin",
      "email" : "test@cybs.com"
    },
    "amountDetails" : {
      "totalAmount" : "0.00",
      "currency" : "usd"
    }
  },
  "paymentInformation" : {
    "card" : {
      "expirationYear" : "2031",
      "number" : "4111111111111111",
      "expirationMonth" : "12"
    }
  }
}
```

Response to a Successful Request

```
{
  "_links" : {
    "authReversal" : {
      "method" : "POST",
      "href" : "/pts/v2/payments/6461731521426399003473/reversals"
    },
    "self" : {
      "method" : "GET",
      "href" : "/pts/v2/payments/6461731521426399003473"
    },
    "capture" : {
      "method" : "POST",
      "href" : "/pts/v2/payments/6461731521426399003473/captures"
    }
  },
  "clientReferenceInformation" : {
    "code" : "1646173152047"
  },
  "id" : "6461731521426399003473",
  "orderInformation" : {
    "amountDetails" : {
      "authorizedAmount" : "0",
      "currency" : "usd"
    }
  },
  "paymentAccountInformation" : {
    "card" : {
      "type" : "001"
    }
  },
  "paymentInformation" : {
    "tokenizedCard" : {
      "type" : "001"
    },
    "card" : {
      "type" : "001"
    }
  },
  "processorInformation" : {
    "systemTraceAuditNumber" : "862481",
    "approvalCode" : "831000",
    "merchantAdvice" : {
      "code" : "01",
      "codeRaw" : "M001"
    },
    "responseDetails" : "ABC",
    "networkTransactionId" : "016153570198200",
    "consumerAuthenticationResponse" : {
      "code" : "2",
      "codeRaw" : "2"
    },
    "transactionId" : "016153570198200",
    "responseCode" : "00",
    "avs" : {
      "code" : "Y",
      "codeRaw" : "Y"
    }
  },
  "reconciliationId" : "6461731521426399003473",
  "status" : "AUTHORIZED",
  "submitTimeUtc" : "2022-03-01T22:19:12Z"
```

```
}
```

Pre-Authorizations

This section provides the information you need in order to process a pre-authorization. A pre-authorization enables you to authorize a payment when the final amount is unknown. It is typically used for lodging, auto rental, e-commerce, and restaurant transactions.
For a pre-authorization:

- The authorization amount must be greater than zero.
- The authorization must be submitted for capture within 30 calendar days of its request.
- When you do not capture the authorization, you must reverse it.
  In the U.S., Canada, Latin America, and Asia Pacific, Mastercard charges an additional fee for a pre-authorization that is not captured and not reversed.
  In Europe, Russia, Middle East, and Africa, Mastercard charges fees for all pre-authorizations.
- Chargeback protection is in effect for 30 days after the authorization.

Endpoint

Production: `POST https://api.cybersource.com/pts/v2/payments`
Test: `POST https://apitest.cybersource.com/pts/v2/payments`

Required Fields for a Pre-Authorization

Use these required fields for processing a pre-authorization.

orderInformation.amountDetails.currency

orderInformation.amountDetails.totalAmount

orderInformation.billTo.address1

orderInformation.billTo.administrativeArea

orderInformation.billTo.country

orderInformation.billTo.email

orderInformation.billTo.firstName

orderInformation.billTo.lastName

orderInformation.billTo.locality

orderInformation.billTo.postalCode

paymentInformation.card.expirationMonth

paymentInformation.card.expirationYear

paymentInformation.card.number

Related Information

- *API field reference guide for the REST API*

REST Example: Processing a Pre-Authorization

Request

```
{
   "orderInformation": {
      "billTo": {
         "country": "US",
         "lastName": "Kim",
         "address1": "201 S. Division St.",
         "postalCode": "48104-2201",
         "locality": "Ann Arbor",
         "administrativeArea": "MI",
         "firstName": "Kyong-Jin",
         "email": "test@cybs.com"
      },
      "amountDetails": {
         "totalAmount": "100.00",
         "currency": "usd"
      }
   },
   "paymentInformation": {
      "card": {
         "expirationYear": "2031",
         "number": "4111111111111111",
         "expirationMonth": "12",
         "type": "001"
      }
   }
}
```

Response to a Successful Request

```
{
  "_links" : {
    "authReversal" : {
      "method" : "POST",
      "href" : "/pts/v2/payments/6461731521426399003473/reversals"
    },
    "self" : {
      "method" : "GET",
      "href" : "/pts/v2/payments/6461731521426399003473"
    },
    "capture" : {
      "method" : "POST",
      "href" : "/pts/v2/payments/6461731521426399003473/captures"
    }
  },
  "clientReferenceInformation" : {
    "code" : "1646173152047"
  },
  "id" : "6461731521426399003473",
  "orderInformation" : {
    "amountDetails" : {
      "authorizedAmount" : "100.00",
      "currency" : "usd"
    }
  },
  "paymentAccountInformation" : {
    "card" : {
      "type" : "001"
    }
  },
  "paymentInformation" : {
    "tokenizedCard" : {
      "type" : "001"
    },
    "card" : {
      "type" : "001"
    }
  },
 "paymentInsightsInformation" : {
    "responseInsights" : {
      "categoryCode" : "01"
    }
  },
  "processorInformation" : {
    "systemTraceAuditNumber" : "862481",
    "approvalCode" : "831000",
    "merchantAdvice" : {
      "code" : "01",
      "codeRaw" : "M001"
    },
    "responseDetails" : "ABC",
    "networkTransactionId" : "016153570198200",
    "consumerAuthenticationResponse" : {
      "code" : "2",
      "codeRaw" : "2"
    },
    "transactionId" : "016153570198200",
    "responseCode" : "00",
    "avs" : {
      "code" : "Y",
      "codeRaw" : "Y"
    }
  },
  "reconciliationId" : "6461731521426399003473",
  "status" : "AUTHORIZED",
  "submitTimeUtc" : "2022-03-01T22:19:12Z"
}
```

Response to a Declined Request

```
{
  "clientReferenceInformation": {
    "code": "TC50171_3"
  },
  "errorInformation": {
    "reason": "PROCESSOR_ERROR",
    "message": "Invalid account"
  },
  "id": "6583553837826789303954",
  "paymentInsightsInformation": {
    "responseInsights": {
      "categoryCode": "01",
      "category": "ISSUER_WILL_NEVER_APPROVE"
    }
  },
  "pointOfSaleInformation": {
    "amexCapnData": "1009S0600100"
  },
  "processorInformation": {
    "systemTraceAuditNumber": "004544",
    "merchantNumber": "1231231222",
    "networkTransactionId": "431736869536459",
    "transactionId": "431736869536459",
   "responseCode": "111",
    "avs": {
      "code": "Y",
      "codeRaw": "Y"
    }
  },
  "status": "DECLINED"
}
```

Final Authorization Indicator

The purpose of this feature is to ensure that unused funds are reversed, so that customer's funds are available again when an order is not fulfilled.
For an authorization with an amount greater than zero, indicate whether the authorization is a final authorization, a pre-authorization, or an undefined authorization.
You can set a default authorization type in your account. To set the default authorization type in your account, contact customer support.
Chargeback protection is in effect for seven days after the authorization.

Supported Services

- Authorization
- Incremental authorization

Supported Card Types

- Maestro (International)
- Maestro (UK Domestic)
- Mastercard

Requirements for Final Authorizations

For a final authorization:

- The authorization amount must be greater than zero.
- The authorization amount must be the final amount that the customer agrees to pay.
- The authorization should not be cancelled after it is approved except when a system failure occurs.
- The authorization must be submitted for capture within seven calendar days of its request.
- The capture amount and currency must be the same as the authorization amount and currency.

Pre-Authorizations

A pre-authorization enables you to authorize a payment when the final amount is unknown. It is typically used for lodging, auto rental, e-commerce, and restaurant transactions.
For a pre-authorization:

- The authorization amount must be greater than zero.
- The authorization must be submitted for capture within 30 calendar days of its request.
- When you do not capture the authorization, you must reverse it.
  In the U.S., Canada, Latin America, and Asia Pacific, Mastercard charges an additional fee for a pre-authorization that is not captured and not reversed.
  In Europe, Russia, Middle East, and Africa, Mastercard charges fees for all pre-authorizations.
- Chargeback protection is in effect for 30 days after the authorization.

Unmarked Authorizations

An authorization is unmarked when the default authorization type is not set in your account and you do not include the **authIndicator** field in the authorization request. To set the default authorization type in your account, contact customer support.
Unmarked authorizations are supported only in the US, Canada, Latin America, and Asia Pacific. They are not supported in Europe, Russia, Middle East, and Africa.
Cybersource does not set a mark or indicator for the type of authorization in the request that is sent to the processor.

Important

Your acquirer processes an unmarked authorization as a final authorization, a preauthorization, or an undefined authorization. Contact your acquirer to learn how they process unmarked authorizations.

Requirements for Unmarked Authorizations

For an unmarked authorization:

- The authorization amount must be greater than zero.

- The authorization amount can be different from the final transaction amount.

Undefined Authorizations

An authorization is undefined when you set the default authorization type in your account to undefined and do not include the **authIndicator** field in the authorization request. To set the default authorization type in your account, contact customer support.
Undefined authorizations are supported only in the U.S., Canada, Latin America, and Asia Pacific. They are not supported in Europe, Russia, Middle East, and Africa.
Chargeback protection is in effect for seven days after the authorization.

Requirements for Undefined Authorizations

For an undefined authorization:

- The authorization amount must be greater than zero.
- The authorization amount can be different from the final transaction amount.
- The authorization should not be cancelled after it is approved except when a system failure occurs.
- The authorization must be submitted for capture within seven calendar days of its request.
- When you do not capture the authorization, you must reverse it; otherwise, Mastercard charges an additional fee for the transaction.

Required Fields for Final Authorizations

Use these required fields for final authorizations and preauthorizations.

orderInformation.amountDetails.currency

orderInformation.amountDetails.totalAmount

orderInformation.billTo.address1

orderInformation.billTo.administrativeArea

orderInformation.billTo.country

orderInformation.billTo.email

orderInformation.billTo.firstName

orderInformation.billTo.lastName

orderInformation.billTo.locality

orderInformation.billTo.postalCode

paymentInformation.card.type

paymentInformation.card.expirationMonth

paymentInformation.card.expirationYear

paymentInformation.card.number

processingInformation.authorizationOptions.authIndicator	**Set the value to `0` for preauthorizations, or to `1` for final authorizations. Do not include this field for unmarked or undefined authorizations.**

REST Example: Final Authorizations

Request

```
{
  "orderInformation" : {
    "billTo" : {
      "firstName" : "RTS",
      "lastName" : "VDP",
      "address1" : "201 S. Division St.",
      "postalCode" : "48104-2201",
      "locality" : "Ann Arbor",
      "administrativeArea" : "MI",
      "country" : "US",
      "email" : "test@cybs.com"
    },
    "amountDetails" : {
      "totalAmount" : "100.00",
      "currency" : "usd"
    }
  },
  "paymentInformation" : {
    "card" : {
      "expirationYear" : "2031",
      "number" : "4111111111111111",
      "expirationMonth" : "12",
      "type" : "001"
    }
  },
  "processingInformation" : {
    "authorizationOptions" : {
        "authIndicator" : "1"
    }
  }
}
```

Response to a Successful Request

```
{
    "_links": {
        "authReversal": {
            "method": "POST",
            "href": "/pts/v2/payments/6910040807416719003955/reversals"
        },
        "self": {
            "method": "GET",
            "href": "/pts/v2/payments/6910040807416719003955"
        },
        "capture": {
```

```
        "method": "POST",
        "href": "/pts/v2/payments/6910040807416719003955/captures"
      }
    },
    "clientReferenceInformation": {
      "code": "1691004080800"
    },
    "id": "6910040807416719003955",
    "orderInformation": {
      "amountDetails": {
        "authorizedAmount": "100.00",
        "currency": "usd"
      }
    },
    "paymentAccountInformation": {
      "card": {
        "type": "001"
      }
    },
    "paymentInformation": {
      "card": {
        "type": "001"
      }
    },
    "pointOfSaleInformation": {
      "terminalId": "111111"
    },
    "processorInformation": {
      "approvalCode": "888888",
      "networkTransactionId": "123456789619999",
      "transactionId": "123456789619999",
      "responseCode": "100",
      "avs": {
        "code": "X",
        "codeRaw": "I1"
      }
    },
    "reconciliationId": "67628631TKRG2OVE",
    "status": "AUTHORIZED",
    "submitTimeUtc": "2023-08-02T19:21:20Z"
}
```

Authorization Reversal

This section provides the information about how to process an authorization reversal. Reversing an authorization releases the hold on the customer's payment card funds that the issuing bank placed when processing the authorization.

For a debit card or prepaid card in which only a partial amount was approved, the amount of the reversal must be the amount that was authorized, not the amount that was requested.

Endpoint

Production: `POST https://api.cybersource.com/pts/v2/payments/{id}/reversals`
Test: `POST https://apitest.cybersource.com/pts/v2/payments/{id}/reversals`
The `{id}` is the transaction ID returned in the authorization response.

Required Fields for Processing an Authorization Reversal

clientReferenceInformation.code	
reversalInformation.amountDetails.currency	
reversalInformation.amountDetails.totalAmount	**The amount of the reversal must be the same as the authorization amount that was included in the authorization response message. Do not use the amount that was requested in the authorization request message.**

REST Example: Processing an Authorization Reversal

Request

```
{
   "clientReferenceInformation": {
     "code": "test123"
   }
   "reversalInformation" : {
      "amountDetails" : {
         "totalAmount" : "100.00",
         "currency" : "USD"
      }
   }
}
```

Response to a Successful Request

```
{
   "_links" : {
     "self" : {
        "method" : "GET",
        "href" : "/pts/v2/reversals/6869460219566537303955"
     }
   },
   "clientReferenceInformation" : {
      "code" : "RTS-Auth-Reversal"
   },
   "id" : "6869460219566537303955",
   "orderInformation" : {
      "amountDetails" : {
         "currency" : "USD"
      }
   },
   "processorInformation" : {
      "responseCode" : "200"
   },
   "reconciliationId" : "82kBK3qDNtls",
   "reversalAmountDetails" : {
      "reversedAmount" : "100.00",
      "currency" : "USD"
   },
   "status" : "REVERSED",
   "submitTimeUtc" : "2023-06-16T20:07:02Z"
}
```

Sales

This section provides the information you need in order to process a sale transaction. A sale combines an authorization and a capture into a single transaction.

Endpoint

Production: `POST https://api.cybersource.com/pts/v2/payments`
Test: `POST https://apitest.cybersource.com/pts/v2/payments`

Required Fields for Processing a Sale

orderInformation.amountDetails.currency	
orderInformation.amountDetails.totalAmount	
orderInformation.billTo.address1	
orderInformation.billTo.administrativeArea	
orderInformation.billTo.country	
orderInformation.billTo.email	
orderInformation.billTo.firstName	
orderInformation.billTo.lastName	
orderInformation.billTo.locality	
orderInformation.billTo.postalCode	
paymentInformation.card.expirationMonth	
paymentInformation.card.expirationYear	
paymentInformation.card.number	
processingInformation.capture	Set the value to `true`.

Related Information

- *API field reference guide for the REST API*

REST Example: Processing a Sale

Request

```
{
 "processingInformation": {
  "capture": true
 },
 "orderInformation" : {
  "billTo" : {
  "country" : "US",
  "lastName" : "VDP",
  "address1" : "201 S. Division St.",
  "postalCode" : "48104-2201",
  "locality" : "Ann Arbor",
  "administrativeArea" : "MI",
  "firstName" : "RTS",
  "email" : "test@cybs.com"
 },
  "amountDetails" : {
   "totalAmount" : "100.00",
   "currency" : "usd"
   }
  },
 "paymentInformation" : {
  "card" : {
   "expirationYear" : "2031",
   "number" : "4111111111111111",
   "expirationMonth" : "12",
   "type" : "001
  }
 }
}
```

Response to a Successful Request

Most processors do not return all of the fields that are shown in this example.

```
{
 "_links" : {
  "void" : {
   "method" : "POST",
   "href" : "/pts/v2/payments/6485004068966546103093/voids"
  },
  "self" : {
   "method" : "GET",
   "href" : "/pts/v2/payments/6485004068966546103093"
  }
 },
 "clientReferenceInformation" : {
  "code" : "RTS-Auth"
 },
 "id" : "6485004068966546103093",
 "orderInformation" : {
  "amountDetails" : {
   "totalAmount" : "100.00",
   "authorizedAmount" : "100.00",
   "currency" : "usd"
  }
 },
 "paymentAccountInformation" : {
  "card" : {
   "type" : "001"
  }
 },
 "paymentInformation" : {
  "tokenizedCard" : {
   "type" : "001"
  },
  "card" : {
   "type" : "001"
  }
 },
 "processorInformation" : {
  "systemTraceAuditNumber" : "841109",
  "approvalCode" : "831000",
  "merchantAdvice" : {
   "code" : "01",
   "codeRaw" : "M001"
  },
  "responseDetails" : "ABC",
  "networkTransactionId" : "016153570198200",
  "retrievalReferenceNumber" : "208720841109",
  "consumerAuthenticationResponse" : {
   "code" : "2",
   "codeRaw" : "2"
  },
  "transactionId" : "016153570198200",
  "responseCode" : "00",
  "avs" : {
   "code" : "Y",
   "codeRaw" : "Y"
  }
 },
 "reconciliationId" : "64850040689665461O3093",
 "status" : "AUTHORIZED",
 "submitTimeUtc" : "2022-03-28T20:46:47Z"
}
```

Sales with Payment Network Tokens

This section shows you how to successfully process a sale with payment network tokens.

Important

Due to mandates from the Reserve Bank of India, Indian merchants cannot store personal account numbers (PAN). Use network tokens instead. For

> more information on network tokens, see *Network Tokenization* in the Token Management Service Developer Guide.

Endpoint

Production: `POST https://api.cybersource.com/pts/v2/payments`
Test: `POST https://apitest.cybersource.com/pts/v2/payments`

Required Fields for Sales with Payment Network Tokens

Use these required fields for processing a sale with payment network tokens.

orderInformation.amountDetails.currency	
orderInformation.amountDetails.totalAmount	
orderInformation.billTo.address1	
orderInformation.billTo.email	
orderInformation.billTo.firstName	
orderInformation.billTo.lastName	
paymentinformation.tokenizedCard.cryptogram	
paymentinformation.tokenizedCard.expirationMonth	
paymentinformation.tokenizedCard.expirationYear	
processingInformation.capture	**Set the value to `true`.**

Related Information

- *API field reference guide for the REST API*

Optional Fields for Sales with Payment Network Tokens

You can use these optional fields to include additional information when processing a sale with a payment network token.

clientReferenceInformation.code	
consumerAuthenticationInformation.cavv	**For 3-D Secure in-app transactions for Visa and JCB, set this field to the 3-D Secure cryptogram. Otherwise, set to the network token cryptogram.**
consumerAuthenticationInformation.ucafAuthenticationData	**For Mastercard requests using 3-D Secure, set this field to the Identity Check cryptogram.**
consumerAuthenticationInformation.ucafCollectionIndicator	**For Mastercard requests using 3-D Secure, set the value to `2`.**

orderInformation.amountDetails.currency	
orderInformation.amountDetails.totalAmount	
orderInformation.billTo.address1	
orderInformation.billTo.country	
orderInformation.billTo.email	
orderInformation.billTo.firstName	
orderInformation.billTo.lastName	
orderInformation.billTo.locality	
orderInformation.billTo.postalCode	**Required only for transactions in the US and Canada.**
orderInformation.billTo.administrativeArea	**Required only for transactions in the US and Canada.**
processingInformation.commerceIndicator	
paymentInformation.tokenizedCard.cardType	**It is strongly recommended that you send the card type even if it is optional for your processor. Omitting the card type can cause the transaction to be processed with the wrong card type.**
paymentInformation.tokenizedCard.cryptogram	
paymentInformation.tokenizedCard.expirationMonth	**Set to the token expiration month that you received from the token service provider.**
paymentInformation.tokenizedCard.expirationYear	**Set to the token expiration year that you received from the token service provider.**
paymentInformation.tokenizedCard.number	**Set to the token value that you received from the token service provider.**
paymentInformation.tokenizedCard.requestorId	
paymentInformation.tokenizedCard.transactionType	

Related Information

- *API field reference guide for the REST API*

REST Example: Sales with Payment Network Tokens

Request

```
{
 "orderInformation" : {
  "billTo": {
   "country": "US",
   "lastName": "Kim",
   "address1": "201 S. Division St.",
   "postalCode": "48104-2201",
   "locality": "Ann Arbor",
   "administrativeArea": "MI",
   "firstName": "Smith",
   "email": "test@cybs.com"
  },
  "amountDetails" : {
   "totalAmount" : "100",
   "currency" : "USD"
  }
 },
  "paymentInformation" : {
  "tokenizedCard" : {
   "expirationYear" : "2031",
   "number" : "4111111111111111",
   "expirationMonth" : "12",
   "transactionType" : "1",
   "cryptogram" : "qE5juRwDzAUFBAkEHuWW9PiBkWv="
  }
 }
}
```

Response to a Successful Request

```
{
   "_links": {
      "authReversal": {
         "method": "POST",
         "href": "/pts/v2/payments/6838294805206235603954/reversals"
      },
      "self": {
         "method": "GET",
         "href": "/pts/v2/payments/6838294805206235603954"
      },
      "capture": {
         "method": "POST",
         "href": "/pts/v2/payments/6838294805206235603954/captures"
      }
   },
   "clientReferenceInformation": {
      "code": "1683829480593"
   },
   "id": "6838294805206235603954",
   "orderInformation": {
      "amountDetails": {
         "authorizedAmount": "100.00",
         "currency": "USD"
      }
   },
   "paymentAccountInformation": {
      "card": {
         "type": "001"
      }
   },
   "paymentInformation": {
      "tokenizedCard": {
        "type": "001"
      },
      "card": {
        "type": "001"
      }
    },
    "pointOfSaleInformation": {
      "terminalId": "111111"
    },
    "processorInformation": {
      "approvalCode": "888888",
      "networkTransactionId": "123456789619999",
      "transactionId": "123456789619999",
      "responseCode": "100",
      "avs": {
        "code": "1"
      }
    },
    "reconciliationId": "60332034UHI9PRJ0",
    "status": "AUTHORIZED",
    "submitTimeUtc": "2023-05-11T18:24:40Z"
}
```

Captures

This section provides the information you need in order to capture an authorized transaction.

Endpoint

Production: `POST https://api.cybersource.com/pts/v2/payments/{id}/captures`
Test: `POST https://apitest.cybersource.com/pts/v2/payments/{id}/captures`
The `{id}` is the transaction ID returned in the authorization response.

Required Fields for Capturing an Authorization

Use these required fields for capturing an authorization.

clientReferenceInformation.code	This field value maps from the original authorization, sale, or credit transaction.
orderInformation.amountDetails.currency	
orderInformation.amountDetails.totalAmount	

REST Example: Capturing an Authorization

Request

```
{
  "clientReferenceInformation": {
      "code": "ABC123"
   },
   "orderInformation": {
      "amountDetails": {
         "totalAmount": "100.00",
         "currency": "EUR"
   }
}
```

Response to a Successful Request

```
{
   "_links": {
      "void": {
         "method": "POST",
         "href": "/pts/v2/captures/6662994431376681303954/voids"
      },
      "self": {
         "method": "GET",
         "href": "/pts/v2/captures/6662994431376681303954"
      }
   },
   "clientReferenceInformation": {
      "code": "1666299443215"
   },
   "id": "6662994431376681303954",
   "orderInformation": {
      "amountDetails": {
         "totalAmount": "100.00",
         "currency": "EUR"
      }
   },
   "reconciliationId": "66535942B9CGT52U",
   "status": "PENDING",
   "submitTimeUtc": "2022-10-20T20:57:23Z"
}
```

Multiple Partial Captures

This section shows you how to process multiple partial captures for an authorization.

This feature enables you to request multiple partial captures for one authorization. A multiple partial capture allows you to incrementally settle authorizations over time. Ensure that the total amount of all the captures does not exceed the authorized amount.

Prerequisite

Contact customer support to have your account enabled for this feature.

Endpoint

Production: `POST https://api.cybersource.com/pts/v2/payments/{id}/captures`

Test: `POST https://apitest.cybersource.com/pts/v2/payments/{id}/captures`
The `{id}` is the transaction ID returned in the authorization response.

Required Fields for Processing Multiple Partial Captures

clientReferenceInformation.code	Set to **clientReferenceInformation.code** value used in corresponding authorization request.
orderInformation.amountDetails.currency	
orderInformation.amountDetails.totalAmount	

Related Information

- *API field reference guide for the REST API*

REST Example: Processing Multiple Partial Captures

Request

```
{
 {
 "clientReferenceInformation": {
  "code": "TC50171_3"
 },
 "processingInformation": {
  "captureOptions": {
    "captureSequenceNumber": "2",
    "totalCaptureCount": "3"
  }
 },
 "orderInformation": {
  "amountDetails": {
    "totalAmount": "102.21",
    "currency": "USD"
  }
 }
}
```

Response to a Successful Request

```
{
 "_links": {
  "void": {
    "method": "POST",
    "href": "/pts/v2/captures/6742496815656503003954/voids"
  },
  "self": {
    "method": "GET",
    "href": "/pts/v2/captures/6742496815656503003954"
  }
 },
 "clientReferenceInformation": {
  "code": "TC50171_3"
 },
 "id": "6742496815656503003954",
 "orderInformation": {
  "amountDetails": {
    "totalAmount": "102.21",
    "currency": "USD"
  }
 },
 "reconciliationId": "67332020GD2G1OO1",
 "status": "PENDING",
 "submitTimeUtc": "2023-01-20T21:21:21Z"
}
```

Refunds

This section provides the information you need in order to process a refund, which is linked to a capture or sale. You must request a refund within 180 days of the authorization.

When your account is enabled for credit authorizations, also known as purchase return authorizations, Cybersource authenticates the card and customer during a refund or credit request. Every credit request is automatically authorized.

Credit authorization results are returned in these response fields:

- **processorInformation.approvalCode**
- **processorInformation.networkTransactionId**
- **processorInformation.responseCode**

When you request a void for the credit and the credit is voided. If your account is enabled for credit authorizations, the credit authorization is also reversed.

Endpoint

Production: `POST https://api.cybersource.com/pts/v2/payments/{id}/refunds`

Test: `POST https://apitest.cybersource.com/pts/v2/payments/{id}/refunds`

The `{id}` is the transaction ID returned in the capture or sale response.

Required Fields for Processing a Refund

Use these required fields for processing a refund.

orderInformation.amountDetails.currency

orderInformation.amountDetails.totalAmount

Related Information

- *API field reference guide for the REST API*

REST Interactive Example: Processing a Refund

Refund a Payment

Live Console URL: *https://developer.cybersource.com/api-reference-assets/index.html#payments_refund_refund-a-payment*

REST Example: Processing a Refund

Request

```
{
  "orderInformation": {
    "amountDetails": {
      "totalAmount": "100.00",
      "currency": "EUR"
    }
  }
}
```

Response to a Successful Request

```
{
  "_links": {
    "void": {
      "method": "POST",
      "href": "/pts/v2/credits/6699964581696622603955/voids"
    },
    "self": {
      "method": "GET",
      "href": "/pts/v2/credits/6699964581696622603955"
    }
  },
  "clientReferenceInformation": {
    "code": "1669996458298"
  },
  "creditAmountDetails": {
    "currency": "eur",
    "creditAmount": "100.00"
  },
  "id": "6699964581696622603955",
  "orderInformation": {
    "amountDetails": {
      "currency": "EUR"
    }
  },
  "paymentAccountInformation": {
    "card": {
      "type": "001"
    }
  },
  "paymentInformation": {
    "tokenizedCard": {
      "type": "001"
    },
      "card": {
        "type": "001"
      }
    },
    "processorInformation": {
      "approvalCode": "888888",
      "networkTransactionId": "016153570198200",
      "responseCode": "100"
    },
    "reconciliationId": "61873329OAILG3Q6",
    "status": "PENDING",
    "submitTimeUtc": "2022-12-02T15:54:18Z"
}
```

Request

```
{
  "clientReferenceInformation": {
    "code": "CNP-FOR-1"
  },
  "orderInformation": {
    "billTo": {
      "firstName": "first"
    },
    "amountDetails": {
      "totalAmount": "0.23",
      "currency": "AUD"
    }
  }
}
```

Response to a Successful Request

```
{
  "_links": {
    "void": {
      "method": "POST",
      "href": "/pts/v2/refunds/7204282190676807103
093/voids"
    },
    "self": {
      "method": "GET",
      "href": "/pts/v2/refunds/7204282190676807103
093"
    }
  },
  "clientReferenceInformation": {
    "code": "CNP-FOR-1"
  },
  "id": "7204282190676807103093",
  "orderInformation": {
    "amountDetails": {
      "currency": "AUD"
    }
  },
  "processorInformation": {
   "systemTraceAuditNumber": "468221",
   "retrievalReferenceNumber": "468221433908",
   "seblementDate": "0708",
   "responseCode": "00"
  },
  "reconciliationId": "7204282190676807103093",
  "refundAmountDetails": {
   "currency": "AUD",
   "refundAmount": "0.23"
  },
  "status": "PENDING",
  "submitTimeUtc": "2024-07-
08T08:43:39Z"
}
```

Credits

This section shows you how to process a credit, which is not linked to a capture or sale. There is no time limit for requesting a credit.

When your account is enabled for credit authorizations, also known as purchase return authorizations, Cybersource authenticates the card and customer during a refund or credit request. Every credit request is automatically authorized.

Credit authorization results are returned in these response fields:

- **processorInformation.approvalCode**
- **processorInformation.networkTransactionId**
- **processorInformation.responseCode**

When you request a void for the credit and the credit is voided. If your account is enabled for credit authorizations, the credit authorization is also reversed.

Endpoint

Production: `POST https://api.cybersource.com/pts/v2/credits/`

Test: `POST https://apitest.cybersource.com/pts/v2/credits/`

Required Fields for Processing a Credit

Use these required fields for processing a credit.

orderInformation.amountDetails.currency

orderInformation.amountDetails.totalAmount

orderInformation.billTo.address1

orderInformation.billTo.administrativeArea

orderInformation.billTo.country

orderInformation.billTo.email

orderInformation.billTo.firstName

orderInformation.billTo.lastName

orderInformation.billTo.locality

orderInformation.billTo.postalCode

paymentInformation.card.expirationMonth

paymentInformation.card.expirationYear

paymentInformation.card.number

REST Interactive Example: Processing a Credit

```
Credit
```

Live Console URL: *https://developer.cybersource.com/api-reference-assets/index.html#payments_credit_process-a-credit*

REST Example: Processing a Credit

Request

```
{
  "orderInformation" : {
    "billTo" : {
      "country" : "US",
      "lastName" : "Kim",
      "address1" : "201 S. Division St.",
      "postalCode" : "48104-2201",
      "locality" : "Ann Arbor",
      "administrativeArea" : "MI",
      "firstName" : "Kyong-Jin",
      "email" : "test@cybs.com"
    },
    "amountDetails" : {
      "totalAmount" : "100.00",
      "currency" : "eur"
    }
  },
  "paymentInformation" : {
    "card" : {
      "expirationYear" : "2031",
      "number" : "4111111111111111",
      "expirationMonth" : "12"
    }
  }
}
```

Response to a Successful Request

```
{
    "_links": {
        "void": {
            "method": "POST",
            "href": "/pts/v2/credits/6663069906146706403954/voids"
```

```
      },
      "self": {
        "method": "GET",
        "href": "/pts/v2/credits/6663069906146706403954"
      }
    },
    "clientReferenceInformation": {
      "code": "1666306990717"
    },
    "creditAmountDetails": {
      "currency": "eur",
      "creditAmount": "100.00"
    },
    "id": "6663069906146706403954",
    "orderInformation": {
      "amountDetails": {
        "currency": "eur"
      }
    },
    "paymentAccountInformation": {
      "card": {
        "type": "001"
      }
    },
    "paymentInformation": {
      "tokenizedCard": {
        "type": "001"
      },
      "card": {
        "type": "001"
      }
    },
    "processorInformation": {
      "approvalCode": "888888",
      "networkTransactionId": "016153570198200",
      "responseCode": "100"
    },
    "reconciliationId": "66490108K9CLFJPN",
    "status": "PENDING",
    "submitTimeUtc": "2022-10-20T23:03:10Z"
}
```

Voids for a Capture or Credit

This section describes how to void a capture or credit that was submitted but not yet processed by the processor.

Endpoint

Void a Capture
Production: `POST https://api.cybersource.com/pts/v2/captures/{id}/voids`
Test: `POST https://apitest.cybersource.com/pts/v2/captures/{id}/voids`

Void a Credit
Production: `POST https://api.cybersource.com/pts/v2/credits/{id}/voids`
Test: `POST https://apitest.cybersource.com/pts/v2/credits/{id}/voids`
The `{id}` is the transaction ID returned during the capture or credit response.

Required Fields for Voiding a Capture or Credit

clientReferenceInformation.code	**Including this field is recommended, but not required.**

REST Example: Voiding a Capture or Credit

Request

```
{
   "clientReferenceInformation": {
      "code": "test123"
   }
}
```

Response to a Successful Request

```
{
   "_links": {
      "self": {
         "method": "GET",
         "href": "/pts/v2/voids/6541933390746728203005"
      }
   },
   "clientReferenceInformation": {
      "code": "1654193339056"
   },
   "id": "6541933390746728203005",
   "orderInformation": {
      "amountDetails": {
      "currency": "USD"
      }
   },
   "status": "VOIDED",
   "submitTimeUtc": "2022-06-02T18:08:59Z",
   "voidAmountDetails": {
      "currency": "usd",
      "voidAmount": "100.00"
   }
}
```

Debit and Prepaid Card Processing

This section shows you how to process authorizations that use a debit or prepaid card.

Related Information

- See *Debit and Prepaid Card Payments* on page 23 for a description of the debit or prepaid card transactions you can process.

Additional Resources for Debit and Prepaid Payments

For more information, see these guides:

- *API field reference guide for the REST API*
- Github repositories: *https://github.com/Cybersource*

Processing Debit and Prepaid Authorizations

This section shows you how to process an authorization using debit and prepaid cards.

Endpoint

Production: `POST https://api.cybersource.com/pts/v2/payments`
Test: `POST https://apitest.cybersource.com/pts/v2/payments`

Required Fields for Processing Debit and Prepaid Authorizations

Use these required fields for processing debit and prepaid authorizations.

clientReferenceInformation.code

orderInformation.amountDetails.currency

orderInformation.amountDetails.totalAmount

orderInformation.billTo.address1

orderInformation.billTo.administrativeArea

orderInformation.billTo.country

orderInformation.billTo.email

orderInformation.billTo.firstName

orderInformation.billTo.lastName

orderInformation.billTo.locality

orderInformation.billTo.postalCode

paymentInformation.card.type

paymentInformation.card.expirationMonth

paymentInformation.card.expirationYear

paymentInformation.card.number

Related Information

- *API field reference guide for the REST API*

Optional Field for Processing Debit and Prepaid Authorizations

You can use this optional field to include additional information when processing debit and prepaid authorizations.

processingInformation.linkId	**Set this field to the request ID that was returned in the response message from the original authorization request.**

Related Information

- *API field reference guide for the REST API*

REST Example: Processing Debit and Prepaid Authorizations

Request

```
{
 "orderInformation" : {
  "billTo" : {
   "country" : "US",
   "firstName" : "John",
   "lastName" : "Deo",
      "address1" : "901 Metro Center Blvd",
      "postalCode" : "40500",
      "locality" : "Foster City",
      "administrativeArea" : "CA",
      "email" : "test@cybs.com"
 },
    "amountDetails" : {
      "totalAmount" : "100.00",
      "currency" : "USD"
    }
  },
  "paymentInformation" : {
    "card" : {
      "expirationYear" : "2031",
      "number" : "4111111111111111",
      "securityCode" : "123",
      "expirationMonth" : "12",
      "type" : "001"
    }
  }
}
```

Response to a Successful Request

```
{
  "_links" : {
    "authReversal" : {
      "method" : "POST",
      "href" : "/pts/v2/payments/6595482584316313203494/reversals"
    },
    "self" : {
      "method" : "GET",
      "href" : "/pts/v2/payments/6595482584316313203494"
    },
    "capture" : {
      "method" : "POST",
      "href" : "/pts/v2/payments/6595482584316313203494/captures"
    }
  },
  "clientReferenceInformation" : {
    "code" : "RTS-Auth"
  },
  "consumerAuthenticationInformation" : {
    "token" : "Axj/7wSTZYq1MhJBMfMmAEQs2auWrRwyauGjNi2ZsWbJgzaOWiaVA+JbK
          AU0qB8S2VpA6cQIp4ZNvG2YbC9eM4E5NlirUyEkEx8yYAAA4A1c"
  },
  "id" : "6595482584316313203494",
  "orderInformation" : {
    "amountDetails" : {
      "authorizedAmount" : "100.00",
      "currency" : "USD"
    }
  },
  "paymentAccountInformation" : {
    "card" : {
      "type" : "001"
    }
  },
  "paymentInformation" : {
    "tokenizedCard" : {
      "type" : "001"
    },
    "card" : {
      "type" : "001"
    }
  },
  "processorInformation" : {
    "systemTraceAuditNumber" : "853428",
    "approvalCode" : "831000",
    "cardVerification" : {
      "resultCodeRaw" : "M",
      "resultCode" : "M"
    },
    "merchantAdvice" : {
      "code" : "01",
      "codeRaw" : "M001"
    },
    "responseDetails" : "ABC",
    "networkTransactionId" : "016153570198200",
    "retrievalReferenceNumber" : "221517853428",
    "consumerAuthenticationResponse" : {
      "code" : "2",
      "codeRaw" : "2"
    },
    "transactionId" : "016153570198200",
    "responseCode" : "00",
    "avs" : {
      "code" : "Y",
      "codeRaw" : "Y"
    }
  }
}
```

Enabling Debit and Prepaid Partial Authorizations

Partial authorizations and balance responses are special features that are available for debit cards and prepaid cards. This section shows you how to enable partial authorizations for a specific transaction.

Field Specific to this Use Case

Include this field in addition to the fields required for a standard authorization request:

- Indicate that this request is a partial authorization.
  Set the **processingInformation.authorizationOptions.partialAuthIndicator** to `true`.

Endpoint

Production: `POST https://api.cybersource.com/pts/v2/payments`
Test: `POST https://apitest.cybersource.com/pts/v2/payments`

Required Fields for Enabling Debit and Prepaid Partial Authorizations

Use these required fields for enabling debit and prepaid partial authorizations.

clientReferenceInformation.code	
orderInformation.amountDetails.currency	
orderInformation.amountDetails.totalAmount	
orderInformation.billTo.address1	
orderInformation.billTo.administrativeArea	
orderInformation.billTo.country	
orderInformation.billTo.email	
orderInformation.billTo.firstName	
orderInformation.billTo.lastName	
orderInformation.billTo.locality	
orderInformation.billTo.postalCode	
paymentInformation.card.type	
paymentInformation.card.expirationMonth	
paymentInformation.card.expirationYear	
paymentInformation.card.number	
processingInformation.authorizationOptions.partialAuthIndicator	Set the value to `true`.

Related Information

- *API field reference guide for the REST API*

Optional Field for Enabling Debit and Prepaid Partial Authorizations

You can use these optional fields to include additional information when enabling debit and prepaid partial authorizations.

processingInformation.linkId	**Set this field to the request ID that was returned in the response message from the original authorization request.**

Related Information

- *API field reference guide for the REST API*

REST Example: Enabling Debit and Prepaid Partial Authorizations

Request

```
{
 "clientReferenceInformation" : {
  "code" : "TC50171_3"
 },
 "orderInformation" : {
  "billTo" : {
   "country" : "US",
   "lastName" : "Deo",
   "address2" : "Address 2",
   "address1" : "201 S. Division St.",
   "postalCode" : "48104-2201",
   "locality" : "Ann Arbor",
   "administrativeArea" : "MI",
   "firstName" : "John",
   "phoneNumber" : "999999999",
   "district" : "MI",
   "buildingNumber" : "123",
   "company" : "Visa",
   "email" : "test@cybs.com"
  },
  "amountDetails" : {
   "totalAmount" : "1000.00",
   "currency" : "USD"
  }
 },
 "paymentInformation" : {
  "card" : {
   "expirationYear" : "2031",
   "number" : "5555555555xxxxxx",
   "securityCode" : "123",
   "expirationMonth" : "12",
   "type" : "002"
  }
 },
"processingInformation" : {
 "authorizationOptions" : {
   "partialAuthIndicator" : "true"
  }
 }
}
```

Response to a Successful Request

```
{
 "_links" : {
  "self" : {
   "method" : "GET",
   "href" : "/pts/v2/payments/6595549144566655003494"
```

```
    }
  },
  "clientReferenceInformation" : {
    "code" : "TC50171_3"
  },
  "id" : "6595549144566655003494",
  "orderInformation" : {
    "amountDetails" : {
      "totalAmount" : "1000.00",
      "authorizedAmount" : "499.01",
      "currency" : "USD"
    }
  },
  "paymentInformation" : {
    "accountFeatures" : {
      "currency" : "usd",
      "balanceAmount" : "0.00"
    }
  },
  "pointOfSaleInformation" : {
    "terminalId" : "261996"
  },
  "processorInformation" : {
    "merchantNumber" : "000000092345678",
    "approvalCode" : "888888",
    "cardVerification" : {
      "resultCode" : ""
    },
    "networkTransactionId" : "123456789619999",
    "transactionId" : "123456789619999",
    "responseCode" : "100",
    "avs" : {
      "code" : "X",
      "codeRaw" : "I1"
    }
  },
  "reconciliationId" : "56059417N6C86KTJ",
  "status" : "PARTIAL_AUTHORIZED",
  "submitTimeUtc" : "2022-08-03T19:28:34Z"
}
```

Disabling Debit and Prepaid Partial Authorizations

This topic shows you how to successfully disable partial authorizations for specific transactions.

Field Specific to this Use Case

Include this field in addition to the fields required for a standard authorization request:

- Indicate that this request is not a partial authorization.
  Set the `processingInformation.authorizationOptions.partialAuthIndicator` to `false`.

Endpoint

Production: `POST https://api.cybersource.com/pts/v2/payments`
Test: `POST https://apitest.cybersource.com/pts/v2/payments`

Required Field for Disabling Debit and Prepaid Partial Authorizations

Use these required fields for disabling debit and prepaid partial authorizations.

clientReferenceInformation.code	
orderInformation.amountDetails.currency	
orderInformation.amountDetails.totalAmount	
orderInformation.billTo.address1	
orderInformation.billTo.administrativeArea	
orderInformation.billTo.country	
orderInformation.billTo.email	
orderInformation.billTo.firstName	
orderInformation.billTo.lastName	
orderInformation.billTo.locality	
orderInformation.billTo.postalCode	
paymentInformation.card.type	
paymentInformation.card.expirationMonth	
paymentInformation.card.expirationYear	
paymentInformation.card.number	
processingInformation.authorizationOptions.partialAuthIndicator	Set the value to `false` in an authorization or sale request. When you do so, only that specific transaction is disabled for partial authorization.

Related Information

- *API field reference guide for the REST API*

Optional Field for Disabling Debit and Prepaid Partial Authorizations

You can use this optional field to include additional information when disabling debit and prepaid partial authorizations.

processingInformation.linkId	**Set this field to the request ID that was returned in the response message from the original authorization request.**

Related Information

- *API field reference guide for the REST API*

REST Example: Disabling Debit and Prepaid Partial Authorizations

Request

```
{
   "processingInformation":{
  "authorizationOptions":{
   "partialAuthIndicator": "false"
  }
 },
  "clientReferenceInformation" : {
    "code" : "TC50171_3"
  },
  "orderInformation" : {
    "billTo" : {
      "country" : "US",
      "lastName" : "Deo",
      "address2" : "Address 2",
      "address1" : "201 S. Division St.",
      "postalCode" : "48104-2201",
      "locality" : "Ann Arbor",
      "administrativeArea" : "MI",
      "firstName" : "John",
      "phoneNumber" : "999999999",
      "district" : "MI",
      "buildingNumber" : "123",
      "company" : "Visa",
      "email" : "test@cybs.com"
    },
    "amountDetails" : {
      "totalAmount" : "501.00",
      "currency" : "USD"
    }
  },
  "paymentInformation" : {
    "card" : {
      "expirationYear" : "2031",
      "number" : "5555555555xxxxxx",
      "securityCode" : "123",
      "expirationMonth" : "12",
      "type" : "002"
    }
  }
}
```

Response to a Successful Request

```
{
   "_links": {
      "self": {
         "method": "GET",
         "href": "/pts/v2/payments/6595545423896900104953"
      }
   },
   "clientReferenceInformation": {
      "code": "TC50171_3"
   },
   "errorInformation": {
      "reason": "PROCESSOR_DECLINED",
      "message": "Decline - General decline of the card.
               No other information provided by the issuing bank."
   },
   "id": "6595545423896900104953",
   "pointOfSaleInformation": {
      "terminalId": "111111"
   },
   "processorInformation": {
      "networkTransactionId": "123456789619999",
      "transactionId": "123456789619999",
      "responseCode": "100",
      "avs": {
         "code": "X",
         "codeRaw": "I1"
      }
   },
   "status": "DECLINED"
}
```

Payer Authentication Processing

This section shows you how to process authorizations that use these payer authentication methods:

- Mastercard: Identity Check
- Visa: Visa Secure

Related Information

- See the *Payer Authentication Developer Guide* for details about payer authentication.

Additional Resources for Payer Authentication

For more information, see these guides:

- *Payer Authentication Developer Guide*
- *API field reference guide for the REST API*
- Github repositories: *https://github.com/Cybersource*

Providing Payer Authentication Information for Authorization

The values that are returned from payer authentication must be provided when seeking authorization for the transaction. Authentication information that is not included when considering authorization may cause the transaction to be refused or downgraded and prevent the normal liability shift from occurring.

The level of security in payer authentication is denoted by the two digit Electronic Commerce Indicator (ECI) that is assigned to the transaction. These digital values have text equivalents which are assigned to the **processingInformation.commerceIndicator** field.

The American Express, Diners, Discover, UPI, and Visa card brands use 05, 06, and 07 digit values to express the authentication level for a 3-D Secure transaction.

Text Values for ECI Values

ECI Value	Meaning	Visa	Diners	Discover	UPI	Amex
05	Authenticated	vbv	pb	dipb	up3ds	aesk
06	Attempted authentication with a cryptogram	vbv_attempted	pb_attempted	dipb_attempted	up3ds_attempted	aesk_attempted
07	Internet, not authenticated	vbv_failure/internet	internet	internet	up3ds_failure/internet	internet

Mastercard and Maestro cards use 00, 01, 02, 06, and 07 digit values to indicate the authentication level of the transaction.

Mastercard/Maestro Text Values for ECI Values

ECI Value	Meaning	Mastercard/Maestro
00	Internet, not authenticated	spa/internet
01	Attempted authentication	spa
02	Authenticated	spa
06	Exemption from authentication or network token without 3#D Secure	spa
07	Authenticated merchant-initiated transaction	spa

The payer authentication response contains other information that needs to be passed on for successful authorization. Be sure to include these fields when requesting a separate authorization:

- **consumerAuthenticationInformation.directoryServerTransactionId** (Mastercard, Maestro, UPI only)
- **consumerAuthenticationInformation.eciRaw**
- **consumerAuthenticationInformation.paresStatus**
- **consumerAuthenticationInformation.paSpecificationVersion**

- **consumerAuthenticationInformation.ucafAuthenticationData** (Mastercard/Maestro only)
- **consumerAuthenticationInformation.ucafCollectionIndicator** (Mastercard/Maestro only)
- **consumerAuthenticationInformation.cavv**
- **consumerAuthenticationInformation.xid**

Mastercard Identity Check

Mastercard Identity Check is the authentication service in the Mastercard card network that uses the 3-D Secure protocol in online transactions to authenticate customers at checkout.

Mastercard Identity Check generates a unique, 32-character transaction token, called the account authentication value (AAV) each time a Mastercard Identity Check-enabled account holder makes an online purchase. The AAV binds the account holder to a specific transaction. Mastercard Identity Check transactions use the universal cardholder authentication field (UCAF) as a standard to collect and pass AAV data.

Before implementing payer authentication for Mastercard Identity Check, contact customer support to have your account configured for this feature.

Fields Specific to the Mastercard Identity Check Use Case

These API fields are required specifically for this use case.

consumerAuthenticationInformation. directory ServerTransactionId	Set this field to the transaction ID returned by Mastercard Identity Check during the authentication process.
consumerAuthenticationInformation. paSpecificationVersion	Set this field to the Mastercard Identity Check version returned by Mastercard Identity Check during the authentication process.
consumerAuthenticationInformation. ucafCollectionIndicator	Set to the last digit of the raw ECI value returned from authentication. For example, if ECI=02, this value should be 2.
processingInformation.commerceIndicator	Set this field to one of these values: • `spa`: Successful authentication (3-D Secure value of `02`). • `spa`: Authentication was attempted (3-D Secure value of `01`). • `spa` or `internet`: Authentication failed or was not attempted (3-D Secure value of `00`)

Endpoint

Production: `POST https://api.cybersource.com/pts/v2/payments`
Test: `POST https://apitest.cybersource.com/pts/v2/payments`

Required Fields for Processing an Authorization Using Mastercard Identity Check

Use these required fields to process an authorization using Mastercard Identity Check.

consumerAuthenticationInformation.directoryServerTransactionId	
consumerAuthenticationInformation.paSpecificationVersion	
consumerAuthenticationInformation.ucafCollectionIndicator	**Set to the last digit of the raw ECI value returned from authentication. For example, if ECI=02, this value should be 2.**
orderInformation.amountDetails.currency	
orderInformation.amountDetails.totalAmount	
orderInformation.billTo.address1	
orderInformation.billTo.administrativeArea	
orderInformation.billTo.country	
orderInformation.billTo.email	
orderInformation.billTo.firstName	
orderInformation.billTo.lastName	
orderInformation.billTo.locality	
orderInformation.billTo.postalCode	
paymentInformation.card.expirationMonth	
paymentInformation.card.expirationYear	
paymentInformation.card.number	
processingInformation.commerceIndicator	**Set this field to one of these values:** • `spa`: Successful authentication (3-D Secure value of `02`). • `spa`: Authentication was attempted (3-D Secure value of `01`). • `spa` or `internet`: Authentication failed or was not attempted (3-D Secure value of `00`)

Related Information

- *API field reference guide for the REST API*

REST Example: Processing an Authorization Using Mastercard Identity Check

Request

```
{
  "clientReferenceInformation" : {
    "code" : "TC50171_6"
  },
  "consumerAuthenticationInformation" : {
    "ucafCollectionIndicator" : "2",
    "ucafAuthenticationData" : "EHuWW9PiBkWvqE5juRwDzAUFBAk",
    "directoryServerTransactionId" : "f38e6948-5388-41a6-bca4-b49723c19437",
    "paSpecificationVersion" : "2.2.0"
  },
  "processingInformation" : {
    "commerceIndicator" : "spa"
    },
    "orderInformation" : {
      "billTo" : {
        "country" : "US",
        "lastName" : "Deo",
        "address1" : "201 S. Division St.",
        "postalCode" : "48104-2201",
        "locality" : "Ann Arbor",
        "administrativeArea" : "MI",
        "firstName" : "John",
        "email" : test@cybs.com
      },
      "amountDetails" : {
        "totalAmount" : "105.00",
        "currency" : "USD"
      }
    },
    "paymentInformation" : {
      "card" : {
        "expirationYear" : "2031",
        "number" : "5555555555555XXXX",
        "securityCode" : "123",
        "expirationMonth" : "12",
        "type" : "002"
    }
  }
}
```

Response to a Successful Request

```
{
  "_links": {
    "authReversal": {
      "method": "POST",
      "href": "/pts/v2/payments/6758990751436655004951/reversals"
    },
    "self": {
      "method": "GET",
      "href": "/pts/v2/payments/6758990751436655004951"
    },
    "capture": {
      "method": "POST",
      "href": "/pts/v2/payments/6758990751436655004951/captures"
    }
  },
  "clientReferenceInformation": {
    "code": "TC50171_3"
  },
  "id": "6758990751436655004951",
  "orderInformation": {
    "amountDetails": {
      "authorizedAmount": "100.00",
      "currency": "USD"
    }
  },
  "paymentAccountInformation": {
    "card": {
      "type": "002"
    }
  },
  "paymentInformation": {
    "tokenizedCard": {
      "type": "002"
    },
    "card": {
      "type": "002"
    }
  },
  "pointOfSaleInformation": {
    "terminalId": "111111"
  },
  "processorInformation": {
    "approvalCode": "888888",
    "authIndicator": "1",
    "networkTransactionId": "123456789619999",
    "transactionId": "123456789619999",
    "responseCode": "100",
    "avs": {
      "code": "X",
      "codeRaw": "I1"
    }
  },
  "reconciliationId": "71183995FDU0YRTK",
  "status": "AUTHORIZED",
  "submitTimeUtc": "2023-02-08T23:31:15Z"
}
```

Visa Secure

Visa Secure is the authentication service in the Visa card network that uses the 3-D Secure protocol to authenticate customers at checkout. This authentication is a two-step process. First, the cardholder is authenticated by 3-D Secure. Then, the transaction is authorized based on the 3-D Secure evaluation. This section explains how to authorize a card payment based on the 3-D Secure evaluation.
Before implementing Visa Secure, contact customer support to have your account configured for this feature.

Fields Specific to the Visa Secure Use Case

These API fields are required specifically for this use case.

processingInformation.commerceIndicator	Set the value to `vbv` for a successful authentication (3-D Secure value of `05`), `vbv_attempted` if authentication was attempted but did not succeed (3-D Secure value of `06`), or `vbv_failure` if authentication failed (3-D Secure value of `07`).
consumerAuthenticationInformation.cavv	Required when payer authentication is successful.

Endpoint

Production: `POST https://api.cybersource.com/pts/v2/payments`
Test: `POST https://apitest.cybersource.com/pts/v2/payments`

Related Information

- *API field reference guide for the REST API*

Required Fields for Processing an Authorization Using Visa Secure

Use these required fields to process an authorization using Visa Secure.

Important

When relaxed requirements for address data and the expiration date are being used, not all fields in this list are required. It is your responsibility to determine whether your account is enabled to use this feature and which fields are required. Refer to the Payments guide for more information about relaxed requirements in payment transactions.

Required Fields

clientReferenceInformation.code	
consumerAuthenticationInformation.cavv	This field is required when payer authentication is successful. Otherwise, this field is optional.
consumerAuthenticationInformation.xid	
orderInformation.amountDetails.currency	
orderInformation.amountDetails.totalAmount	
orderInformation.billTo.address1	
orderInformation.billTo.administrativeArea	
orderInformation.billTo.country	
orderInformation.billTo.email	
orderInformation.billTo.firstName	
orderInformation.billTo.lastName	
orderInformation.billTo.locality	
orderInformation.billTo.postalCode	
paymentInformation.card.expirationMonth	
paymentInformation.card.expirationYear	
paymentInformation.card.number	
paymentInformation.card.type	
processingInformation.commerceIndicator	Set this field to one of these values: • `vbv`: Successful authentication (EMV 3-D Secure value of `05`). • `vbv_attempted`: Authentication was attempted (EMV 3-D Securevalue of `06`). • `vbv_failure`: or `internet`: Authentication failed or was not attempted (EMV 3-D Secure value of `07`).

Related Information

- *API field reference guide for the REST API*

REST Example: Validating and Authorizing a Transaction

Request

```
{
  "clientReferenceInformation": {
    "code": "TC50171_3"
  },
  "processingInformation": {
    "actionList": ["VALIDATE_CONSUMER_AUTHENTICATION"]
  },
  "paymentInformation": {
    "card": {
      "number": "4000000000001091",
      "expirationMonth": "01",
      "expirationYear": "2030"
    }
  },
  "orderInformation": {
    "billTo": {
      "firstName": "John",
      "lastName": "Smith",
      "address1": "201 S. Division St._1",
      "address2": "Suite 500",
      "locality": "Foster City",
      "administrativeArea": "CA",
      "postalCode": "94404",
      "country": "US",
      "email": "accept@cybs.com",
      "phoneNumber": "6504327113"
    }
  },
  "consumerAuthenticationInformation": {
    "authenticationTransactionId": "OiCtXA1j1AxtSNDh5lt1",
    "authenticationBrand": "VISA"
  }
}
```

Response to a Successful Request

```
{
  "_links": {
    "authReversal": {
      "method": "POST",
      "href": "/pts/v2/payments/6758954108726900304951/reversals"
    },
    "self": {
      "method": "GET",
      "href": "/pts/v2/payments/6758954108726900304951"
    },
    "capture": {
      "method": "POST",
      "href": "/pts/v2/payments/6758954108726900304951/captures"
    }
  },
  "clientReferenceInformation": {
    "code": "TC50171_3"
  },
  "id": "6758954108726900304951",
  "orderInformation": {
    "amountDetails": {
      "authorizedAmount": "100.00",
      "currency": "USD"
    }
  },
  "paymentAccountInformation": {
    "card": {
      "type": "001"
    }
  },
  "paymentInformation": {
    "tokenizedCard": {
      "type": "001"
    },
    "card": {
      "type": "001"
    }
  },
  "pointOfSaleInformation": {
    "terminalId": "111111"
  },
  "processorInformation": {
    "approvalCode": "888888",
    "networkTransactionId": "123456789619999",
    "transactionId": "123456789619999",
    "responseCode": "100",
    "avs": {
      "code": "X",
      "codeRaw": "I1"
    }
  },
  "reconciliationId": "711764833DU1FCQD",
  "status": "AUTHORIZED",
  "submitTimeUtc": "2023-02-08T22:30:11Z"
}
```

Processing Payments Using Credentials

This section provides the information you need in order to process payments using credentials.

Additional Resources for Credentialed Transactions

For more information, see these guides:

- *API field reference guide for the REST API*
- Github repositories: *https://github.com/Cybersource*

Customer-Initiated Transactions with Credentials on File

A customer-initiated transaction (CIT) is a transaction initiated by the customer. There are two types of CITs:

- Customer transactions during which the credentials are stored for future customer-initiated transactions.
- Customer transactions during which the credentials are stored for future merchant-initiated transactions.

Customers can initiate a CIT at a merchant payment terminal, through an online purchase transaction, or by making a purchase using a previously stored credential. When storing cardholder data for a CIT, you must also include 3-D Secure authentication credentials to ensure that the CIT can successfully process. Authentication credentials can be stored for future use with the card credentials by doing a non-payment authentication (NPA).

Business Center

You can create a new customer-initiated transaction in the Business Center by going to the One-Time Payments section and requesting a new authorization. When you have entered the customer's information, you can store the customer's credentials with the customer's permission in the Payment Information section. By doing so, you can perform merchant-initiated transactions for payments that the customer has pre-approved. For more information on how to perform a MIT in the Business Center, see *Merchant-Initiated No-Show Transactions with PAN* on page 119.

Storing Customer Credentials with a CIT and PAN

Before you can perform a merchant-initiated transaction (MIT) or a customer-initiated transaction (CIT) with credentials-on-file (COF), you must store the customer's credentials for later use. Further, before you can store the user's credentials, you must get the customer's consent to store their private information. This is also known as establishing a relationship with the customer.

Endpoint

Production: `POST https://api.cybersource.com/pts/v2/payments`
Test: `POST https://apitest.cybersource.com/pts/v2/payments`

Required Fields for Storing Customer Credentials During a CIT

Use these required fields for storing customer credentials during a customer-initiated transaction.

orderInformation.amountDetails.totalAmount	
orderInformation.billTo.address1	
orderInformation.billTo.administrativeArea	
orderInformation.billTo.country	
orderInformation.billTo.email	
orderInformation.billTo.firstName	
orderInformation.billTo.lastName	
orderInformation.billTo.locality	
orderInformation.billTo.phoneNumber	
orderInformation.billTo.postalCode	
paymentInformation.card.expirationMonth	
paymentInformation.card.expirationYear	
paymentInformation.card.number	
processingInformation.authorizationOptions. initiator. credentialStoredOnFile	**Set the value to `true`.**

REST Example: Storing Customer Credentials During a CIT

Request

```
{
   "processingInformation": {
      "authorizationOptions": {
         "initiator": {
            "credentialStoredOnFile": "true"
         }
      }
   },
   "orderInformation": {
      "billTo": {
         "firstName": "John",
         "lastName": "Doe",
         "address1": "201 S. Division St.",
         "postalCode": "48104-2201",
         "locality": "Ann Arbor",
         "administrativeArea": "MI",
         "country": "US",
         "email": "test@cybs.com",
         "phoneNumber": "5554327113"
      },
      "amountDetails": {
         "totalAmount": "100.00",
         "currency": "USD"
      }
   },
   "paymentInformation": {
      "card": {
         "expirationYear": "2031",
         "number": "4111xxxxxxxxxxxx",
         "expirationMonth": "12"
      }
   }
}
```

Response to a Successful Request

```
{
   "_links": {
      "authReversal": {
         "method": "POST",
         "href": "/pts/v2/payments/6528187198946076303004/reversals"
      },
      "self": {
         "method": "GET",
         "href": "/pts/v2/payments/6528187198946076303004"
      },
      "capture": {
         "method": "POST",
         "href": "/pts/v2/payments/6528187198946076303004/captures"
      }
   },
   "clientReferenceInformation": {
      "code": "1652818719876"
    },
    "id": "6528187198946076303004",
    "orderInformation": {
      "amountDetails": {
        "authorizedAmount": "100.00",
        "currency": "USD"
      }
    },
    "paymentAccountInformation": {
      "card": {
        "type": "001"
      }
    },
    "paymentInformation": {
      "tokenizedCard": {
        "type": "001"
      },
      "card": {
        "type": "001"
      }
    },
    "pointOfSaleInformation": {
      "terminalId": "111111"
    },
    "processorInformation": {
      "approvalCode": "888888",
      "networkTransactionId": "123456789619999",
      "transactionId": "123456789619999",
      "responseCode": "100",
      "avs": {
        "code": "X",
        "codeRaw": "I1"
      }
    },
    "reconciliationId": "63165088Z3AHV91G",
    "status": "AUTHORIZED",
    "submitTimeUtc": "2022-05-17T20:18:40Z"
}
```

Storing Customer Credentials with a CIT and TMS

Before you can perform a merchant-initiated transaction (MIT) or a customer-initiated transaction (CIT) with credentials-on-file (COF), you must get the customer's consent to store their payment credentials. This is also known as establishing a relationship with the customer. After you have their consent, you can store their payment credentials for later use.

Creating a TMS Token

When sending the initial CIT, you can create a TMS token to store the customer's credentials for the subsequent MITs. To create a TMS token, include the **processingInformation.actionTokenTypes** field in the authorization request. Set the field to one of these values based on the TMS token type you want to create:

Customer

Customer tokens store one or more customer payment instrument tokens and shipping address tokens.

Including a customer token in subsequent MITs eliminates the need to include billing information, card information, and the previous transaction's ID.

```
"processingInformation": {
  "actionTokenTypes": [
    "customer"
]
```

For more information about this TMS token type, see *Customer Tokens* in the Token Management Service Developer Guide.

Payment Instrument

Payment instrument tokens store an instrument identifier token, card information, and billing information. Payment instruments are not linked to a customer token. Including a payment instrument in subsequent MITs eliminates the need to include billing information, card information, and the previous transaction's ID.

```
"processingInformation": {
  "actionTokenTypes": [
    "paymentInstrument"
]
```

For more information about this TMS token type, see *Payment Instrument Token* in the Token Management Service Developer Guide.

Instrument Identifier

Instrument identifier tokens store a PAN. Including an instrument identifier in subsequent MITs eliminates the need to include a PAN and the previous transaction's ID.

```
"processingInformation": {
  "actionTokenTypes": [
    "instrumentIdentifier"
]
```

For more information about this TMS token type, see *Instrument Identifier Token* in

	the Token Management Service Developer Guide.
Instrument Identifier, Payment Instrument, and Customer Identifier	**You can also create multiple TMS token types in the same authorization. This example includes an instrument identifier, a payment instrument, and a customer token in the same authorization:** `"processingInformation": { "actionTokenTypes": [ "instrumentIdentifier", "paymentInstrument", "customer" ]`

```
"processingInformation": {
  "actionTokenTypes": [
     "instrumentIdentifier",
     "paymentInstrument",
     "customer"
]
```

Endpoint

Production: `POST https://api.cybersource.com/pts/v2/payments`
Test: `POST https://apitest.cybersource.com/pts/v2/payments`

Required Fields for Storing Customer Credentials with a CIT and TMS

orderInformation.amountDetails.currency	
orderInformation.amountDetails.totalAmount	
orderInformation.billTo.address1	
orderInformation.billTo.administrativeArea	
orderInformation.billTo.country	
orderInformation.billTo.email	
orderInformation.billTo.firstName	
orderInformation.billTo.lastName	
orderInformation.billTo.locality	
orderInformation.billTo.phoneNumber	
orderInformation.billTo.postalCode	
paymentInformation.card.expirationMonth	
paymentInformation.card.expirationYear	
paymentInformation.card.number	
processingInformation.actionList	**Set the value to**`TOKEN_CREATE`
processingInformation.actionTokenTypes	**Set to one or more of these values:** • `customer` • `instrumentIdentifier`

- `paymnentInstrument`

REST Example: Storing Customer Credentials with a CIT and TMS

Request

```
{
  "processingInformation": {
    "actionList": [
      "TOKEN_CREATE"
    ],
    "actionTokenTypes": [
      "instrumentIdentifier"
    ]
  },
  "paymentInformation": {
    "card": {
      "number": "4111111111111111",
      "expirationMonth": "12",
      "expirationYear": "2031",
      "securityCode": "123"
    }
  },
  "orderInformation": {
    "amountDetails": {
      "totalAmount": "102.21",
      "currency": "USD"
    },
    "billTo": {
      "firstName": "John",
      "lastName": "Doe",
      "address1": "1 Market St",
      "locality": "san francisco",
      "administrativeArea": "CA",
      "postalCode": "94105",
      "country": "US",
      "email": "test@cybs.com",
      "phoneNumber": "4158880000"
    }
  }
}
```

Response to a Successful Request

```
{
  "_links": {
    "authReversal": {
      "method": "POST",
      "href": "/pts/v2/payments/6972267090226779103955/reversals"
    },
    "self": {
      "method": "GET",
      "href": "/pts/v2/payments/6972267090226779103955"
    },
    "capture": {
      "method": "POST",
      "href": "/pts/v2/payments/6972267090226779103955/captures"
    }
  },
  "clientReferenceInformation": {
    "code": "TC50171_3"
  },
  "id": "6972267090226779103955",
  "orderInformation": {
    "amountDetails": {
      "authorizedAmount": "102.21",
      "currency": "USD"
    }
  },
  "paymentAccountInformation": {
    "card": {
      "type": "001"
    }
  },
  "paymentInformation": {
    "tokenizedCard": {
      "type": "001"
    },
    "card": {
      "type": "001"
    }
  },
  "pointOfSaleInformation": {
    "terminalId": "111111"
  },
  "processorInformation": {
    "paymentAccountReferenceNumber": "V0010013022298169667504231315",
    "approvalCode": "888888",
    "networkTransactionId": "123456789619999",
    "transactionId": "123456789619999",
    "responseCode": "100",
    "avs": {
      "code": "X",
      "codeRaw": "I1"
    }
  },
  "reconciliationId": "62506622XNMR6Q1Y",
  "status": "AUTHORIZED",
  "submitTimeUtc": "2023-10-13T19:51:49Z",
  "tokenInformation": {
    "instrumentidentifierNew": false,
    "instrumentIdentifier": {
      "state": "ACTIVE",
      "id": "7010000000016241111"
    }
  }
}
```

Retrieving Stored Customer Credentials During a CIT

After customers store their credentials on file, you can retrieve these credentials to use with subsequent transactions.

Endpoint

Production: `POST https://api.cybersource.com/pts/v2/payments`
Test: `POST https://apitest.cybersource.com/pts/v2/payments`

Required Fields for Retrieving Customer Credentials During a Customer-Initiated Transaction

Use these required fields to retrieve customer credentials during a customer-initiated transaction.

orderInformation.amountDetails.currency	
orderInformation.amountDetails.totalAmount	
orderInformation.billTo.address1	
orderInformation.billTo.administrativeArea	
orderInformation.billTo.country	
orderInformation.billTo.email	
orderInformation.billTo.firstName	
orderInformation.billTo.lastName	
orderInformation.billTo.locality	
orderInformation.billTo.phoneNumber	
orderInformation.billTo.postalCode	
paymentInformation.card.expirationMonth	
paymentInformation.card.expirationYear	
paymentInformation.card.number	
processingInformation.authorizationOptions.initiator. storedCredentialUsed	**Set field to `true`.**

Card-Specific Required Field for Retrieving Customer Credentials During a CIT

Discover

Discover requires the authorization amount from the original transaction in addition to the above required fields.

processingInformation.authorizationOptions.initiator.merchantInitiatedTransaction.originalAuthorizedAmount

REST Example: Retrieving Customer Credentials During a CIT

Request

```
{
   "processingInformation": {
      "authorizationOptions": {
         "initiator": {
            "storedCredentialUsed": "true"
         }
      }
   },
   "orderInformation": {
      "billTo": {
         "firstName": "John",
         "lastName": "Doe",
         "address1": "201 S. Division St.",
         "postalCode": "48104-2201",
         "locality": "Ann Arbor",
         "administrativeArea": "MI",
         "country": "US",
         "email": "test@cybs.com",
         "phoneNumber": "5554327113"
      },
      "amountDetails": {
         "totalAmount": "100.00",
         "currency": "USD",
         "originalAmount": "100"
           // Discover card Only
      }
   },
   "paymentInformation": {
      "card": {
         "expirationYear": "2031",
         "number": "4111xxxxxxxxxxxx",
         "expirationMonth": "12"
      }
   },
  "processorInformation": {
   "transactionId": "12345678961000"
  }
}
```

Response to a Successful Request

```
},
   "paymentAccountInformation": {
      "card": {
         "type": "002"
      }
   },
   "paymentInformation": {
      "tokenizedCard": {
         "type": "002"
      },
      "card": {
         "type": "002"
      }
   },
   "pointOfSaleInformation": {
      "terminalId": "111111"
   },
   "processorInformation": {
      "approvalCode": "888888",
      "authIndicator": "1",
      "networkTransactionId": "123456789619999",
      "transactionId": "123456789619999",
      "responseCode": "100",
      "avs": {
         "code": "X",
         "codeRaw": "I1"
      }
   },
   "reconciliationId": "63740353A3AJ2NSH",
   "status": "AUTHORIZED",
   "submitTimeUtc": "2022-05-20T19:13:06Z"
}
```

Delayed Transaction

Delayed charge transaction is performed to process a supplemental account charge after original services have been rendered and respective payment has been processed. This section describes how to process a merchant-initiated delayed transaction, also known as a delayed charge, using these payment types:

- *Merchant-Initiated Delayed Transaction with PAN* on page 107
- *Merchant-Initiated Delayed Transaction with TMS* on page 110

Merchant-Initiated Delayed Transaction with PAN

Delayed charge transaction is performed to process a supplemental account charge after original services have been rendered and respective payment has been processed.

Supported Card Types

These are the supported card types for processing credentialed transactions:

- Mastercard
- Visa

Endpoint

Production: `POST https://api.cybersource.com/pts/v2/payments`
Test: `POST https://apitest.cybersource.com/pts/v2/payments`

Required Fields for Processing a Merchant-Initiated Delayed Transaction

Use these required fields to process a merchant-initiated delayed transaction.

orderInformation.amountDetails.currency	
orderInformation.amountDetails.totalAmount	
orderInformation.billTo.address1	
orderInformation.billTo.administrativeArea	
orderInformation.billTo.country	
orderInformation.billTo.email	
orderInformation.billTo.firstName	
orderInformation.billTo.lastName	
orderInformation.billTo.locality	
orderInformation.billTo.phoneNumber	
orderInformation.billTo.postalCode	
paymentInformation.card.expirationMonth	
paymentInformation.card.expirationYear	
paymentInformation.card.number	
processingInformation. authorizationOptions.initiator. merchantInitiatedTransaction. previousTransactionId	• American Express: set to the transaction ID from the original transaction. • Discover: set to the transaction ID from the original transaction. • Visa: set to the last successful transaction ID.
processingInformation.authorizationOptions.initiator. merchantInitiatedTransaction.reason	**Set the value to** 2**.** **Required only for Discover, Mastercard, and Visa.**
processingInformation. authorizationOptions. initiator. type	**Set the value to** merchant**.**

Card-Specific Required Field for Processing a Merchant-Initiated Transactions

Discover

The listed card requires an additional field:

processingInformation. authorizationOptions. initiator. merchantInitiatedTransaction. originalAuthorizedAmount	**Provide the original transaction amount.**

REST Example: Processing a Merchant-Initiated Delayed Authorization Transaction

Request

```
{
   "orderInformation": {
  "billTo" : {
     "country" : "US",
     "lastName" : "Kim",
     "address1" : "201 S. Division St.",
     "postalCode" : "48104-2201",
     "locality" : "Ann Arbor",
     "administrativeArea" : "MI",
     "firstName" : "Kyong-Jin",
         "phoneNumber": "5554327113",
     "email" : "test@cybs.com"
    },
      "amountDetails": {
         "totalAmount": "120.00",
         "currency": "USD"
      }
   },
   "paymentInformation": {
      "card": {
         "expirationYear": "2031",
         "number": "4111xxxxxxxxxxxx",
         "expirationMonth": "12"
      }
   },
   "processingInformation": {
      "authorizationOptions": {
         "initiator": {
        "type": "merchant",
          "merchantInitiatedTransaction": {
          "originalAuthorizedAmount": "100",
             // Discover only
          "previousTransactionId": "123456789619999",
          "reason": "2"
          }
         }
      }
   }
}
```

Response to a Successful Request

```
{
   "_links": {
      "authReversal": {
         "method": "POST",
         "href": "/pts/v2/payments/6534213653516599003001/reversals"
      },
      "self": {
         "method": "GET",
         "href": "/pts/v2/payments/6534213653516599003001"
      },
      "capture": {
        "method": "POST",
        "href": "/pts/v2/payments/6534213653516599003001/captures"
      }
    },
    "clientReferenceInformation": {
      "code": "1653421365327"
    },
    "id": "6534213653516599003001",
    "orderInformation": {
      "amountDetails": {
        "authorizedAmount": "120.00",
        "currency": "USD"
      }
    },
    "paymentAccountInformation": {
      "card": {
        "type": "002"
      }
    },
    "paymentInformation": {
      "tokenizedCard": {
        "type": "002"
      },
      "card": {
        "type": "002"
      }
    },
    "pointOfSaleInformation": {
      "terminalId": "111111"
    },
    "processorInformation": {
      "approvalCode": "888888",
      "authIndicator": "1",
      "networkTransactionId": "123456789619999",
      "transactionId": "123456789619999",
      "responseCode": "100",
      "avs": {
        "code": "X",
        "codeRaw": "I1"
      }
    },
    "reconciliationId": "64365475T3K10Q1D",
    "status": "AUTHORIZED",
    "submitTimeUtc": "2022-05-24T19:42:45Z"
}
```

Merchant-Initiated Delayed Transaction with TMS

Delayed charge transaction is performed to process a supplemental account charge after original services have been rendered and respective payment has been processed.
This section describes how to process a merchant-initiated delayed transaction using these TMS token types:

Customer	Customer tokens store one or more customer payment instrument tokens and shipping address tokens.
	Including a customer token eliminates the need to include billing information, card information, and the previous transaction's ID.
	`"paymentInformation": {`
	` "customer": {`
	`  "id": "07C9CA98022DA498E063A2598D0AA400"`
	` }`
	`}`
	For more information about this TMS token type, see *Customer Tokens* in the Token Management Service Developer Guide.

Payment Instrument	Payment instrument tokens store an instrument identifier token, card information, and billing information. Payment instruments are not linked to a customer token.
	Including a payment instrument eliminates the need to include billing information, card information, and the previous transaction's ID.
	`"paymentInformation": {`
	` "paymentInstrument": {`
	`  "id": "07CA24EF20F9E2C9E063A2598D0A8565"`
	` }`
	`}`
	For more information about this TMS token type, see *Payment Instrument Token* in the Token Management Service Developer Guide.

Instrument Identifier	Instrument identifier tokens store only a PAN. Including an instrument identifier eliminates the need to include a PAN and the previous transaction's ID.
	`"paymentInformation": {`
	` "instrumentIdentifier": {`
	`  "id": "7010000000016241111"`
	` }`
	`}`

	For more information about this TMS token type, see *Instrument Identifier Token* in the Token Management Service Developer Guide.

Supported Card Types

These are the supported card types for processing credentialed transactions:

- Mastercard
- Visa

Endpoint

Production: `POST https://api.cybersource.com/pts/v2/payments`
Test: `POST https://apitest.cybersource.com/pts/v2/payments`

Required Fields for MIT Delayed Transaction with TMS

Include these Required Fields

orderInformation.amountDetails.currency	
orderInformation.amountDetails.totalAmount	
paymentInformation.[tokentype].id	Where **[tokentype]** is the TMS token type you are using: • **customer** • **instrumentIdentifier** • **paymentInstrument**
processingInformation. authorizationOptions. initiator. merchantInitiatedTransaction. reason	Set the value to `2`. Required only for Discover, Mastercard, and Visa.

Instrument Identifier Required Fields

If you are using the **paymentInformation.instrumentIdentifier.id** token, include these required fields in addition to the required fields listed above.

orderInformation.billTo.address1

orderInformation.billTo.administrativeArea

orderInformation.billTo.country

orderInformation.billTo.email

orderInformation.billTo.firstName

orderInformation.billTo.lastName

orderInformation.billTo.locality

orderInformation.billTo.phoneNumber

orderInformation.billTo.postalCode

paymentInformation.card.expirationMonth

paymentInformation.card.expirationYear

Card-Specific Fields

The listed card type requires an additional field.

Discover	**processingInformation.authorizationOptions.initiator. merchantInitiatedTransaction.originalAuthorizedAmou** **Set to the original transaction amount.**

Example: MIT Delayed Transaction with TMS Instrument Identifier

Request

```
{
  "processingInformation": {
    "authorizationOptions": {
      "initiator": {
        "merchantInitiatedTransaction": {
          "reason": "2"
        }
      }
    }
  },
  "paymentInformation": {
    "card": {
      "expirationMonth": "12",
      "expirationYear": "2031"
    },
    "instrumentIdentifier": {
      "id": "7010000000016241111"
    }
  },
  "orderInformation": {
    "amountDetails": {
      "totalAmount": "102.21",
      "currency": "USD"
    },
    "billTo": {
      "firstName": "John",
      "lastName": "Doe",
      "address1": "1 Market St",
      "locality": "san francisco",
      "administrativeArea": "CA",
      "postalCode": "94105",
      "country": "US",
      "email": "test@cybs.com",
      "phoneNumber": "4158880000"
    }
  }
}
```

Response to a Successful Request

```
{
  "_links": {
    "authReversal": {
      "method": "POST",
      "href": "/pts/v2/payments/6976922830456934003954/reversals"
    },
    "self": {
      "method": "GET",
      "href": "/pts/v2/payments/6976922830456934003954"
    },
    "capture": {
      "method": "POST",
      "href": "/pts/v2/payments/6976922830456934003954/captures"
    }
  },
  "clientReferenceInformation": {
    "code": "1697692283160"
  },
  "id": "6976922830456934003954",
  "orderInformation": {
    "amountDetails": {
      "authorizedAmount": "102.21",
      "currency": "USD"
    }
  },
  "paymentAccountInformation": {
    "card": {
      "type": "001"
    }
  },
  "paymentInformation": {
    "tokenizedCard": {
      "type": "001"
    },
    "instrumentIdentifier": {
      "id": "7010000000016241111",
      "state": "ACTIVE"
    },
    "card": {
      "type": "001"
    }
  },
  "pointOfSaleInformation": {
    "terminalId": "111111"
  },
  "processingInformation": {
    "paymentSolution": "015"
  },
  "processorInformation": {
    "paymentAccountReferenceNumber": "V0010013022298169667504231315",
    "approvalCode": "888888",
    "networkTransactionId": "123456789619999",
    "transactionId": "123456789619999",
    "responseCode": "100",
    "avs": {
      "code": "X",
      "codeRaw": "I1"
    }
  },
  "reconciliationId": "62700184NNMR6XFK",
  "status": "AUTHORIZED",
  "submitTimeUtc": "2023-10-19T05:11:23Z"
}
```

Example: MIT Delayed Transaction with TMS Payment Instrument

Request

```
{
  "processingInformation": {
    "authorizationOptions": {
      "initiator": {
        "merchantInitiatedTransaction": {
          "reason": "2"
        }
      }
    }
  },
  "paymentInformation": {
    "paymentInstrument": {
      "id": "080AE120369A7947E063A2598D0A718F"
    }
  },
  "orderInformation": {
    "amountDetails": {
      "totalAmount": "102.21",
      "currency": "USD"
    }
  }
}
```

Response to a Successful Request

```
{
  "_links": {
    "authReversal": {
      "method": "POST",
      "href": "/pts/v2/payments/6976917718796256603955/reversals"
    },
    "self": {
      "method": "GET",
      "href": "/pts/v2/payments/6976917718796256603955"
    },
    "capture": {
      "method": "POST",
      "href": "/pts/v2/payments/6976917718796256603955/captures"
    }
  },
  "clientReferenceInformation": {
    "code": "1697691771976"
  },
  "id": "6976917718796256603955",
  "orderInformation": {
    "amountDetails": {
      "authorizedAmount": "102.21",
      "currency": "USD"
    }
  },
  "paymentAccountInformation": {
    "card": {
      "type": "001"
    }
  },
  "paymentInformation": {
    "tokenizedCard": {
      "type": "001"
    },
    "instrumentIdentifier": {
      "id": "7010000000016241111",
      "state": "ACTIVE"
    },
    "paymentInstrument": {
      "id": "080AE120369A7947E063A2598D0A718F"
    },
    "card": {
      "type": "001"
    }
  },
  "pointOfSaleInformation": {
    "terminalId": "111111"
  },
  "processingInformation": {
    "paymentSolution": "015"
  },
  "processorInformation": {
    "paymentAccountReferenceNumber": "V0010013022298169667504231315",
    "approvalCode": "888888",
    "networkTransactionId": "123456789619999",
    "transactionId": "123456789619999",
    "responseCode": "100",
    "avs": {
      "code": "X",
      "codeRaw": "I1"
    }
  },
  "reconciliationId": "62700629BNN13VGW",
  "status": "AUTHORIZED",
  "submitTimeUtc": "2023-10-19T05:02:52Z"
}
```

Example: MIT Delayed Transaction with TMS Customer token

Request

```
{
  "processingInformation": {
    "authorizationOptions": {
      "initiator": {
        "merchantInitiatedTransaction": {
          "reason": "2"
        }
      }
    }
  },
  "paymentInformation": {
    "customer": {
      "id": "080AC9AB60C92AA2E063A2598D0A0C74"
    }
  },
  "orderInformation": {
    "amountDetails": {
      "totalAmount": "102.21",
      "currency": "USD"
    }
  }
}
```

Response to a Successful Request

```
{
  "_links": {
    "authReversal": {
      "method": "POST",
      "href": "/pts/v2/payments/6976916433716228003955/reversals"
    },
    "self": {
      "method": "GET",
      "href": "/pts/v2/payments/6976916433716228003955"
    },
    "capture": {
      "method": "POST",
      "href": "/pts/v2/payments/6976916433716228003955/captures"
    }
  },
  "clientReferenceInformation": {
    "code": "1697691643458"
  },
  "id": "6976916433716228003955",
  "orderInformation": {
    "amountDetails": {
      "authorizedAmount": "102.21",
      "currency": "USD"
    }
  },
  "paymentAccountInformation": {
    "card": {
      "type": "001"
    }
  },
  "paymentInformation": {
    "tokenizedCard": {
      "type": "001"
    },
    "instrumentIdentifier": {
      "id": "7010000000016241111",
      "state": "ACTIVE"
    },
    "paymentInstrument": {
      "id": "080AE6DB37B09557E063A2598D0AA4C9"
    },
    "card": {
      "type": "001"
    },
    "customer": {
      "id": "080AC9AB60C92AA2E063A2598D0A0C74"
    }
  },
  "pointOfSaleInformation": {
    "terminalId": "111111"
  },
  "processingInformation": {
    "paymentSolution": "015"
  },
  "processorInformation": {
    "paymentAccountReferenceNumber": "V0010013022298169667504231315",
    "approvalCode": "888888",
    "networkTransactionId": "123456789619999",
    "transactionId": "123456789619999",
    "responseCode": "100",
    "avs": {
      "code": "X",
      "codeRaw": "I1"
    }
  },
  "reconciliationId": "62700435FNN143RY",
  "status": "AUTHORIZED",
  "submitTimeUtc": "2023-10-19T05:00:43Z"
}
```

No-Show Transactions

A no-show authorization occurs when a merchant charges a customer after the customer makes a reservation, and does not show up to claim the reservation. In this situation, the customer is charged an agreed upon fee for not showing up as expected.
This section describes how to process a merchant-initiated no-show transaction using these payment types:

- *Merchant-Initiated No-Show Transactions with PAN* on page 119

- *Merchant-Initiated No-Show Transaction with TMS* on page 122

Merchant-Initiated No-Show Transactions with PAN

A no-show authorization occurs when a merchant charges a customer after the customer makes a reservation, and does not show up to claim the reservation. In this situation, the customer is charged an agreed upon fee for not showing up as expected.

Supported Card Types

These are the supported card types for processing credentialed transactions:

- Mastercard
- Visa

Endpoint

Production: `POST https://api.cybersource.com/pts/v2/payments`
Test: `POST https://apitest.cybersource.com/pts/v2/payments`

Required Fields for Processing Merchant-Initiated No-Show Charges

Use these required fields to process a merchant-initiated no-show charges transaction.

orderInformation.amountDetails.currency	
orderInformation.amountDetails.totalAmount	
orderInformation.billTo.address1	
orderInformation.billTo.administrativeArea	
orderInformation.billTo.country	
orderInformation.billTo.email	
orderInformation.billTo.firstName	
orderInformation.billTo.lastName	
orderInformation.billTo.locality	
orderInformation.billTo.phoneNumber	
orderInformation.billTo.postalCode	
paymentInformation.card.expirationMonth	
paymentInformation.card.expirationYear	
paymentInformation.card.number	
processingInformation. authorizationOptions. initiator. merchantInitiatedTransaction. previousTransactionId	• American Express: set to the transaction ID from the original transaction. • Discover: set to the transaction ID from the original transaction.

	• Visa: set to the last successful transaction ID.
processingInformation. authorizationOptions. initiator. merchantInitiatedTransaction. reason	**Set the value to `4`.** **Required only for Discover, Mastercard, and Visa.**
processingInformation. authorizationOptions. initiator. type	**Set the value to `merchant`.**

Card-Specific Required Field for Processing a Merchant-Initiated Transactions

Discover

The listed card requires an additional field:

processingInformation. authorizationOptions. initiator. merchantInitiatedTransaction. originalAuthorizedAmount	**Provide the original transaction amount.**

Optional Field for Processing Merchant-Initiated No-Show Charges

You can use these optional fields to include additional information when authorizing a request for an MIT no-show charge:

processingInformation. authorizationOptions. initiator. storedCredentialUsed	**If the payment information is COF information, set to `true`.**

REST Example: Processing Merchant-Initiated No-Show Transactions

Request

```
{
   "processingInformation": {
      "authorizationOptions": {
         "initiator": {
       "type": "merchant",
          "merchantInitiatedTransaction": {
           "originalAuthorizedAmount": "100", //Discover only
           "previousTransactionId": "123456789619999",
           "reason": "4"
          }
         }
      }
   },
   "orderInformation": {
  "billTo" : {
     "country" : "US",
     "lastName" : "Kim",
     "address1" : "201 S. Division St.",
     "postalCode" : "48104-2201",
     "locality" : "Ann Arbor",
    "administrativeArea" : "MI",
    "firstName" : "Kyong-Jin",
        "phoneNumber": "5554327113",
    "email" : "test@cybs.com"
   },
     "amountDetails": {
       "totalAmount": "150.00",
       "currency": "USD"
     }
   },
   "paymentInformation": {
     "card": {
       "expirationYear": "2031",
       "number": "4111xxxxxxxxxxxx",
       "expirationMonth": "12"
     }
   }
}
```

Response to a Successful Request

```
{
   "_links": {
     "authReversal": {
       "method": "POST",
       "href": "/pts/v2/payments/6534214295466223903006/reversals"
     },
     "self": {
       "method": "GET",
       "href": "/pts/v2/payments/6534214295466223903006"
     },
     "capture": {
       "method": "POST",
       "href": "/pts/v2/payments/6534214295466223903006/captures"
     }
   },
   "clientReferenceInformation": {
     "code": "1653421429522"
   },
   "id": "6534214295466223903006",
   "orderInformation": {
     "amountDetails": {
       "authorizedAmount": "150.00",
       "currency": "USD"
     }
   },
   "paymentAccountInformation": {
     "card": {
       "type": "001"
     }
   },
   "paymentInformation": {
     "tokenizedCard": {
       "type": "001"
     },
     "card": {
       "type": "001"
     }
   },
   "pointOfSaleInformation": {
     "terminalId": "111111"
   },
   "processorInformation": {
     "approvalCode": "888888",
     "networkTransactionId": "123456789619999",
     "transactionId": "123456789619999",
     "responseCode": "100",
     "avs": {
       "code": "X",
       "codeRaw": "I1"
     }
   },
   "reconciliationId": "64365823G3K7HFAM",
   "status": "AUTHORIZED",
   "submitTimeUtc": "2022-05-24T19:43:49Z"
}
```

Merchant-Initiated No-Show Transaction with TMS

A no-show authorization occurs when a merchant charges a customer after the customer makes a reservation, and does not show up to claim the reservation. In this situation, the customer is charged an agreed upon fee for not showing up as expected.
This section describes how to process a merchant-initiated no-show transaction using these TMS token types:

Customer	Customer tokens store one or more customer payment instrument tokens and shipping address tokens. Including a customer token eliminates the need to include billing information, card information, and the previous transaction's ID. `"paymentInformation": { "customer": { "id": "07C9CA98022DA498E063A2598D0AA400" } }` For more information about this TMS token type, see *Customer Tokens* in the Token Management Service Developer Guide.
Payment Instrument	Payment instrument tokens store an instrument identifier token, card information, and billing information. Payment instruments are not linked to a customer token.

Including a payment instrument eliminates the need to include billing information, card information, and the previous transaction's ID.

```
"paymentInformation": {
  "paymentInstrument": {
    "id": "07CA24EF20F9E2C9E063A2598D0A8565"
  }
}
```

For more information about this TMS token type, see *Payment Instrument Token* in the Token Management Service Developer Guide.

Instrument Identifier

Instrument identifier tokens store only a PAN. Including an instrument identifier eliminates the need to include a PAN and the previous transaction's ID.

```
"paymentInformation": {
  "instrumentIdentifier": {
    "id": "7010000000016241111"
  }
}
```

For more information about this TMS token type, see *Instrument Identifier Token* in the Token Management Service Developer Guide.

Supported Card Types

These are the supported card types for processing credentialed transactions:

- Mastercard
- Visa

Endpoint

Production: `POST https://api.cybersource.com/pts/v2/payments`
Test: `POST https://apitest.cybersource.com/pts/v2/payments`

Required Fields for MIT No-Show Transaction with TMS

Include these Required Fields

orderInformation.amountDetails.currency

orderInformation.amountDetails.totalAmount

paymentInformation.[tokentype].id	Where **[tokentype]** is the TMS token type you are using: • customer • instrumentIdentifier • paymentInstrument
processingInformation. authorizationOptions. initiator. merchantInitiatedTransaction.reason	Set the value to 4. Required only for Discover, Mastercard, and Visa.

Instrument Identifier Required Fields

If you are using the **paymentInformation.instrumentIdentifier.id** token, include these required fields in addition to the required fields listed above.

orderInformation.billTo.address1

orderInformation.billTo.administrativeArea

orderInformation.billTo.country

orderInformation.billTo.email

orderInformation.billTo.firstName

orderInformation.billTo.lastName

orderInformation.billTo.locality

orderInformation.billTo.phoneNumber

orderInformation.billTo.postalCode

paymentInformation.card.expirationMonth

paymentInformation.card.expirationYear

Card-Specific Fields

The listed card type requires an additional field.

Discover	**processingInformation.authorizationOptions.initiator. merchantInitiatedTransaction.originalAuthorizedAmou** **Set to the original transaction amount.**

Example: MIT No-Show Transaction with a TMS Instrument Identifier

Request

```
{
 "processingInformation": {
  "authorizationOptions": {
   "initiator": {
    "merchantInitiatedTransaction": {
     "reason": "4"
```

```
      }
    }
  }
},
"paymentInformation": {
  "card": {
    "expirationMonth": "12",
    "expirationYear": "2031"
  },
  "instrumentIdentifier": {
    "id": "7010000000016241111"
  }
},
"orderInformation": {
  "amountDetails": {
    "totalAmount": "102.21",
    "currency": "USD"
  },
  "billTo": {
    "firstName": "John",
    "lastName": "Doe",
    "address1": "1 Market St",
    "locality": "san francisco",
    "administrativeArea": "CA",
    "postalCode": "94105",
    "country": "US",
    "email": "test@cybs.com",
    "phoneNumber": "4158880000"
  }
}
}
```

Response to a Successful Request

```
{
  "_links": {
    "authReversal": {
      "method": "POST",
      "href": "/pts/v2/payments/6976922830456934003954/reversals"
    },
    "self": {
      "method": "GET",
      "href": "/pts/v2/payments/6976922830456934003954"
    },
    "capture": {
      "method": "POST",
      "href": "/pts/v2/payments/6976922830456934003954/captures"
    }
  },
  "clientReferenceInformation": {
    "code": "1697692283160"
  },
  "id": "6976922830456934003954",
  "orderInformation": {
    "amountDetails": {
      "authorizedAmount": "102.21",
    "currency": "USD"
   }
  },
  "paymentAccountInformation": {
   "card": {
    "type": "001"
   }
  },
  "paymentInformation": {
   "tokenizedCard": {
    "type": "001"
   },
   "instrumentIdentifier": {
    "id": "7010000000016241111",
    "state": "ACTIVE"
   },
   "card": {
    "type": "001"
   }
  },
  "pointOfSaleInformation": {
   "terminalId": "111111"
  },
  "processingInformation": {
   "paymentSolution": "015"
  },
  "processorInformation": {
   "paymentAccountReferenceNumber": "V0010013022298169667504231315",
   "approvalCode": "888888",
   "networkTransactionId": "123456789619999",
   "transactionId": "123456789619999",
   "responseCode": "100",
   "avs": {
    "code": "X",
    "codeRaw": "I1"
   }
  },
  "reconciliationId": "62700184NNMR6XFK",
  "status": "AUTHORIZED",
  "submitTimeUtc": "2023-10-19T05:11:23Z"
 }
```

Example: MIT No-Show Transaction with a TMS Payment Instrument

Request

```
{
  "processingInformation": {
   "authorizationOptions": {
    "initiator": {
     "merchantInitiatedTransaction": {
      "reason": "4"
     }
    }
   }
  },
  "paymentInformation": {
    "paymentInstrument": {
      "id": "080AE120369A7947E063A2598D0A718F"
    }
  },
  "orderInformation": {
    "amountDetails": {
      "totalAmount": "102.21",
      "currency": "USD"
    }
  }
}
```

Response to a Successful Request

```
{
  "_links": {
    "authReversal": {
      "method": "POST",
      "href": "/pts/v2/payments/6976917718796256603955/reversals"
    },
    "self": {
      "method": "GET",
      "href": "/pts/v2/payments/6976917718796256603955"
    },
    "capture": {
      "method": "POST",
      "href": "/pts/v2/payments/6976917718796256603955/captures"
    }
  },
  "clientReferenceInformation": {
    "code": "1697691771976"
  },
  "id": "6976917718796256603955",
  "orderInformation": {
    "amountDetails": {
      "authorizedAmount": "102.21",
      "currency": "USD"
    }
  },
  "paymentAccountInformation": {
    "card": {
      "type": "001"
    }
  },
  "paymentInformation": {
    "tokenizedCard": {
      "type": "001"
    },
    "instrumentIdentifier": {
      "id": "7010000000016241111",
      "state": "ACTIVE"
    },
    "paymentInstrument": {
      "id": "080AE120369A7947E063A2598D0A718F"
    },
    "card": {
      "type": "001"
    }
  },
  "pointOfSaleInformation": {
    "terminalId": "111111"
  },
  "processingInformation": {
    "paymentSolution": "015"
  },
  "processorInformation": {
    "paymentAccountReferenceNumber": "V0010013022298169667504231315",
    "approvalCode": "888888",
    "networkTransactionId": "123456789619999",
    "transactionId": "123456789619999",
    "responseCode": "100",
    "avs": {
      "code": "X",
      "codeRaw": "I1"
    }
  },
  "reconciliationId": "62700629BNN13VGW",
  "status": "AUTHORIZED",
  "submitTimeUtc": "2023-10-19T05:02:52Z"
}
```

Example: MIT No-Show Transaction with a TMS Customer

Request

```
{
  "processingInformation": {
    "authorizationOptions": {
      "initiator": {
        "merchantInitiatedTransaction": {
          "reason": "4"
        }
      }
    }
  },
  "paymentInformation": {
    "customer": {
      "id": "080AC9AB60C92AA2E063A2598D0A0C74"
    }
  },
  "orderInformation": {
    "amountDetails": {
      "totalAmount": "102.21",
      "currency": "USD"
    }
  }
}
```

Response to a Successful Request

```
{
  "_links": {
    "authReversal": {
      "method": "POST",
      "href": "/pts/v2/payments/6976916433716228003955/reversals"
    },
    "self": {
      "method": "GET",
      "href": "/pts/v2/payments/6976916433716228003955"
    },
    "capture": {
      "method": "POST",
      "href": "/pts/v2/payments/6976916433716228003955/captures"
    }
  },
  "clientReferenceInformation": {
    "code": "1697691643458"
  },
  "id": "6976916433716228003955",
  "orderInformation": {
    "amountDetails": {
      "authorizedAmount": "102.21",
      "currency": "USD"
    }
  },
  "paymentAccountInformation": {
    "card": {
      "type": "001"
    }
  },
  "paymentInformation": {
    "tokenizedCard": {
      "type": "001"
    },
    "instrumentIdentifier": {
      "id": "7010000000016241111",
      "state": "ACTIVE"
    },
    "paymentInstrument": {
      "id": "080AE6DB37B09557E063A2598D0AA4C9"
    },
    "card": {
      "type": "001"
    },
    "customer": {
      "id": "080AC9AB60C92AA2E063A2598D0A0C74"
    }
  },
  "pointOfSaleInformation": {
    "terminalId": "111111"
  },
  "processingInformation": {
    "paymentSolution": "015"
  },
  "processorInformation": {
    "paymentAccountReferenceNumber": "V0010013022298169667504231315",
    "approvalCode": "888888",
    "networkTransactionId": "123456789619999",
    "transactionId": "123456789619999",
    "responseCode": "100",
    "avs": {
      "code": "X",
      "codeRaw": "I1"
    }
  },
  "reconciliationId": "62700435FNN143RY",
  "status": "AUTHORIZED",
  "submitTimeUtc": "2023-10-19T05:00:43Z"
}
```

Reauthorization Transaction

A reauthorization occurs when the completion or fulfillment of the original order or service extends beyond the authorized amount time limit. There are two common reauthorization scenarios:

- Split or delayed shipments by a retailer
- Extended car rentals, hotel stays, or cruise line bookings

This section describes how to process a reauthorization transaction using these payment methods:

- *Merchant-Initiated Reauthorization Transactions with PAN* on page 130
- *Merchant-Initiated Reauthorization Transactions with TMS* on page 133

Merchant-Initiated Reauthorization Transactions with PAN

A reauthorization occurs when the completion or fulfillment of the original order or service extends beyond the authorized amount time limit. There are two common reauthorization scenarios:

- Split or delayed shipments by a retailer
- Extended car rentals, hotel stays, or cruise line bookings

Supported Card Types

These are the supported card types for processing credentialed transactions:

- Mastercard
- Visa

Endpoint

Production: `POST https://api.cybersource.com/pts/v2/payments`
Test: `POST https://apitest.cybersource.com/pts/v2/payments`

Required Fields for Processing Merchant-Initiated Reauthorized Transactions

Use these required fields to process a merchant-initiated reauthorization transaction.

orderInformation.amountDetails.currency

orderInformation.amountDetails.totalAmount

orderInformation.billTo.address1

orderInformation.billTo.administrativeArea

orderInformation.billTo.country

orderInformation.billTo.email

orderInformation.billTo.firstName

orderInformation.billTo.lastName

orderInformation.billTo.locality

orderInformation.billTo.phoneNumber

orderInformation.billTo.postalCode

paymentInformation.card.expirationMonth

paymentInformation.card.expirationYear

paymentInformation.card.number

processingInformation. authorizationOptions. initiator. merchantInitiatedTransaction. previousTransactionId	• American Express: set to the transaction ID from the original transaction. • Discover: set to the transaction ID from the original transaction. • Visa: set to the last successful transaction ID.
processingInformation. authorizationOptions. initiator. merchantInitiatedTransaction. reason	**Set the value to** 3**.** **Required only for Discover and Visa.**
processingInformation. authorizationOptions. initiator. type	**Set the value to** merchant**.**

Card-Specific Required Field for Processing a Merchant-Initiated Transactions

Discover

The listed card requires an additional field:

processingInformation. authorizationOptions. initiator. merchantInitiatedTransaction. originalAuthorizedAmount	**Provide the original transaction amount.**

REST Example: Processing a Merchant-Initiated Reauthorized Transaction

Request

```
{
   "processingInformation": {
      "authorizationOptions": {
         "initiator": {
        "type": "merchant",
          "merchantInitiatedTransaction": {
           "originalAuthorizedAmount": "100", // Discover Only
           "previousTransactionId": "123456789619999",
           "reason": "3"
          }
         }
      }
   },
   "orderInformation": {
 "billTo" : {
    "country" : "US",
    "lastName" : "Kim",
    "address1" : "201 S. Division St.",
    "postalCode" : "48104-2201",
    "locality" : "Ann Arbor",
    "administrativeArea" : "MI",
    "firstName" : "Kyong-Jin",
         "phoneNumber": "5554327113",
    "email" : "test@cybs.com"
   },
      "amountDetails": {
         "totalAmount": "130.00",
         "currency": "USD"
      }
   },
   "paymentInformation": {
      "card": {
         "expirationYear": "2031",
         "number": "4111xxxxxxxxxxxx",
         "expirationMonth": "12"
      }
   }
}
```

Response to a Successful Request

```
{
   "_links": {
      "authReversal": {
         "method": "POST",
         "href": "/pts/v2/payments/6541178668686490403003/reversals"
      },
      "self": {
         "method": "GET",
         "href": "/pts/v2/payments/6541178668686490403003"
      },
      "capture": {
            "method": "POST",
            "href": "/pts/v2/payments/6541178668686490403003/captures"
        }
    },
    "clientReferenceInformation": {
        "code": "1654117866849"
    },
    "id": "6541178668686490403003",
    "orderInformation": {
        "amountDetails": {
            "authorizedAmount": "130.00",
            "currency": "USD"
        }
    },
    "paymentAccountInformation": {
        "card": {
            "type": "001"
        }
    },
    "paymentInformation": {
        "tokenizedCard": {
            "type": "001"
        },
        "card": {
            "type": "001"
        }
    },
    "pointOfSaleInformation": {
        "terminalId": "111111"
    },
    "processorInformation": {
        "approvalCode": "888888",
        "networkTransactionId": "123456789619999",
        "transactionId": "123456789619999",
        "responseCode": "100",
        "avs": {
            "code": "X",
            "codeRaw": "I1"
        }
    },
    "reconciliationId": "65313868D3TXXC05",
    "status": "AUTHORIZED",
    "submitTimeUtc": "2022-06-01T21:11:06Z"
}
```

Merchant-Initiated Reauthorization Transactions with TMS

A reauthorization occurs when the completion or fulfillment of the original order or service extends beyond the authorized amount time limit. There are two common reauthorization scenarios:

- Split or delayed shipments by a retailer
- Extended car rentals, hotel stays, or cruise line bookings

This section describes how to process a merchant-initiated reauthorization transactions using one or more TMS token types:

Customer	Customer tokens store one or more customer payment instrument tokens and shipping address tokens.
	Including a customer token eliminates the need to include billing information, card information, and the previous transaction's ID.
	`"paymentInformation": {`
	`  "customer": {`
	`    "id": "07C9CA98022DA498E063A2598D0AA400"`
	`  }`
	`}`
	For more information about this TMS token type, see *Customer Tokens* in the Token Management Service Developer Guide.

Payment Instrument	Payment instrument tokens store an instrument identifier token, card information, and billing information. Payment instruments are not linked to a customer token.
	Including a payment instrument eliminates the need to include billing information, card information, and the previous transaction's ID.
	`"paymentInformation": {`
	`  "paymentInstrument": {`
	`    "id": "07CA24EF20F9E2C9E063A2598D0A8565"`
	`  }`
	`}`
	For more information about this TMS token type, see *Payment Instrument Token* in the Token Management Service Developer Guide.

Instrument Identifier	Instrument identifier tokens store only a PAN. Including an instrument identifier eliminates the need to include a PAN and the previous transaction's ID.
	`"paymentInformation": {`
	`  "instrumentIdentifier": {`
	`    "id": "7010000000016241111"`
	`  }`

```
}
```

For more information about this TMS token type, see *Instrument Identifier Token* in the Token Management Service Developer Guide.

Supported Card Types

These are the supported card types for processing credentialed transactions:

- Mastercard
- Visa

Endpoint

Production: `POST https://api.cybersource.com/pts/v2/payments`
Test: `POST https://apitest.cybersource.com/pts/v2/payments`

Required Fields for MIT Reauthorization Transaction with TMS

Include these Required Fields

orderInformation.amountDetails.currency	
orderInformation.amountDetails.totalAmount	
paymentInformation.[tokentype].id	Where **[tokentype]** is the TMS token type you are using: • **customer** • **instrumentIdentifier** • **paymentInstrument**
processingInformation. authorizationOptions. initiator. merchantInitiatedTransaction. reason	Set the value to `3`. Required only for Discover and Visa.

Instrument Identifier Required Fields

If you are using the **paymentInformation.instrumentIdentifier.id** token, include these required fields in addition to the required fields listed above.

orderInformation.billTo.address1

orderInformation.billTo.administrativeArea

orderInformation.billTo.country

orderInformation.billTo.email

orderInformation.billTo.firstName

orderInformation.billTo.lastName

orderInformation.billTo.locality

orderInformation.billTo.phoneNumber

orderInformation.billTo.postalCode

paymentInformation.card.expirationMonth

paymentInformation.card.expirationYear

Card-Specific Fields

The listed card type requires an additional field.

Discover	**processingInformation.authorizationOptions.initiator. merchantInitiatedTransaction.originalAuthorizedAmou** **Set to the original transaction amount.**

Example: MIT Reauthorization Transaction with a TMS Instrument Identifier

Request

```
{
  "processingInformation": {
    "authorizationOptions": {
      "initiator": {
        "merchantInitiatedTransaction": {
          "reason": "3"
        }
      }
    }
  },
  "paymentInformation": {
    "card": {
      "expirationMonth": "12",
      "expirationYear": "2031"
    },
    "instrumentIdentifier": {
      "id": "7010000000016241111"
    }
  },
  "orderInformation": {
    "amountDetails": {
      "totalAmount": "102.21",
      "currency": "USD"
    },
    "billTo": {
      "firstName": "John",
      "lastName": "Doe",
      "address1": "1 Market St",
      "locality": "san francisco",
      "administrativeArea": "CA",
      "postalCode": "94105",
      "country": "US",
      "email": "test@cybs.com",
      "phoneNumber": "4158880000"
    }
  }
}
```

Response to a Successful Request

```
{
  "_links": {
    "authReversal": {
      "method": "POST",
      "href": "/pts/v2/payments/6976922830456934003954/reversals"
    },
    "self": {
      "method": "GET",
      "href": "/pts/v2/payments/6976922830456934003954"
    },
    "capture": {
      "method": "POST",
      "href": "/pts/v2/payments/6976922830456934003954/captures"
    }
  },
  "clientReferenceInformation": {
    "code": "1697692283160"
  },
  "id": "6976922830456934003954",
  "orderInformation": {
    "amountDetails": {
      "authorizedAmount": "102.21",
      "currency": "USD"
    }
  },
  "paymentAccountInformation": {
    "card": {
      "type": "001"
    }
  },
  "paymentInformation": {
    "tokenizedCard": {
      "type": "001"
    },
    "instrumentIdentifier": {
      "id": "7010000000016241111",
      "state": "ACTIVE"
    },
    "card": {
      "type": "001"
    }
  },
  "pointOfSaleInformation": {
    "terminalId": "111111"
  },
  "processingInformation": {
    "paymentSolution": "015"
  },
  "processorInformation": {
    "paymentAccountReferenceNumber": "V0010013022298169667504231315",
    "approvalCode": "888888",
    "networkTransactionId": "123456789619999",
    "transactionId": "123456789619999",
    "responseCode": "100",
    "avs": {
      "code": "X",
      "codeRaw": "I1"
    }
  },
  "reconciliationId": "62700184NNMR6XFK",
  "status": "AUTHORIZED",
  "submitTimeUtc": "2023-10-19T05:11:23Z"
}
```

Example: MIT Reauthorization Transaction with a TMS Payment Instrument

Request

```
{
  "processingInformation": {
    "authorizationOptions": {
      "initiator": {
        "merchantInitiatedTransaction": {
          "reason": "3"
        }
      }
    }
  },
  "paymentInformation": {
    "paymentInstrument": {
      "id": "080AE120369A7947E063A2598D0A718F"
    }
  },
  "orderInformation": {
    "amountDetails": {
      "totalAmount": "102.21",
      "currency": "USD"
    }
  }
}
```

Response to a Successful Request

```
{
  "_links": {
    "authReversal": {
      "method": "POST",
      "href": "/pts/v2/payments/6976917718796256603955/reversals"
    },
    "self": {
      "method": "GET",
      "href": "/pts/v2/payments/6976917718796256603955"
    },
    "capture": {
      "method": "POST",
      "href": "/pts/v2/payments/6976917718796256603955/captures"
    }
  },
  "clientReferenceInformation": {
    "code": "1697691771976"
  },
  "id": "6976917718796256603955",
  "orderInformation": {
    "amountDetails": {
      "authorizedAmount": "102.21",
      "currency": "USD"
    }
  },
  "paymentAccountInformation": {
    "card": {
      "type": "001"
    }
  },
  "paymentInformation": {
    "tokenizedCard": {
      "type": "001"
    },
    "instrumentIdentifier": {
      "id": "7010000000016241111",
      "state": "ACTIVE"
    },
    "paymentInstrument": {
      "id": "080AE120369A7947E063A2598D0A718F"
    },
    "card": {
      "type": "001"
    }
  },
  "pointOfSaleInformation": {
    "terminalId": "111111"
  },
  "processingInformation": {
    "paymentSolution": "015"
  },
  "processorInformation": {
    "paymentAccountReferenceNumber": "V0010013022298169667504231315",
    "approvalCode": "888888",
    "networkTransactionId": "123456789619999",
    "transactionId": "123456789619999",
    "responseCode": "100",
    "avs": {
      "code": "X",
      "codeRaw": "I1"
    }
  },
  "reconciliationId": "62700629BNN13VGW",
  "status": "AUTHORIZED",
  "submitTimeUtc": "2023-10-19T05:02:52Z"
}
```

Example: MIT Reauthorization Transaction with a TMS Customer

Request

```
{
  "processingInformation": {
    "authorizationOptions": {
      "initiator": {
        "merchantInitiatedTransaction": {
          "reason": "3"
        }
      }
    }
  },
  "paymentInformation": {
    "customer": {
      "id": "080AC9AB60C92AA2E063A2598D0A0C74"
    }
  },
  "orderInformation": {
    "amountDetails": {
      "totalAmount": "102.21",
      "currency": "USD"
    }
  }
}
```

Response to a Successful Request

```
{
  "_links": {
    "authReversal": {
      "method": "POST",
      "href": "/pts/v2/payments/6976916433716228003955/reversals"
    },
    "self": {
      "method": "GET",
      "href": "/pts/v2/payments/6976916433716228003955"
    },
    "capture": {
      "method": "POST",
      "href": "/pts/v2/payments/6976916433716228003955/captures"
    }
  },
  "clientReferenceInformation": {
    "code": "1697691643458"
  },
  "id": "6976916433716228003955",
  "orderInformation": {
    "amountDetails": {
      "authorizedAmount": "102.21",
      "currency": "USD"
    }
  },
  "paymentAccountInformation": {
    "card": {
```

```
    "type": "001"
   }
  },
  "paymentInformation": {
   "tokenizedCard": {
    "type": "001"
   },
   "instrumentIdentifier": {
    "id": "7010000000016241111",
    "state": "ACTIVE"
   },
   "paymentInstrument": {
    "id": "080AE6DB37B09557E063A2598D0AA4C9"
   },
   "card": {
    "type": "001"
   },
   "customer": {
    "id": "080AC9AB60C92AA2E063A2598D0A0C74"
   }
  },
  "pointOfSaleInformation": {
   "terminalId": "111111"
  },
  "processingInformation": {
   "paymentSolution": "015"
  },
  "processorInformation": {
   "paymentAccountReferenceNumber": "V0010013022298169667504231315",
   "approvalCode": "888888",
   "networkTransactionId": "123456789619999",
   "transactionId": "123456789619999",
   "responseCode": "100",
   "avs": {
    "code": "X",
    "codeRaw": "I1"
   }
  },
  "reconciliationId": "62700435FNN143RY",
  "status": "AUTHORIZED",
  "submitTimeUtc": "2023-10-19T05:00:43Z"
}
```

Recurring Payments

A recurring payment is a credentials-on-file (COF) transaction in a series of payments that you bill to a customer for a fixed amount at regular intervals that do not exceed one year between transactions. The series of recurring payments is the result of an agreement between you and the customer for the purchase of goods or services that are provided at regular intervals. Recurring payments are also known as subscriptions.

Mastercard uses standing order and subscription payments instead of recurring payments. See *Mastercard Standing Order Payments* on page 167 and *Mastercard Subscription Payments* on page 173.

Recurring Billing Service for Recurring Payments

Important

Do not use this document for the Recurring Billing service.
Use the *Recurring Billing Developer Guide*. When you use the Recurring Billing service, Cybersource saves and stores payment credentials for recurring transactions, ensuring compliance with COF best practices.

Customer-Initiated Recurring Payment with PAN

A recurring payment is a credentials-on-file (COF) transaction in a series of payments that you bill to a customer at a fixed amount, at regular intervals that do not exceed one year between transactions. The series of recurring payments is the result of an agreement between you and the customer for the purchase of goods or services that are provided at regular intervals.

Supported Card Types

These are the supported card types for processing credentialed transactions:

- Visa

Mastercard uses standing order and subscription payments instead of recurring payments. See *Mastercard Standing Order Payments* on page 167 and *Mastercard Subscription Payments* on page 173.

Recurring Billing Service for Recurring Payments

Important

Do not use this document for the Recurring Billing service.
Use the *Recurring Billing Developer Guide*. When you use the Recurring Billing service, Cybersource saves and stores payment credentials for recurring transactions, ensuring compliance with COF best practices.

Address Verification Service for Recurring Payments

If your processor supports the Address Verification Service (AVS), then the AVS should verify every authorization request. Cybersource recommends checking the AVS's results for the first recurring payment to ensure that the payment information is accurate and to reduce the risk of fraud.

You must determine how to handle the AVS results for any subsequent recurring payments that are not the same as the already-verified billing address information from the first recurring payment.

Endpoint

Production: `POST https://api.cybersource.com/pts/v2/payments`
Test: `POST https://apitest.cybersource.com/pts/v2/payments`

Successful Response

You must store the network transaction ID from the successful response message to include in subsequent MIT authorization requests in order to associate the CIT to the MIT. The network transaction ID is the **processorInformation.networkTransactionId** field value. Store the network transaction ID, which is the **processorInformation.networkTransactionId** field value, from the successful response message. You must include the network transaction ID in subsequent MIT authorization requests in order to associate the CIT to the MIT.

Required Fields for Authorizing a Customer-Initiated Recurring Payment with PAN

Use these required fields to request an initial customer-initiated recurring payment.

orderInformation.amountDetails.currency	
orderInformation.amountDetails.totalAmount	
orderInformation.billTo.address1	
orderInformation.billTo.administrativeArea	
orderInformation.billTo.country	
orderInformation.billTo.email	
orderInformation.billTo.firstName	
orderInformation.billTo.lastName	
orderInformation.billTo.locality	
orderInformation.billTo.postalCode	
paymentInformation.card.expirationMonth	
paymentInformation.card.expirationYear	
paymentInformation.card.number	
processingInformation. authorizationOptions. initiator. credentialStoredOnFile	**Set the value to `true`.**
processingInformation. authorizationOptions. initiator. type	**Set the value to `customer`.**

processingInformation. commerceIndicator	**Set the value to `internet`, a payer authentication value, or `MOTO`.**
processingInformation. recurringOptions. firstRecurringPayment	**Set the value to `true`.**

REST Example: Authorizing a Customer-Initiated Recurring Payment with a PAN

Request

```
{
  "processingInformation": {
    "commerceIndicator": "internet",
    "authorizationOptions": {
      "initiator": {
        "credentialStoredOnFile": "true",
        "type": "customer"
      }
    }
  },
  "orderInformation": {
    "billTo": {
      "firstName": "John",
      "lastName": "Doe",
      "address1": "201 S. Division St.",
      "postalCode": "48104-2201",
      "locality": "Ann Arbor",
      "administrativeArea": "MI",
      "country": "US",
      "phoneNumber": "5554327113",
      "email": "test@cybs.com"
    },
    "amountDetails": {
      "totalAmount": "100.00",
      "currency": "USD"
    }
  },
  "paymentInformation": {
    "card": {
      "expirationYear": "2031",
      "number": "4111xxxxxxxxxxxx",
      "expirationMonth": "12"
    }
  }
}
```

Response to a Successful Request

```
{
  "_links": {
    "authReversal": {
      "method": "POST",
      "href": "/pts/v2/payments/6528187198946076303004/reversals"
    },
    "self": {
      "method": "GET",
      "href": "/pts/v2/payments/6528187198946076303004"
    },
    "capture": {
      "method": "POST",
      "href": "/pts/v2/payments/6528187198946076303004/captures"
    }
  },
  "clientReferenceInformation": {
    "code": "1652818719876"
  },
  "id": "6528187198946076303004",
  "orderInformation": {
    "amountDetails": {
      "authorizedAmount": "100.00",
      "currency": "USD"
    }
  },
  "paymentAccountInformation": {
    "card": {
      "type": "001"
    }
  },
  "paymentInformation": {
    "tokenizedCard": {
      "type": "001"
    },
    "card": {
      "type": "001"
    }
  },
  "pointOfSaleInformation": {
    "terminalId": "111111"
  },
  "processorInformation": {
    "approvalCode": "888888",
    "networkTransactionId": "123456789619999",
    "transactionId": "123456789619999",
    "responseCode": "100",
    "avs": {
      "code": "X",
      "codeRaw": "I1"
    }
  },
  "reconciliationId": "63165088Z3AHV91G",
  "status": "AUTHORIZED",
  "submitTimeUtc": "2022-05-17T20:18:40Z"
}
```

Customer-Initiated Recurring Payment with TMS

A recurring payment is a credentials-on-file (COF) transaction in a series of payments that you bill to a customer at a fixed amount, at regular intervals that do not exceed one year between transactions. The series of recurring payments is the result of an agreement between you and the customer for the purchase of goods or services that are provided at regular intervals.

Supported Card Types

These are the supported card types for processing credentialed transactions:

- Visa

Mastercard uses standing order and subscription payments instead of recurring payments. See *Mastercard Standing Order Payments* on page 167 and *Mastercard Subscription Payments* on page 173.

Recurring Billing Service for Recurring Payments

Important

Do not use this document for the Recurring Billing service.
Use the *Recurring Billing Developer Guide*. When you use the Recurring Billing service, Cybersource saves and stores payment credentials for recurring transactions, ensuring compliance with COF best practices.

Creating a TMS Token

When sending the initial CIT, you can create a TMS token to store the customer's credentials for the subsequent MITs. To create a TMS token, include the **processingInformation.actionTokenTypes** field in the authorization request. Set the field to one of these values based on the TMS token type you want to create:

Customer

Customer tokens store one or more customer payment instrument tokens and shipping address tokens.

Including a customer token in subsequent MITs eliminates the need to include billing information, card information, and the previous transaction's ID.

```
"processingInformation": {
  "actionTokenTypes": [
   "customer"
]
```

For more information about this TMS token type, see *Customer Tokens* in the Token Management Service Developer Guide.

Payment Instrument

Payment instrument tokens store an instrument identifier token, card information, and billing information. Payment instruments are not linked to

a customer token. Including a payment instrument in subsequent MITs eliminates the need to include billing information, card information, and the previous transaction's ID.

```
"processingInformation": {
  "actionTokenTypes": [
    "paymentInstrument"
]
```

For more information about this TMS token type, see *Payment Instrument Token* in the Token Management Service Developer Guide.

Instrument Identifier

Instrument identifier tokens store a PAN. Including an instrument identifier in subsequent MITs eliminates the need to include a PAN and the previous transaction's ID.

```
"processingInformation": {
  "actionTokenTypes": [
    "instrumentIdentifier"
]
```

For more information about this TMS token type, see *Instrument Identifier Token* in the Token Management Service Developer Guide.

Instrument Identifier, Payment Instrument, and Customer Identifier

You can also create multiple TMS token types in the same authorization. This example includes an instrument identifier, a payment instrument, and a customer token in the same authorization:

```
 "processingInformation": {
  "actionTokenTypes": [
     "instrumentIdentifier",
     "paymentInstrument",
     "customer"
]
```

Address Verification Service for Recurring Payments

If your processor supports the Address Verification Service (AVS), then the AVS should verify every authorization request. Cybersource recommends checking the AVS's results

for the first recurring payment to ensure that the payment information is accurate and to reduce the risk of fraud.
You must determine how to handle the AVS results for any subsequent recurring payments that are not the same as the already-verified billing address information from the first recurring payment.

Endpoint

Production: `POST https://api.cybersource.com/pts/v2/payments`
Test: `POST https://apitest.cybersource.com/pts/v2/payments`

Required Fields for Authorizing a Customer-Initiated Recurring Payment with TMS

Use these required fields to request a customer-initiated recurring payment with TMS.

orderInformation.amountDetails.currency	
orderInformation.amountDetails.totalAmount	
orderInformation.billTo.address1	
orderInformation.billTo.administrativeArea	
orderInformation.billTo.country	
orderInformation.billTo.email	
orderInformation.billTo.firstName	
orderInformation.billTo.lastName	
orderInformation.billTo.locality	
orderInformation.billTo.postalCode	
paymentInformation.card.expirationMonth	
paymentInformation.card.expirationYear	
paymentInformation.card.number	
processingInformation.actionList	**Set the value to `TOKEN_CREATE`.**
processingInformation.actionTokenTypes	**Set to one or more of these values:** • `customer` • `instrumentIdentifier` • `paymentInstrument`
processingInformation.commerceIndicator	**Set the value to `internet`, `MOTO`, or a payer authentication value.**
processingInformation.recurringOptions.firstRecurringPayment	**Set the value to `true`.**

REST Example: Authorizing a Customer-Initiated Recurring Payment with TMS

Request

```
{
  "processingInformation": {
    "actionList": [
      "TOKEN_CREATE"
    ],
    "actionTokenTypes": [
      "customer"
    ],
    "commerceIndicator": "internet",
    "recurringOptions": {
      "firstRecurringPayment": true
    }
  },
  "paymentInformation": {
    "card": {
      "number": "4111111111111111",
      "expirationMonth": "12",
      "expirationYear": "2031"
    }
  },
  "orderInformation": {
    "amountDetails": {
      "totalAmount": "102.21",
      "currency": "USD"
    },
    "billTo": {
      "firstName": "John",
      "lastName": "Doe",
      "address1": "1 Market St",
      "locality": "san francisco",
      "administrativeArea": "CA",
      "postalCode": "94105",
      "country": "US",
      "email": "test@cybs.com",
      "phoneNumber": ""
    }
  }
}
```

Response to a Successful Request

```
{
  "_links": {
    "authReversal": {
      "method": "POST",
      "href": "/pts/v2/payments/6976858134106105703954/reversals"
    },
    "self": {
      "method": "GET",
      "href": "/pts/v2/payments/6976858134106105703954"
    },
    "capture": {
      "method": "POST",
      "href": "/pts/v2/payments/6976858134106105703954/captures"
    }
  },
  "clientReferenceInformation": {
    "code": "1697685813462"
  },
  "id": "6976858134106105703954",
  "orderInformation": {
    "amountDetails": {
      "authorizedAmount": "102.21",
      "currency": "USD"
    }
  },
  "paymentAccountInformation": {
    "card": {
      "type": "001"
    }
  },
  "paymentInformation": {
    "tokenizedCard": {
      "type": "001"
    },
    "card": {
      "type": "001"
    }
  },
  "pointOfSaleInformation": {
    "terminalId": "111111"
  },
  "processorInformation": {
    "approvalCode": "888888",
    "networkTransactionId": "123456789619999",
    "transactionId": "123456789619999",
    "responseCode": "100",
    "avs": {
      "code": "X",
      "codeRaw": "I1"
    }
  },
  "reconciliationId": "62698397FNN143CC",
  "status": "AUTHORIZED",
  "submitTimeUtc": "2023-10-19T03:23:33Z",
  "tokenInformation": {
    "customer": {
      "id": "080A3A742BF87171E063A2598D0AEABE"
    }
  }
}
```

Customer-Initiated Recurring Payment with Enrollable Network Tokens

A recurring payment is a credentials-on-file (COF) transaction in a series of payments that you bill to a customer at a fixed amount, at regular intervals that do not exceed one

year between transactions. The series of recurring payments is the result of an agreement between you and the customer for the purchase of goods or services that are provided at regular intervals.
Mastercard uses standing order and subscription payments instead of recurring payments. See *Mastercard Standing Order Payments* on page 167 and *Mastercard Subscription Payments* on page 173.

Recurring Billing Service for Recurring Payments

Important

Do not use this document for the Recurring Billing service.
Use the *Recurring Billing Developer Guide*. When you use the Recurring Billing service, Cybersource saves and stores payment credentials for recurring transactions, ensuring compliance with COF best practices.

Using Enrollable Network Tokens

The Token Management Service can enroll certain network tokens, known as device tokens, into an instrument identifier token for future payments. Device tokens store and encrypt card-on-file information which enables customers to make quick and easy purchases using their mobile device. When authorizing a credentialed payment with a device token, you must create and store the device token in a TMS instrument identifier token. To do this, include the device token information in the **paymentInformation.tokenizedCard** fields and set the token creation fields to create an instrument identifier token.
Follow-on merchant-initiated transactions are performed using the created instrument identifier as the payment information. For more information about how to request a merchant-initiated transaction, see *Merchant-Initiated Recurring Payments with TMS* on page 159.
Device tokens are also known as digital payments, digital wallets, and tokenized cards.

Network Token Types

In your request, include the **processingInformation.paymentSolution** field to identify the device token type you are using, and set it to one of these possible values:

- `001`: Apple Pay
- `004`: Cybersource In-App Solution
- `005`: Masterpass
- `006`: Android Pay
- `007`: Chase Pay
- `008`: Samsung Pay
- `012`: Google Pay
- `014`: Mastercard credential-on-file (COF) payment network token
- `015`: Visa credential-on-file (COF) payment network token
- `027`: Click to Pay

- `visacheckout`: Visa Click to Pay.

Endpoint

Production: `POST https://api.cybersource.com/pts/v2/payments`
Test: `POST https://apitest.cybersource.com/pts/v2/payments`

Required Fields for Authorizing a Customer-Initiated Recurring Payments with Enrollable Network Tokens

orderInformation.amountDetails.currency	
orderInformation.amountDetails.totalAmount	
orderInformation.billTo.address1	
orderInformation.billTo.administrativeArea	
orderInformation.billTo.country	
orderInformation.billTo.email	
orderInformation.billTo.firstName	
orderInformation.billTo.lastName	
orderInformation.billTo.locality	
orderInformation.billTo.phoneNumber	
orderInformation.billTo.postalCode	
paymentInformation.tokenizedCard.expirationMonth	
paymentInformation.tokenizedCard.expirationYear	
paymentInformation.tokenizedCard.number	
paymentInformation.tokenizedCard.transactionType	Set the value to `1`.
processingInformation.actionList	Set the value to `TOKEN_CREATE`.
processingInformation.actionTokenTypes	Set the value to `instrumentIdentifier`.
processingInformation.commerceIndicator	Set the value to `internet`, `MOTO`, or a payer authentication value.
processingInformation.paymentSolution	Set to one of these possible values: • `001`: Apple Pay • `004`: Cybersource In-App Solution • `005`: Masterpass • `006`: Android Pay • `007`: Chase Pay • `008`: Samsung Pay • `012`: Google Pay

- `014`: Mastercard credential-on-file (COF) payment network token
- `015`: Visa credential-on-file (COF) payment network token
- `027`: Click to Pay
- `visacheckout`: Visa Click to Pay.

REST Example: Authorizing a Customer-Initiated Recurring Payment with Enrollable Network Tokens

Request

```
{
  "processingInformation": {
    "actionList": [
      "TOKEN_CREATE"
    ],
    "actionTokenTypes": [
      "instrumentIdentifier"
    ],
    "commerceIndicator": "internet",
    "paymentSolution": "001"
  },
  "paymentInformation": {
    "tokenizedCard": {
      "number": "4111111111111111",
      "expirationMonth": "02",
      "expirationYear": "2025",
      "transactionType": "1"
    }
  },
  "orderInformation": {
    "amountDetails": {
      "totalAmount": "102.21",
      "currency": "USD"
    },
    "billTo": {
      "firstName": "John",
      "lastName": "Smith",
      "address1": "123 Happy St",
      "locality": "Austin",
      "administrativeArea": "TX",
      "postalCode": "78757",
      "country": "US",
      "email": "test@cybs.com",
      "phoneNumber": "444-4444-4444"
    }
  }
}
```

Response to a Successful Request

```
{
  "_links": {
   "authReversal": {
     "method": "POST",
     "href": "/pts/v2/payments/7094060020036241803954/reversals"
   },
   "self": {
     "method": "GET",
     "href": "/pts/v2/payments/7094060020036241803954"
   },
   "capture": {
     "method": "POST",
     "href": "/pts/v2/payments/7094060020036241803954/captures"
   }
  },
  "clientReferenceInformation": {
   "code": "1709406002076"
  },
  "id": "7094060020036241803954",
  "orderInformation": {
   "amountDetails": {
     "authorizedAmount": "102.21",
     "currency": "USD"
   }
  },
  "paymentAccountInformation": {
   "card": {
     "type": "001"
   }
  },
  "paymentInformation": {
   "tokenizedCard": {
     "type": "001"
   },
   "card": {
     "type": "001"
   }
  },
  "pointOfSaleInformation": {
   "terminalId": "111111"
  },
  "processorInformation": {
   "approvalCode": "888888",
   "networkTransactionId": "123456789619999",
   "transactionId": "123456789619999",
   "responseCode": "100",
   "avs": {
     "code": "X",
     "codeRaw": "I1"
   }
  },
  "reconciliationId": "60616704ST7Q27K2",
  "status": "AUTHORIZED",
  "submitTimeUtc": "2024-03-02T19:00:02Z",
  "tokenInformation": {
   "instrumentidentifierNew": false,
   "instrumentIdentifier": {
     "state": "ACTIVE",
     "id": "7010000000016241111"
   }
  }
 }
```

Merchant-Initiated Recurring Payments with PAN

After the initial recurring payment (CIT), subsequent recurring payments are merchant-initiated transactions (MITs).

Prerequisites

The first transaction in a recurring payment is a customer-initiated transaction (CIT). Before you can perform a subsequent merchant-initiated transaction (MIT), you must store the customer's credentials for later use. Before you can store the customer's credentials, you must get their consent to store their private information. This is also known as establishing a relationship with the customer.

Supported Card Types

These are the supported card types for processing credentialed transactions:

- Visa

Mastercard uses standing order and subscription payments instead of recurring payments. See *Mastercard Standing Order Payments* on page 167 and *Mastercard Subscription Payments* on page 173.

Address Verification Service for Recurring Payments

If your processor supports the Address Verification Service (AVS), then the AVS should verify every authorization request. Cybersource recommends checking the AVS's results for the first recurring payment to ensure that the payment information is accurate and to reduce the risk of fraud.
You must determine how to handle the AVS results for any subsequent recurring payments that are not the same as the already-verified billing address information from the first recurring payment.

Replacing Expiration Dates

If the customer's card-on-file is going to expire before a scheduled subsequent recurring payment, your processor may allow you to replace the expiration date with the date 12/2099.

Important

Do not replace a card's expiration date if the card is not expired.

Using this replacement expiration date does not guarantee a successful authorization request. It is your responsibility to know if your processor supports this feature. Not all

issuing banks support the 12/2099 expiration date and may decline the authorization request.
To include this date in the authorization request, use these fields and values.

paymentInformation.card.expirationMonth	**Set to** `12`**.**
paymentInformation.card.expirationYear	**Set to** `99`**.**

Endpoint

Production: `POST https://api.cybersource.com/pts/v2/payments`
Test: `POST https://apitest.cybersource.com/pts/v2/payments`

Required Fields for Authorizing a Merchant-Initiated Recurring Payment

Use these required fields to authorize subsequent recurring payments.

orderInformation.amountDetails.currency	
orderInformation.amountDetails.totalAmount	
orderInformation.billTo.address1	
orderInformation.billTo.administrativeArea	
orderInformation.billTo.country	
orderInformation.billTo.email	
orderInformation.billTo.firstName	
orderInformation.billTo.lastName	
orderInformation.billTo.locality	
orderInformation.billTo.phoneNumber	
orderInformation.billTo.postalCode	
paymentInformation.card.expirationMonth	
paymentInformation.card.expirationYear	
paymentInformation. card. number	
processingInformation. authorizationOptions. initiator. merchantInitiatedTransaction. previousTransactionID	
processingInformation. authorizationOptions. initiator. storedCredentialUsed	**Set the value to** `true`**.**
processingInformation. authorizationOptions. initiator. type	**Set the value to** `merchant`**.**
processingInformation. commerceIndicator	**Set the value to** `recurring`**.**

Card-Specific Required Fields for Authorizing Subsequent Recurring Payments

Some card companies require additional information when making authorizations with stored credentials.

Discover

Include the authorization amount from the original transaction in this field:

processingInformation.authorizationOptions.initiator.merchantInitiatedTransaction.originalAuthorizedAmount

Mastercard

Mastercard supports subscription and standing order payments instead of recurring payments.
See *Mastercard Subscription Payments* on page 173 and *Mastercard Standing Order Payments* on page 167.

REST Example: Authorizing a Merchant-Initiated Recurring Payment

Request

```
{
  "processingInformation": {
    "commerceIndicator": "recurring",
    "authorizationOptions": {
      "initiator": {
        "storedCredentialUsed": "true",
        "type": "merchant",
        "merchantInitiatedTransaction": {
          "previousTransactionId": "123456789619999",
          "originalAuthorizedAmount": "100"   //Discover Only
        }
      }
    }
  },
  "orderInformation": {
    "billTo": {
      "firstName": "John",
      "lastName": "Doe",
      "address1": "201 S. Division St.",
      "postalCode": "48104-2201",
      "locality": "Ann Arbor",
      "administrativeArea": "MI",
      "country": "US",
      "phoneNumber": "5554327113",
      "email": "test@cybs.com"
    },
    "amountDetails": {
      "totalAmount": "100.00",
      "currency": "USD"
    }
  },
  "paymentInformation": {
    "card": {
      "expirationYear": "2031",
      "number": "4111xxxxxxxxxxxx",
      "expirationMonth": "12"
    }
  }
}
```

Response to a Successful Request

```
{
  "_links": {
    "authReversal": {
      "method": "POST",
      "href": "/pts/v2/payments/6530824710046809304002/reversals"
    },
    "self": {
      "method": "GET",
      "href": "/pts/v2/payments/6530824710046809304002"
    },
    "capture": {
      "method": "POST",
      "href": "/pts/v2/payments/6530824710046809304002/captures"
    }
  },
  "clientReferenceInformation": {
    "code": "1653082470983"
  },
  "id": "6530824710046809304002",
  "orderInformation": {
    "amountDetails": {
      "authorizedAmount": "100.00",
      "currency": "USD"
    }
  },
  "paymentAccountInformation": {
    "card": {
      "type": "002"
    }
  },
  "paymentInformation": {
    "tokenizedCard": {
      "type": "002"
    },
    "card": {
      "type": "002"
    }
  },
  "pointOfSaleInformation": {
    "terminalId": "111111"
  },
  "processorInformation": {
    "approvalCode": "888888",
    "authIndicator": "1",
    "networkTransactionId": "123456789619999",
    "transactionId": "123456789619999",
      "responseCode": "100",
      "avs": {
        "code": "X",
        "codeRaw": "I1"
      }
    },
    "reconciliationId": "79710341A39WTT5W",
    "status": "AUTHORIZED",
    "submitTimeUtc": "2022-05-20T21:34:31Z"
}
```

Merchant-Initiated Recurring Payments with TMS

After the customer-initiated recurring payment, you can send merchant-initiated recurring payments using one or more TMS token types:

Customer	Customer tokens store one or more customer payment instrument tokens and shipping address tokens. Including a customer token eliminates the need to include billing information, card information, and the previous transaction's ID. `"paymentInformation": { "customer": { "id": "07C9CA98022DA498E063A2598D0AA400" } }` For more information about this TMS token type, see *Customer Tokens* in the Token Management Service Developer Guide.
Payment Instrument	Payment instrument tokens store an instrument identifier token, card information, and billing information. Payment instruments are not linked to a customer token. Including a payment instrument eliminates the need to include billing information, card information, and the previous transaction's ID. `"paymentInformation": { "paymentInstrument": { "id": "07CA24EF20F9E2C9E063A2598D0A8565" } }`

	For more information about this TMS token type, see *Payment Instrument Token* in the Token Management Service Developer Guide.

Instrument Identifier	Instrument identifier tokens store only a PAN. Including an instrument identifier eliminates the need to include a PAN and the previous transaction's ID.
	`"paymentInformation": {`
	`  "instrumentIdentifier": {`
	`    "id": "7010000000016241111"`
	`  }`
	`}`
	For more information about this TMS token type, see *Instrument Identifier Token* in the Token Management Service Developer Guide.

Prerequisites

The first transaction in a recurring payment is a customer-initiated transaction (CIT). Before you can perform a subsequent merchant-initiated transaction (MIT), you must store the customer's credentials for later use. Before you can store the customer's credentials, you must get their consent to store their private information. This is also known as establishing a relationship with the customer.

Supported Card Types

These are the supported card types for processing credentialed transactions:

- Mastercard
- Visa

Mastercard uses standing order and subscription payments instead of recurring payments. See *Mastercard Standing Order Payments* on page 167 and *Mastercard Subscription Payments* on page 173.

Address Verification Service for Recurring Payments

If your processor supports the Address Verification Service (AVS), then the AVS should verify every authorization request. Cybersource recommends checking the AVS's results for the first recurring payment to ensure that the payment information is accurate and to reduce the risk of fraud.
You must determine how to handle the AVS results for any subsequent recurring payments that are not the same as the already-verified billing address information from the first recurring payment.

Replacing Expiration Dates

If the customer's card-on-file is going to expire before a scheduled subsequent recurring payment, your processor may allow you to replace the expiration date with the date 12/2099.

Important

Do not replace a card's expiration date if the card is not expired.

Using this replacement expiration date does not guarantee a successful authorization request. It is your responsibility to know if your processor supports this feature. Not all issuing banks support the 12/2099 expiration date and may decline the authorization request.
To include this date in the authorization request, use these fields and values.

paymentInformation.card.expirationMonth	**Set to** `12`**.**
paymentInformation.card.expirationYear	**Set to** `99`**.**

Endpoint

Production: `POST https://api.cybersource.com/pts/v2/payments`
Test: `POST https://apitest.cybersource.com/pts/v2/payments`

Required Fields for Authorizing a Merchant-Initiated Recurring Payments with TMS

Use these required fields to authorize subsequent recurring payments.

orderInformation.amountDetails.currency	
orderInformation.amountDetails.totalAmount	
paymentInformation.[tokentype].id	**Where [tokentype] is the TMS token type you are using:** • **customer** • **instrumentIdentifier** • **paymentInstrument**
processingInformation.commerceIndicator	**Set the value to** `recurring`**.**

Instrument Identifier Required Fields

If you are using the **paymentInformation.instrumentIdentifier.id** token, include these required fields in addition to the required fields listed above.

orderInformation.billTo.address1

orderInformation.billTo.administrativeArea

orderInformation.billTo.country

orderInformation.billTo.email

orderInformation.billTo.firstName

orderInformation.billTo.lastName

orderInformation.billTo.locality

orderInformation.billTo.phoneNumber

orderInformation.billTo.postalCode

paymentInformation.card.expirationMonth

paymentInformation.card.expirationYear

Card-Specific Field

Some card companies require additional fields when making authorizations with stored credentials. Include this field if you are using these card types:

Discover	**processingInformation.authorizationOptions.initiator.merchantInitiatedTransaction.originalAuthorizedAmou**
Mastercard	Mastercard supports subscription and standing order payments instead of recurring payments. See *Mastercard Subscription Payments* on page 173 and *Mastercard Standing Order Payments* on page 167.

REST Example: Authorizing a Merchant-Initiated Recurring Payment with a TMS Instrument Identifier

Request

```
{
  "processingInformation": {
    "commerceIndicator": "recurring"
  },
  "paymentInformation": {
    "card": {
      "expirationMonth": "12",
      "expirationYear": "2025"
    },
    "instrumentIdentifier": {
      "id": "4111xxxxxxxxxxxx"
    }
  },
  "orderInformation": {
    "amountDetails": {
      "totalAmount": "102.21",
      "currency": "USD"
    },
    "billTo": {
      "firstName": "John",
      "lastName": "Smith",
      "address1": "1 Market St",
      "locality": "san francisco",
      "administrativeArea": "CA",
      "postalCode": "94105",
      "country": "US",
      "email": "test@cybs.com",
      "phoneNumber": "4158880000"
    }
  }
}
```

Response to a Successful Request

```
{
  "_links": {
    "authReversal": {
      "method": "POST",
      "href": "/pts/v2/payments/6530824710046809304002/reversals"
    },
    "self": {
      "method": "GET",
      "href": "/pts/v2/payments/6530824710046809304002"
    },
    "capture": {
      "method": "POST",
      "href": "/pts/v2/payments/6530824710046809304002/captures"
    }
  },
  "clientReferenceInformation": {
    "code": "1653082470983"
  },
  "id": "6530824710046809304002",
  "orderInformation": {
    "amountDetails": {
      "authorizedAmount": "100.00",
      "currency": "USD"
    }
  },
  "paymentAccountInformation": {
    "card": {
      "type": "002"
    }
  },
  "paymentInformation": {
    "tokenizedCard": {
      "type": "002"
    },
    "card": {
      "type": "002"
    }
  },
  "pointOfSaleInformation": {
    "terminalId": "111111"
  },
  "processorInformation": {
    "approvalCode": "888888",
    "authIndicator": "1",
    "networkTransactionId": "123456789619999",
    "transactionId": "123456789619999",
    "responseCode": "100",
    "avs": {
      "code": "X",
      "codeRaw": "I1"
    }
  },
  "reconciliationId": "79710341A39WTT5W",
  "status": "AUTHORIZED",
  "submitTimeUtc": "2022-05-20T21:34:31Z"
}
```

REST Example: Authorizing a Merchant-Initiated Recurring Payment with TMS Payment Instrument

Request

```
{
  "clientReferenceInformation": {
    "code": "TC50171_3"
  },
  "processingInformation": {
    "commerceIndicator": "recurring"
  },
  "paymentInformation": {
    "paymentInstrument": {
      "id": "07DB0915C20F2DDBE063A2598D0A6F26"
    }
  },
  "orderInformation": {
    "amountDetails": {
      "totalAmount": "102.21",
      "currency": "USD"
    }
  }
}
```

Response to a Successful Request

```
{
  "_links": {
    "authReversal": {
      "method": "POST",
      "href": "/pts/v2/payments/6974839908106304103955/reversals"
    },
    "self": {
      "method": "GET",
      "href": "/pts/v2/payments/6974839908106304103955"
    },
    "capture": {
      "method": "POST",
      "href": "/pts/v2/payments/6974839908106304103955/captures"
    }
  },
  "clientReferenceInformation": {
    "code": "TC50171_3"
  },
  "id": "6974839908106304103955",
  "orderInformation": {
    "amountDetails": {
      "authorizedAmount": "102.21",
      "currency": "USD"
    }
  },
  "paymentAccountInformation": {
    "card": {
      "type": "001"
    }
  },
  "paymentInformation": {
    "tokenizedCard": {
      "type": "001"
    },
    "instrumentIdentifier": {
      "id": "7010000000016241111",
      "state": "ACTIVE"
    },
    "paymentInstrument": {
      "id": "07DB0915C20F2DDBE063A2598D0A6F26"
    },
    "card": {
      "type": "001"
    }
  },
  "pointOfSaleInformation": {
    "terminalId": "111111"
  },
  "processingInformation": {
    "paymentSolution": "015"
  },
  "processorInformation": {
    "paymentAccountReferenceNumber": "V0010013022298169667504231315",
    "approvalCode": "888888",
    "networkTransactionId": "123456789619999",
    "transactionId": "123456789619999",
    "responseCode": "100",
    "avs": {
      "code": "X",
      "codeRaw": "I1"
    }
  },
  "reconciliationId": "62599243NNMR6324",
  "status": "AUTHORIZED",
  "submitTimeUtc": "2023-10-16T19:19:51Z"
}
```

REST Example: Authorizing a Merchant-Initiated Recurring Payment with a TMS Customer Token

Request

```
{
 "clientReferenceInformation": {
  "code": "TC50171_3"
 },
 "processingInformation": {
  "commerceIndicator": "recurring"
 },
 "paymentInformation": {
  "customer": {
   "id": "07DB50E35AE11DA2E063A2598D0A9995"
  }
 },
 "orderInformation": {
  "amountDetails": {
   "totalAmount": "102.21",
   "currency": "USD"
  }
 }
}
```

Response to a Successful Request

```
{
 "_links": {
  "authReversal": {
   "method": "POST",
   "href": "/pts/v2/payments/6974846967476340503955/reversals"
  },
  "self": {
   "method": "GET",
   "href": "/pts/v2/payments/6974846967476340503955"
  },
  "capture": {
   "method": "POST",
   "href": "/pts/v2/payments/6974846967476340503955/captures"
  }
 },
 "clientReferenceInformation": {
  "code": "TC50171_3"
 },
 "id": "6974846967476340503955",
 "orderInformation": {
  "amountDetails": {
   "authorizedAmount": "102.21",
   "currency": "USD"
  }
 },
 "paymentAccountInformation": {
  "card": {
   "type": "001"
  }
  },
  "paymentInformation": {
    "tokenizedCard": {
      "type": "001"
    },
    "card": {
      "type": "001"
    }
  },
  "pointOfSaleInformation": {
    "terminalId": "111111"
  },
  "processingInformation": {
    "paymentSolution": "015"
  },
  "processorInformation": {
    "paymentAccountReferenceNumber": "V0010013022298169667504231315",
    "approvalCode": "888888",
    "networkTransactionId": "123456789619999",
    "transactionId": "123456789619999",
    "responseCode": "100",
    "avs": {
      "code": "X",
      "codeRaw": "I1"
    }
  },
  "reconciliationId": "62599950BNN133LK",
  "status": "AUTHORIZED",
  "submitTimeUtc": "2023-10-16T19:31:36Z"
}
```

Mastercard Standing Order Payments

A standing order payment is a recurring COF transaction that is a variable amount at a regular interval, such as a utility bill, not to exceed one year between transactions. The series of recurring payments is the result of an agreement between you and the customer for the purchase of goods or services that are provided at regular intervals.

Mastercard Initial CIT Standing Order Payment

The first transaction in a standing order payment is a customer-initiated transaction (CIT). Before you can perform a subsequent merchant-initiated transaction (MIT), you must store the customer's credentials for later use. Before you can store the user's credentials, you must get the customer's consent to store their private information. This process is also known as establishing a relationship with the customer.

Endpoint

Production: `POST https://api.cybersource.com/pts/v2/payments`
Test: `POST https://apitest.cybersource.com/pts/v2/payments`

Successful Response

You must store the network transaction ID from the successful response message to include in subsequent MIT authorization requests in order to associate the CIT to the MIT. The network transaction ID is the **processorInformation.networkTransactionId** field value. Store the network transaction ID, which is the **processorInformation.networkTransactionId** field value, from the successful response message. You must include the network transaction ID in subsequent MIT authorization requests in order to associate the CIT to the MIT.

Required Fields for Authorizing Initial CIT Standing Order Payments

Use these required fields to authorize initial customer-initated standing order payments.

orderInformation.amountDetails.currency	
orderInformation.amountDetails.totalAmount	
orderInformation.billTo.address1	
orderInformation.billTo.administrativeArea	
orderInformation.billTo.country	
orderInformation.billTo.email	
orderInformation.billTo.firstName	
orderInformation.billTo.lastName	
orderInformation.billTo.locality	
orderInformation.billTo.phoneNumber	
orderInformation.billTo.postalCode	
paymentInformation.card.expirationMonth	
paymentInformation.card.expirationYear	
paymentInformation.card.number	
processingInformation. authorizationOptions. initiator. credentialStoredOnFile	**Set the value to `true`.**
processingInformation. authorizationOptions. initiator.type	**Set the value to `customer`.**
processingInformation. commerceIndicator	**Set the value to `internet`, `MOTO`, or a payer authentication value.**
processingInformation. authorizationOptions. initiator. merchantInitiatedTransaction. reason	**Set the value to `8`.**

Mastercard Initial CIT Standing Order Payment with TMS

The first transaction in a standing order payment is a customer-initiated transaction (CIT). Before you can perform a subsequent merchant-initiated transaction (MIT), you must store the customer's credentials for later use. Before you can store the user's credentials, you must get the customer's consent to store their private information. This process is also known as establishing a relationship with the customer.

Creating a TMS Token

When sending the initial CIT, you can create a TMS token to store the customer's credentials for the subsequent MITs. To create a TMS token, include the **processingInformation.actionTokenTypes** field in the authorization request. Set the field to one of these values based on the TMS token type you want to create:

Customer

Customer tokens store one or more customer payment instrument tokens and shipping address tokens.

Including a customer token in subsequent MITs eliminates the need to include billing information, card information, and the previous transaction's ID.

```
"processingInformation": {
   "actionTokenTypes": [
     "customer"
]
```

For more information about this TMS token type, see *Customer Tokens* in the Token Management Service Developer Guide.

Payment Instrument

Payment instrument tokens store an instrument identifier token, card information, and billing information. Payment instruments are not linked to a customer token. Including a payment instrument in subsequent MITs eliminates the need to include billing information, card information, and the previous transaction's ID.

```
"processingInformation": {
   "actionTokenTypes": [
     "paymentInstrument"
]
```

For more information about this TMS token type, see *Payment Instrument Token* in

the Token Management Service Developer Guide.

Instrument Identifier

Instrument identifier tokens store a PAN. Including an instrument identifier in subsequent MITs eliminates the need to include a PAN and the previous transaction's ID.

```
"processingInformation": {
  "actionTokenTypes": [
   "instrumentIdentifier"
]
```

For more information about this TMS token type, see *Instrument Identifier Token* **in the Token Management Service Developer Guide.**

Instrument Identifier, Payment Instrument, and Customer Identifier

You can also create multiple TMS token types in the same authorization. This example includes an instrument identifier, a payment instrument, and a customer token in the same authorization:

```
 "processingInformation": {
  "actionTokenTypes": [
     "instrumentIdentifier",
     "paymentInstrument",
     "customer"
]
```

Endpoint

Production: `POST https://api.cybersource.com/pts/v2/payments`
Test: `POST https://apitest.cybersource.com/pts/v2/payments`

Required Fields for Authorizing Initial CIT Standing Order Payments with TMS

orderInformation.amountDetails.currency

orderInformation.amountDetails.totalAmount

orderInformation.billTo.address1

orderInformation.billTo.administrativeArea

orderInformation.billTo.country

orderInformation.billTo.email

orderInformation.billTo.firstName

orderInformation.billTo.lastName

orderInformation.billTo.locality	
orderInformation.billTo.phoneNumber	
orderInformation.billTo.postalCode	
paymentInformation.card.expirationMonth	
paymentInformation.card.expirationYear	
paymentInformation.card.number	
processingInformation.actionList	Set the value to `TOKEN_CREATE`
processingInformation.actionTokenTypes	Set to one or more of these values: • `customer` • `instrumentIdentifier` • `paymentInstrument`
processingInformation.authorizationOptions.initiator.merchantInitiatedTransaction.reason	Set the value to `8`.
processingInformation.commerceIndicator	Set the value to `internet`, `MOTO`, or a payer authentication value.

REST Example: Authorizing Initial CIT Standing Order Payments with TMS

Request

```
{
  "processingInformation": {
    "actionList": ["TOKEN_CREATE"],
    "actionTokenTypes": ["customer"],
    "commerceIndicator": "internet",
    "authorizationOptions": {
      "initiator": {
        "merchantInitiatedTransaction": {
          "reason": "8"
        }
      }
    }
  },
  "paymentInformation": {
    "card": {
      "number": "555555555555xxxx",
      "expirationMonth": "12",
      "expirationYear": "2031"
    }
  },
  "orderInformation": {
    "amountDetails": {
      "totalAmount": "100.00",
      "currency": "USD"
    },
    "billTo": {
      "firstName": "John",
      "lastName": "Smith",
      "address1": "123 Happy St",
      "locality": "Sunnyville",
      "administrativeArea": "CA",
      "postalCode": "55555",
      "country": "US",
      "email": "test@cybs.com",
      "phoneNumber": "444-4444-4444"
    }
  }
}
```

Response to a Successful Request

```
{
  "_links": {
    "authReversal": {
      "method": "POST",
      "href": "/pts/v2/payments/7064959411486706503954/reversals"
    },
    "self": {
      "method": "GET",
      "href": "/pts/v2/payments/7064959411486706503954"
    },
    "capture": {
      "method": "POST",
      "href": "/pts/v2/payments/7064959411486706503954/captures"
    }
  },
  "clientReferenceInformation": {
    "code": "1706495941197"
  },
  "id": "7064959411486706503954",
  "orderInformation": {
    "amountDetails": {
      "authorizedAmount": "100.00",
      "currency": "USD"
    }
  },
  "paymentAccountInformation": {
    "card": {
      "type": "002"
    }
  },
  "paymentInformation": {
    "tokenizedCard": {
      "type": "002"
    },
    "card": {
      "type": "002"
    }
  },
  "pointOfSaleInformation": {
    "terminalId": "111111"
  },
  "processorInformation": {
    "approvalCode": "888888",
    "authIndicator": "1",
    "networkTransactionId": "123456789619999",
    "transactionId": "123456789619999",
    "responseCode": "100",
    "avs": {
      "code": "X",
      "codeRaw": "I1"
    }
  },
  "reconciliationId": "680915409RRMGL34",
  "status": "AUTHORIZED",
  "submitTimeUtc": "2024-01-29T02:39:01Z",
  "tokenInformation": {
    "customer": {
      "id": "100D6CDA178DD64DE063A2598D0AD3D5"
    }
  }
}
```

Mastercard Subscription Payments

A subscription payment is a recurring COF transaction that is processed at a fixed amount at regular intervals not to exceed one year between transactions. The series of recurring payments is the result of an agreement between you and the customer for the purchase of goods or services that are provided at regular intervals.

Mastercard CIT Initial Subscription Payment

The first transaction in a subscription payment is a customer-initiated transaction (CIT). Before you can perform a subsequent merchant-initiated transaction (MIT), you must store the customer's credentials for later use. Before you can store the user's credentials, you must get the customer's consent to store their private information. This process is also known as establishing a relationship with the customer.

Endpoint

Production: `POST https://api.cybersource.com/pts/v2/payments`
Test: `POST https://apitest.cybersource.com/pts/v2/payments`

Successful Response

You must store the network transaction ID from the successful response message to include in subsequent MIT authorization requests in order to associate the CIT to the MIT. The network transaction ID is the **processorInformation.networkTransactionId** field value. Store the network transaction ID, which is the **processorInformation.networkTransactionId** field value, from the successful response message. You must include the network transaction ID in subsequent MIT authorization requests in order to associate the CIT to the MIT.

Required Fields for Authorizing CIT Initial Subscription Payments

orderInformation.amountDetails.currency	
orderInformation.amountDetails.totalAmount	
orderInformation.billTo.address1	
orderInformation.billTo.administrativeArea	
orderInformation.billTo.country	
orderInformation.billTo.email	
orderInformation.billTo.firstName	
orderInformation.billTo.lastName	
orderInformation.billTo.locality	
orderInformation.billTo.phoneNumber	
orderInformation.billTo.postalCode	
paymentInformation.card.expirationMonth	
paymentInformation.card.expirationYear	
paymentInformation.card.number	
processingInformation.authorizationOptions.initiator.credentialStoredOnFile	Set the value to `true`.
processingInformation.authorizationOptions.initiator.type	Set the value to `customer`.
processingInformation.commerceIndicator	Set the value to `recurring`.
processingInformation.authorizationOptions.initiator.merchantInitiatedTransaction.reason	Set the value to `7`.

Mastercard CIT Initial Subscription Payment with TMS

The first transaction in a subscription payment is a customer-initiated transaction (CIT). Before you can perform a subsequent merchant-initiated transaction (MIT), you must store the customer's credentials for later use. Before you can store the user's credentials, you must get the customer's consent to store their private information. This process is also known as establishing a relationship with the customer.

Creating a TMS Token

When sending the initial CIT, you can create a TMS token to store the customer's credentials for the subsequent MITs. To create a TMS token, include the **processingInformation.actionTokenTypes** field in the authorization request. Set the field to one of these values based on the TMS token type you want to create:

Customer	**Customer tokens store one or more customer payment instrument tokens and shipping address tokens.** **Including a customer token in subsequent MITs eliminates the need to include billing**

information, card information, and the previous transaction's ID.

```
"processingInformation": {
   "actionTokenTypes": [
     "customer"
 ]
```

For more information about this TMS token type, see *Customer Tokens* in the Token Management Service Developer Guide.

Payment Instrument

Payment instrument tokens store an instrument identifier token, card information, and billing information. Payment instruments are not linked to a customer token. Including a payment instrument in subsequent MITs eliminates the need to include billing information, card information, and the previous transaction's ID.

```
"processingInformation": {
   "actionTokenTypes": [
     "paymentInstrument"
 ]
```

For more information about this TMS token type, see *Payment Instrument Token* in the Token Management Service Developer Guide.

Instrument Identifier

Instrument identifier tokens store a PAN. Including an instrument identifier in subsequent MITs eliminates the need to include a PAN and the previous transaction's ID.

```
"processingInformation": {
   "actionTokenTypes": [
     "instrumentIdentifier"
 ]
```

For more information about this TMS token type, see *Instrument Identifier Token* in the Token Management Service Developer Guide.

Instrument Identifier, Payment Instrument, and Customer Identifier

You can also create multiple TMS token types in the same authorization. This example includes an instrument identifier, a

payment instrument, and a customer token in the same authorization:

```
"processingInformation": {
  "actionTokenTypes": [
     "instrumentIdentifier",
     "paymentInstrument",
     "customer"
]
```

Endpoint

Production: `POST https://api.cybersource.com/pts/v2/payments`
Test: `POST https://apitest.cybersource.com/pts/v2/payments`

Required Fields for Authorizing CIT Initial Subscription Payments with TMS

orderInformation.amountDetails.currency	
orderInformation.amountDetails.totalAmount	
orderInformation.billTo.address1	
orderInformation.billTo.administrativeArea	
orderInformation.billTo.country	
orderInformation.billTo.email	
orderInformation.billTo.firstName	
orderInformation.billTo.lastName	
orderInformation.billTo.locality	
orderInformation.billTo.phoneNumber	
orderInformation.billTo.postalCode	
paymentInformation.card.expirationMonth	
paymentInformation.card.expirationYear	
paymentInformation.card.number	
processingInformation.actionList	Set the value to `TOKEN_CREATE`
processingInformation.actionTokenTypes	Set to one or more of these values: • `customer` • `instrumentIdentifier` • `paymentInstrument`
processingInformation.commerceIndicator	Set the value to `recurring`.
processingInformation.authorizationOptions.initiator.merchantInitiatedTransaction.reason	Set the value to `7`.

REST Example: Authorizing Initial CIT Subscription Payments with TMS

Request

```
{
  "processingInformation": {
    "actionList": ["TOKEN_CREATE"],
    "actionTokenTypes": ["customer"],
    "commerceIndicator": "recurring",
    "authorizationOptions": {
      "initiator": {
        "merchantInitiatedTransaction": {
          "reason": "7"
        }
      }
    }
  },
  "paymentInformation": {
    "card": {
      "number": "555555555555xxxx",
      "expirationMonth": "12",
      "expirationYear": "2031"
    }
  },
  "orderInformation": {
    "amountDetails": {
      "totalAmount": "100.00",
      "currency": "USD"
    },
    "billTo": {
      "firstName": "John",
      "lastName": "Smith",
      "address1": "123 Happy St",
      "locality": "Sunnyville",
      "administrativeArea": "CA",
      "postalCode": "55555",
      "country": "US",
      "email": "test@cybs.com",
      "phoneNumber": "444-4444-4444"
    }
  }
}
```

Response to a Successful Request

```
{
  "_links": {
    "authReversal": {
      "method": "POST",
      "href": "/pts/v2/payments/7064946846256410103954/reversals"
    },
    "self": {
      "method": "GET",
      "href": "/pts/v2/payments/7064946846256410103954"
    },
    "capture": {
      "method": "POST",
      "href": "/pts/v2/payments/7064946846256410103954/captures"
    }
  },
  "clientReferenceInformation": {
    "code": "1706494684667"
  },
  "id": "7064946846256410103954",
  "orderInformation": {
    "amountDetails": {
      "authorizedAmount": "100.00",
      "currency": "USD"
    }
  },
  "paymentAccountInformation": {
    "card": {
      "type": "002"
    }
  },
  "paymentInformation": {
    "tokenizedCard": {
      "type": "002"
    },
    "card": {
      "type": "002"
    }
  },
  "pointOfSaleInformation": {
    "terminalId": "111111"
  },
  "processorInformation": {
    "approvalCode": "888888",
    "authIndicator": "1",
    "networkTransactionId": "123456789619999",
    "transactionId": "123456789619999",
    "responseCode": "100",
    "avs": {
      "code": "X",
      "codeRaw": "I1"
    }
  },
  "reconciliationId": "68091233JRRDUQ34",
  "status": "AUTHORIZED",
  "submitTimeUtc": "2024-01-29T02:18:04Z",
  "tokenInformation": {
    "customer": {
      "id": "100D1DC40CC7C803E063A2598D0A29BD"
    }
  }
}
```

Unscheduled COF Payments

An unscheduled credentials-on-file (COF) transaction uses stored payment information for a fixed or variable amount that does not occur regularly. An account top-up is one kind of unscheduled COF.

Customer-Initiated Unscheduled COF Payment with PAN

An unscheduled credentials-on-file (COF) transaction uses stored payment information for a fixed or variable amount that does not occur regularly. An account top-up is one kind of unscheduled COF.

Supported Card Types

These are the supported card types for processing credentialed transactions:

- Mastercard
- Visa

Endpoint

Production: `POST https://api.cybersource.com/pts/v2/payments`
Test: `POST https://apitest.cybersource.com/pts/v2/payments`

Successful Response

You must store the network transaction ID from the successful response message to include in subsequent MIT authorization requests in order to associate the CIT to the MIT. The network transaction ID is the **processorInformation.networkTransactionId** field value. Store the network transaction ID, which is the **processorInformation.networkTransactionId** field value, from the successful response message. You must include the network transaction ID in subsequent MIT authorization requests in order to associate the CIT to the MIT.

Required Fields for a Customer-Initiated Unscheduled COF Payment with PAN

These fields are required in a subsequent authorization request for an initial unscheduled COF payment:

orderInformation.amountDetails.currency

orderInformation.amountDetails.totalAmount

orderInformation.billTo.address1

orderInformation.billTo.administrativeArea

orderInformation.billTo.country

orderInformation.billTo.email

orderInformation.billTo.firstName

orderInformation.billTo.lastName	
orderInformation.billTo.locality	
orderInformation.billTo.phoneNumber	
orderInformation.billTo.postalCode	
paymentInformation.card.expirationMonth	
paymentInformation.card.expirationYear	
paymentInformation.card.number	
processingInformation. authorizationOptions. initiator. credentialStoredOnFile	Set the value to `true`.
processingInformation. authorizationOptions. initiator. type	Set the value to `customer`.
processingInformation. commerceIndicator	Set the value to `internet`, `MOTO`, or a payer authentication value.

REST Example: Customer-Initiated Unscheduled COF Payment with PAN

Request

```
{
   "processingInformation": {
      "commerceIndicator": "internet",
      "authorizationOptions": {
         "initiator": {
            "credentialStoredOnFile": "true",
            "type": "customer"
         }
      }
   },
   "orderInformation": {
      "billTo": {
         "firstName": "John",
         "lastName": "Doe",
         "address1": "201 S. Division St.",
         "postalCode": "48104-2201",
         "locality": "Ann Arbor",
         "administrativeArea": "MI",
         "country": "US",
         "phoneNumber": "5554327113",
         "email": "test@cybs.com"
      },
      "amountDetails": {
         "totalAmount": "100.00",
         "currency": "USD"
      }
   },
   "paymentInformation": {
      "card": {
         "expirationYear": "2031",
      "number": "4111xxxxxxxxxxxx",
      "expirationMonth": "12"
    }
  }
}
```

Response to a Successful Request

```
{
  "_links": {
    "authReversal": {
      "method": "POST",
      "href": "/pts/v2/payments/6528187198946076303004/reversals"
    },
    "self": {
      "method": "GET",
      "href": "/pts/v2/payments/6528187198946076303004"
    },
    "capture": {
      "method": "POST",
      "href": "/pts/v2/payments/6528187198946076303004/captures"
    }
  },
  "clientReferenceInformation": {
    "code": "1652818719876"
  },
  "id": "6528187198946076303004",
  "orderInformation": {
    "amountDetails": {
      "authorizedAmount": "100.00",
      "currency": "USD"
    }
  },
  "paymentAccountInformation": {
    "card": {
      "type": "001"
    }
  },
  "paymentInformation": {
    "tokenizedCard": {
      "type": "001"
    },
    "card": {
      "type": "001"
    }
  },
  "pointOfSaleInformation": {
    "terminalId": "111111"
  },
  "processorInformation": {
    "approvalCode": "888888",
    "networkTransactionId": "123456789619999",
    "transactionId": "123456789619999",
    "responseCode": "100",
    "avs": {
      "code": "X",
        "codeRaw": "I1"
      }
   },
   "reconciliationId": "63165088Z3AHV91G",
   "status": "AUTHORIZED",
   "submitTimeUtc": "2022-05-17T20:18:40Z"
}
```

Customer-Initiated Unscheduled COF Payments with TMS

An unscheduled credentials-on-file (COF) transaction uses stored payment information for a fixed or variable amount that does not occur regularly. An account top-up is one kind of unscheduled COF.

Supported Card Types

These are the supported card types for processing credentialed transactions:

- Mastercard
- Visa

Creating a TMS Token

When sending the initial CIT, you can create a TMS token to store the customer's credentials for the subsequent MITs. To create a TMS token, include the **processingInformation.actionTokenTypes** field in the authorization request. Set the field to one of these values based on the TMS token type you want to create:

Customer	**Customer tokens store one or more customer payment instrument tokens and shipping address tokens.** **Including a customer token in subsequent MITs eliminates the need to include billing information, card information, and the previous transaction's ID.** `"processingInformation": { "actionTokenTypes": [ "customer" ]` **For more information about this TMS token type, see *Customer Tokens* in the Token Management Service Developer Guide.**
Payment Instrument	**Payment instrument tokens store an instrument identifier token, card information, and billing information. Payment instruments are not linked to a customer token. Including a payment instrument in subsequent MITs eliminates**

	the need to include billing information, card information, and the previous transaction's ID.
	`"processingInformation": {`
	`  "actionTokenTypes": [`
	`    "paymentInstrument"`
	`]`
	For more information about this TMS token type, see *Payment Instrument Token* **in the Token Management Service Developer Guide.**

Instrument Identifier	**Instrument identifier tokens store a PAN. Including an instrument identifier in subsequent MITs eliminates the need to include a PAN and the previous transaction's ID.**
	`"processingInformation": {`
	`  "actionTokenTypes": [`
	`    "instrumentIdentifier"`
	`]`
	For more information about this TMS token type, see *Instrument Identifier Token* **in the Token Management Service Developer Guide.**

Instrument Identifier, Payment Instrument, and Customer Identifier	**You can also create multiple TMS token types in the same authorization. This example includes an instrument identifier, a payment instrument, and a customer token in the same authorization:**
	`"processingInformation": {`
	`  "actionTokenTypes": [`
	`    "instrumentIdentifier",`
	`    "paymentInstrument",`
	`    "customer"`
	`]`

Endpoint

Production: `POST https://api.cybersource.com/pts/v2/payments`
Test: `POST https://apitest.cybersource.com/pts/v2/payments`

Required Fields for CIT Unscheduled COF Payments with TMS

orderInformation.amountDetails.currency

orderInformation.amountDetails.totalAmount	
orderInformation.billTo.address1	
orderInformation.billTo.administrativeArea	
orderInformation.billTo.country	
orderInformation.billTo.email	
orderInformation.billTo.firstName	
orderInformation.billTo.lastName	
orderInformation.billTo.locality	
orderInformation.billTo.phoneNumber	
orderInformation.billTo.postalCode	
paymentInformation.card.expirationMonth	
paymentInformation.card.expirationYear	
paymentInformation.card.number	
processingInformation.actionList	Set the value to `TOKEN_CREATE`
processingInformation.actionTokenTypes	Set to one or more of these values: • `customer` • `instrumentIdentifier` • `paymnentInstrument`
processingInformation.commerceIndicator	Set the value to `internet`, `MOTO`, or a payer authentication value.

REST Example: Initial CIT Unscheduled COF Payment in TMS

Request

```
{
  "processingInformation": {
    "actionList": [
      "TOKEN_CREATE"
    ],
    "actionTokenTypes": [
      "customer"
    ],
    "commerceIndicator": "internet"
  },
  "paymentInformation": {
    "card": {
      "number": "4111111111111111",
      "expirationMonth": "12",
      "expirationYear": "2031"
    }
  },
  "orderInformation": {
    "amountDetails": {
      "totalAmount": "102.21",
      "currency": "USD"
    },
    "billTo": {
      "firstName": "John",
      "lastName": "Doe",
      "address1": "1 Market St",
      "locality": "san francisco",
      "administrativeArea": "CA",
      "postalCode": "94105",
      "country": "US",
      "email": "test@cybs.com",
      "phoneNumber": "444-4444-4444"
    }
  }
}
```

Response to a Successful Request

```
{
  "_links": {
    "authReversal": {
      "method": "POST",
      "href": "/pts/v2/payments/6976866073586557303955/reversals"
    },
    "self": {
      "method": "GET",
      "href": "/pts/v2/payments/6976866073586557303955"
    },
    "capture": {
      "method": "POST",
      "href": "/pts/v2/payments/6976866073586557303955/captures"
    }
  },
  "clientReferenceInformation": {
    "code": "1697686607441"
  },
  "id": "6976866073586557303955",
  "orderInformation": {
    "amountDetails": {
      "authorizedAmount": "102.21",
      "currency": "USD"
    }
  },
  "paymentAccountInformation": {
    "card": {
      "type": "001"
    }
  },
  "paymentInformation": {
    "tokenizedCard": {
      "type": "001"
    },
    "card": {
      "type": "001"
    }
  },
  "pointOfSaleInformation": {
    "terminalId": "111111"
  },
  "processorInformation": {
    "approvalCode": "888888",
    "networkTransactionId": "123456789619999",
    "transactionId": "123456789619999",
    "responseCode": "100",
    "avs": {
      "code": "X",
      "codeRaw": "I1"
    }
  },
  "reconciliationId": "62699023FNN143DG",
  "status": "AUTHORIZED",
  "submitTimeUtc": "2023-10-19T03:36:47Z",
  "tokenInformation": {
    "customer": {
      "id": "080A6C3842C72DCBE063A2598D0AA98B"
    }
  }
}
```

Customer-Initiated Unscheduled COF Payment with Enrollable Network Tokens

An unscheduled credentials-on-file (COF) transaction uses stored payment information for a fixed or variable amount that does not occur regularly. An account top-up is one kind of unscheduled COF.

Using Enrollable Network Tokens

The Token Management Service can enroll certain network tokens, known as device tokens, into an instrument identifier token for future payments. Device tokens store and encrypt card-on-file information which enables customers to make quick and easy purchases using their mobile device. When authorizing a credentialed payment with a device token, you must create and store the device token in a TMS instrument identifier token. To do this, include the device token information in the **paymentInformation.tokenizedCard** fields and set the token creation fields to create an instrument identifier token.

Follow-on merchant-initiated transactions are performed using the created instrument identifier as the payment information. For more information about how to request a merchant-initiated transaction, see *Merchant-Initiated Unscheduled COF Payments with TMS* on page 193.

Device tokens are also known as digital payments, digital wallets, and tokenized cards.

Network Token Types

In your request, include the **processingInformation.paymentSolution** field to identify the device token type you are using, and set it to one of these possible values:

- `001`: Apple Pay
- `004`: Cybersource In-App Solution
- `005`: Masterpass
- `006`: Android Pay
- `007`: Chase Pay
- `008`: Samsung Pay
- `012`: Google Pay
- `014`: Mastercard credential-on-file (COF) payment network token
- `015`: Visa credential-on-file (COF) payment network token
- `027`: Click to Pay
- `visacheckout`: Visa Click to Pay.

Endpoint

Production: `POST https://api.cybersource.com/pts/v2/payments`
Test: `POST https://apitest.cybersource.com/pts/v2/payments`

Required Fields for CIT Unscheduled COF Payment with Enrollable Network Tokens

orderInformation.amountDetails.currency	
orderInformation.amountDetails.totalAmount	
orderInformation.billTo.address1	
orderInformation.billTo.administrativeArea	
orderInformation.billTo.country	
orderInformation.billTo.email	
orderInformation.billTo.firstName	
orderInformation.billTo.lastName	
orderInformation.billTo.locality	
orderInformation.billTo.phoneNumber	
orderInformation.billTo.postalCode	
paymentInformation.tokenizedCard.expirationMonth	
paymentInformation.tokenizedCard.expirationYear	
paymentInformation.tokenizedCard.number	
paymentInformation.tokenizedCard.transactionType	Set the value to `1`.
processingInformation.actionList	Set the value to `TOKEN_CREATE`.
processingInformation.actionTokenTypes	Set the value to `instrumentIdentifier`.
processingInformation.commerceIndicator	Set the value to `internet`, `MOTO`, or a payer authentication value.

processingInformation.paymentSolution	**Set to one of these possible values:** • `001`: Apple Pay • `004`: Cybersource In-App Solution • `005`: Masterpass • `006`: Android Pay • `007`: Chase Pay • `008`: Samsung Pay • `012`: Google Pay • `014`: Mastercard credential-on-file (COF) payment network token • `015`: Visa credential-on-file (COF) payment network token • `027`: Click to Pay • `visacheckout`: Visa Click to Pay.

REST API Example: CIT Unscheduled COF Payment with Enrollable Network Tokens

Request

```
{
  "processingInformation": {
    "actionList": [
      "TOKEN_CREATE"
    ],
    "actionTokenTypes": [
      "instrumentIdentifier"
    ],
    "commerceIndicator": "internet",
    "paymentSolution": "001"
  },
  "paymentInformation": {
    "tokenizedCard": {
      "number": "4111111111111111",
      "expirationMonth": "02",
      "expirationYear": "2025",
      "transactionType": "1"
    }
  },
  "orderInformation": {
    "amountDetails": {
      "totalAmount": "102.21",
      "currency": "USD"
    },
    "billTo": {
      "firstName": "John",
      "lastName": "Smith",
      "address1": "123 Happy St",
      "locality": "Austin",
      "administrativeArea": "TX",
      "postalCode": "78757",
      "country": "US",
      "email": "test@cybs.com",
      "phoneNumber": "444-4444-4444"
    }
  }
}
```

Response to a Successful Request

```
{
  "_links": {
    "authReversal": {
      "method": "POST",
      "href": "/pts/v2/payments/7094060020036241803954/reversals"
    },
    "self": {
      "method": "GET",
      "href": "/pts/v2/payments/7094060020036241803954"
    },
    "capture": {
      "method": "POST",
      "href": "/pts/v2/payments/7094060020036241803954/captures"
    }
  },
  "clientReferenceInformation": {
    "code": "1709406002076"
  },
  "id": "7094060020036241803954",
  "orderInformation": {
    "amountDetails": {
      "authorizedAmount": "102.21",
      "currency": "USD"
    }
  },
  "paymentAccountInformation": {
    "card": {
      "type": "001"
    }
  },
  "paymentInformation": {
    "tokenizedCard": {
      "type": "001"
    },
    "card": {
      "type": "001"
    }
  },
  "pointOfSaleInformation": {
    "terminalId": "111111"
  },
  "processorInformation": {
    "approvalCode": "888888",
    "networkTransactionId": "123456789619999",
    "transactionId": "123456789619999",
    "responseCode": "100",
    "avs": {
      "code": "X",
      "codeRaw": "I1"
    }
  },
  "reconciliationId": "60616704ST7Q27K2",
  "status": "AUTHORIZED",
  "submitTimeUtc": "2024-03-02T19:00:02Z",
  "tokenInformation": {
    "instrumentidentifierNew": false,
    "instrumentIdentifier": {
      "state": "ACTIVE",
      "id": "7010000000016241111"
    }
  }
}
```

Merchant-Initiated Unscheduled COF Payments with PAN

After the initial CIT unscheduled COF payment, subsequent unscheduled COF transactions are merchant-initiated transactions (MITs).

Prerequisites

The first transaction in an unscheduled COF payment is a customer-initiated transaction (CIT). Before you can perform a subsequent merchant-initiated transaction (MIT), you must store the customer's credentials for later use. Before you can store the user's credentials, you must get the customer's consent to store their private information. This process is also known as establishing a relationship with the customer.

Supported Card Types

These are the supported card types for processing credentialed transactions:

- Mastercard
- Visa

Endpoint

Production: `POST https://api.cybersource.com/pts/v2/payments`
Test: `POST https://apitest.cybersource.com/pts/v2/payments`

Required Fields for Authorizing Subsequent MIT Unscheduled COF Payments

These fields are required in a subsequent authorization request for a subsequent unscheduled COF payment:

orderInformation.amountDetails.currency

orderInformation.amountDetails.totalAmount

orderInformation.billTo.address1

orderInformation.billTo.administrativeArea

orderInformation.billTo.country	
orderInformation.billTo.email	
orderInformation.billTo.firstName	
orderInformation.billTo.lastName	
orderInformation.billTo.locality	
orderInformation.billTo.phoneNumber	
orderInformation.billTo.postalCode	
paymentInformation.card.expirationMonth	
paymentInformation.card.expirationYear	
paymentInformation.card.number	
processingInformation. authorizationOptions. initiator. merchantInitiatedTransaction. previousTransactionID	
processingInformation. authorizationOptions. initiator. merchantInitiatedTransaction.reason	**Set the value to `10`.** **Required only for American Express, Discover and Mastercard.**
processingInformation. authorizationOptions. initiator. storedCredentialUsed	**Set the value to `true`.**
processingInformation. authorizationOptions. initiator. type	**Set the value to `merchant`.**
processingInformation. commerceIndicator	**Set the value to `internet`.**

Card-Specific Required Field for Processing a Merchant-Initiated Transactions

Discover

The listed card requires an additional field:

processingInformation. authorizationOptions. initiator. merchantInitiatedTransaction. originalAuthorizedAmount	**Provide the original transaction amount.**

REST Example: Authorizing Subsequent MIT Unscheduled COF Payments

Request

```
{
  "processingInformation": {
    "commerceIndicator": "internet",
    "authorizationOptions": {
      "initiator": {
        "storedCredentialUsed": "true",
        "type": "merchant",
        "merchantInitiatedTransaction": {
          "previousTransactionId": "123456789619999",
          "originalAuthorizedAmount": "100"   <--Discover Only-->
        }
      }
    }
  },
  "orderInformation": {
    "billTo": {
      "firstName": "John",
      "lastName": "Doe",
      "address1": "201 S. Division St.",
      "postalCode": "48104-2201",
      "locality": "Ann Arbor",
      "administrativeArea": "MI",
      "country": "US",
      "phoneNumber": "5554327113",
      "email": "test@cybs.com"
    },
    "amountDetails": {
      "totalAmount": "100.00",
      "currency": "USD"
    }
  },
  "paymentInformation": {
    "card": {
      "expirationYear": "2031",
      "number": "4111xxxxxxxxxxxx",
      "expirationMonth": "12"
    }
  }
}
```

Response to a Successful Request

```
{
  "_links": {
    "authReversal": {
      "method": "POST",
      "href": "/pts/v2/payments/6530824710046809304002/reversals"
    },
    "self": {
      "method": "GET",
      "href": "/pts/v2/payments/6530824710046809304002"
    },
    "capture": {
      "method": "POST",
      "href": "/pts/v2/payments/6530824710046809304002/captures"
    }
  },
  "clientReferenceInformation": {
    "code": "1653082470983"
  },
  "id": "6530824710046809304002",
  "orderInformation": {
    "amountDetails": {
      "authorizedAmount": "100.00",
      "currency": "USD"
    }
  },
  "paymentAccountInformation": {
    "card": {
      "type": "002"
    }
  },
  "paymentInformation": {
    "tokenizedCard": {
      "type": "002"
    },
    "card": {
      "type": "002"
    }
  },
  "pointOfSaleInformation": {
    "terminalId": "111111"
  },
  "processorInformation": {
    "approvalCode": "888888",
    "authIndicator": "1",
    "networkTransactionId": "123456789619999",
    "transactionId": "123456789619999",
    "responseCode": "100",
    "avs": {
      "code": "X",
      "codeRaw": "I1"
    }
  },
  "reconciliationId": "79710341A39WTT5W",
  "status": "AUTHORIZED",
  "submitTimeUtc": "2022-05-20T21:34:31Z"
}
```

Merchant-Initiated Unscheduled COF Payments with TMS

After the customer-initiated unscheduled COF payment, you can send merchant-initiated unscheduled COF payments using one or more TMS token types:

Customer

Customer tokens store one or more customer payment instrument tokens and shipping address tokens.

Including a customer token eliminates the need to include billing information, card information, and the previous transaction's ID.

```
"paymentInformation": {
  "customer": {
    "id": "07C9CA98022DA498E063A2598D0AA400"
  }
}
```

For more information about this TMS token type, see *Customer Tokens* in the Token Management Service Developer Guide.

Payment Instrument

Payment instrument tokens store an instrument identifier token, card information, and billing information. Payment instruments are not linked to a customer token.

Including a payment instrument eliminates the need to include billing information, card information, and the previous transaction's ID.

```
"paymentInformation": {
  "paymentInstrument": {
    "id": "07CA24EF20F9E2C9E063A2598D0A8565"
  }
}
```

For more information about this TMS token type, see *Payment Instrument Token* in the Token Management Service Developer Guide.

Instrument Identifier

Instrument identifier tokens store only a PAN. Including an instrument identifier eliminates the need to include a PAN and the previous transaction's ID.

```
"paymentInformation": {
  "instrumentIdentifier": {
    "id": "7010000000016241111"
  }
}
```

For more information about this TMS token type, see *Instrument Identifier Token* in the Token Management Service Developer Guide.

Prerequisites

The first transaction in an unscheduled COF payment is a customer-initiated transaction (CIT). Before you can perform a subsequent merchant-initiated transaction (MIT), you must store the customer's credentials for later use. Before you can store the user's credentials, you must get the customer's consent to store their private information. This process is also known as establishing a relationship with the customer.

Supported Card Types

These are the supported card types for processing credentialed transactions:

- Mastercard
- Visa

Endpoint

Production: `POST https://api.cybersource.com/pts/v2/payments`
Test: `POST https://apitest.cybersource.com/pts/v2/payments`

Required Fields for MIT Unscheduled COF Payments with TMS

Include these Required Fields

orderInformation.amountDetails.currency	
orderInformation.amountDetails.totalAmount	
paymentInformation.[tokentype].id	Where **[tokentype]** is the TMS token type you are using: • **customer** • **instrumentIdentifier** • **paymentInstrument**
processingInformation. authorizationOptions. initiator. merchantInitiatedTransaction.reason	Set the value to `10`. Required only for American Express, Discover, and Mastercard.
processingInformation.commerceIndicator	Set the value to `internet`.

Instrument Identifier Required Fields

If you are using the **paymentInformation.instrumentIdentifier.id** token, include these required fields in addition to the required fields listed above.

orderInformation.billTo.address1

orderInformation.billTo.administrativeArea

orderInformation.billTo.country

orderInformation.billTo.email

orderInformation.billTo.firstName

orderInformation.billTo.lastName

orderInformation.billTo.locality

orderInformation.billTo.phoneNumber

orderInformation.billTo.postalCode

paymentInformation.card.expirationMonth

paymentInformation.card.expirationYear

Card-Specific Field

The listed card type requires an additional field.

Discover	**processingInformation.authorizationOptions.initiator. merchantInitiatedTransaction.originalAuthorizedAmou** **Provide the original transaction amount.**

Example: MIT Unscheduled COF Payment with TMS Instrument Identifier

Request

```
{
  "processingInformation": {
    "commerceIndicator": "internet"
  },
  "paymentInformation": {
    "card": {
      "expirationMonth": "12",
      "expirationYear": "2031"
    },
    "instrumentIdentifier": {
      "id": "7010000000016241111"
    }
  },
  "orderInformation": {
    "amountDetails": {
      "totalAmount": "102.21",
      "currency": "USD"
    },
    "billTo": {
      "firstName": "John",
      "lastName": "Doe",
      "address1": "1 Market St",
      "locality": "san francisco",
      "administrativeArea": "CA",
      "postalCode": "94105",
      "country": "US",
      "email": "test@cybs.com",
      "phoneNumber": "4158880000"
    }
  }
}
```

Response to a Successful Request

```
{
  "_links": {
    "authReversal": {
      "method": "POST",
      "href": "/pts/v2/payments/6976892714556134003954/reversals"
    },
    "self": {
      "method": "GET",
      "href": "/pts/v2/payments/6976892714556134003954"
    },
    "capture": {
      "method": "POST",
      "href": "/pts/v2/payments/6976892714556134003954/captures"
    }
  },
  "clientReferenceInformation": {
    "code": "1697689271513"
  },
  "id": "6976892714556134003954",
  "orderInformation": {
    "amountDetails": {
      "authorizedAmount": "102.21",
      "currency": "USD"
    }
  },
  "paymentAccountInformation": {
    "card": {
      "type": "001"
    }
  },
  "paymentInformation": {
    "tokenizedCard": {
      "type": "001"
    },
    "instrumentIdentifier": {
      "id": "7010000000016241111",
      "state": "ACTIVE"
    },
    "card": {
      "type": "001"
    }
  },
  "pointOfSaleInformation": {
    "terminalId": "111111"
  },
  "processingInformation": {
    "paymentSolution": "015"
  },
  "processorInformation": {
    "paymentAccountReferenceNumber": "V0010013022298169667504231315",
    "approvalCode": "888888",
    "networkTransactionId": "123456789619999",
    "transactionId": "123456789619999",
    "responseCode": "100",
    "avs": {
      "code": "X",
      "codeRaw": "I1"
    }
  },
  "reconciliationId": "62699554NNMR6X7R",
  "status": "AUTHORIZED",
  "submitTimeUtc": "2023-10-19T04:21:11Z"
}
```

Example: MIT Unscheduled COF Payment with TMS Payment Instrument

Request

```
{
  "processingInformation": {
    "commerceIndicator": "internet"
  },
  "paymentInformation": {
    "paymentInstrument": {
      "id": "080AE120369A7947E063A2598D0A718F"
    }
  },
  "orderInformation": {
    "amountDetails": {
      "totalAmount": "102.21",
      "currency": "USD"
    }
  }
}
```

Response to a Successful Request

```
{
  "_links": {
    "authReversal": {
      "method": "POST",
      "href": "/pts/v2/payments/6976891300676431103955/reversals"
    },
    "self": {
      "method": "GET",
      "href": "/pts/v2/payments/6976891300676431103955"
    },
    "capture": {
      "method": "POST",
      "href": "/pts/v2/payments/6976891300676431103955/captures"
    }
  },
  "clientReferenceInformation": {
    "code": "1697689130124"
  },
  "id": "6976891300676431103955",
  "orderInformation": {
    "amountDetails": {
      "authorizedAmount": "102.21",
      "currency": "USD"
    }
  },
  "paymentAccountInformation": {
    "card": {
      "type": "001"
    }
  },
  "paymentInformation": {
    "tokenizedCard": {
      "type": "001"
    },
    "instrumentIdentifier": {
      "id": "7010000000016241111",
      "state": "ACTIVE"
    },
    "paymentInstrument": {
      "id": "080AE120369A7947E063A2598D0A718F"
    },
    "card": {
      "type": "001"
    }
  },
  "pointOfSaleInformation": {
    "terminalId": "111111"
  },
  "processingInformation": {
    "paymentSolution": "015"
  },
  "processorInformation": {
    "paymentAccountReferenceNumber": "V0010013022298169667504231315",
    "approvalCode": "888888",
    "networkTransactionId": "123456789619999",
    "transactionId": "123456789619999",
    "responseCode": "100",
    "avs": {
      "code": "X",
      "codeRaw": "I1"
    }
  },
  "reconciliationId": "62699372XNMR85HS",
  "status": "AUTHORIZED",
  "submitTimeUtc": "2023-10-19T04:18:50Z"
}
```

Example: MIT Unscheduled COF Payment with TMS Customer

Request

```
{
  "processingInformation": {
    "commerceIndicator": "internet"
  },
  "paymentInformation": {
    "customer": {
      "id": "080AC9AB60C92AA2E063A2598D0A0C74"
    }
  },
  "orderInformation": {
    "amountDetails": {
      "totalAmount": "102.21",
      "currency": "USD"
    }
  }
}
```

Response to a Successful Request

```
{
 "_links": {
  "authReversal": {
   "method": "POST",
   "href": "/pts/v2/payments/6976889582016147703955/reversals"
  },
  "self": {
   "method": "GET",
   "href": "/pts/v2/payments/6976889582016147703955"
  },
  "capture": {
   "method": "POST",
   "href": "/pts/v2/payments/6976889582016147703955/captures"
  }
 },
 "clientReferenceInformation": {
  "code": "1697688958296"
 },
 "id": "6976889582016147703955",
 "orderInformation": {
  "amountDetails": {
   "authorizedAmount": "102.21",
   "currency": "USD"
  }
 },
 "paymentAccountInformation": {
  "card": {
   "type": "001"
  }
 },
 "paymentInformation": {
  "tokenizedCard": {
   "type": "001"
  },
  "instrumentIdentifier": {
   "id": "7010000000016241111",
   "state": "ACTIVE"
  },
  "paymentInstrument": {
   "id": "080AE6DB37B09557E063A2598D0AA4C9"
  },
  "card": {
   "type": "001"
  },
  "customer": {
   "id": "080AC9AB60C92AA2E063A2598D0A0C74"
  }
 },
 "pointOfSaleInformation": {
  "terminalId": "111111"
 },
 "processingInformation": {
    "paymentSolution": "015"
  },
  "processorInformation": {
    "paymentAccountReferenceNumber": "V0010013022298169667504231315",
    "approvalCode": "888888",
    "networkTransactionId": "123456789619999",
    "transactionId": "123456789619999",
    "responseCode": "100",
    "avs": {
      "code": "X",
      "codeRaw": "I1"
    }
  },
  "reconciliationId": "62699842BNN13VA0",
  "status": "AUTHORIZED",
  "submitTimeUtc": "2023-10-19T04:15:58Z"
}
```

Token Management Service Processing

This section provides the information you need in order to process Token Management Service authorization and credit transactions.

Important

Due to mandates from the Reserve Bank of India, Indian merchants cannot store personal account numbers (PANs). Use network tokens instead. For more information on network tokens, see the Network Tokenization section of the *Token Management Service Guide.*

Additional Resources for TMS

For more information, see these guides:

- *Token Management Service Developer Guide*
- *API field reference guide for the REST API*
- Github repositories: *https://github.com/Cybersource*

Authorizing a Payment with a Customer Token

This section provides the information you need to authorize a payment with a customer token.

Endpoint

Production: `POST https://api.cybersource.com/pts/v2/payments`
Test: `POST https://apitest.cybersource.com/pts/v2/payments`

Production in India: `POST https://api.in.cybersource.com/pts/v2/payments`

Required Fields for Authorizing a Payment with a Customer Token

clientReferenceInformation.code	
orderInformation.amountDetails.currency	
orderInformation.amountDetails.totalAmount	
paymentInformation.customer.id	**Set to the ID of the customer token you want to use.**

Related Information

- *API field reference guide for the REST API*

REST Example: Authorizing a Payment with a Customer Token

Request

```
{
 "clientReferenceInformation": {
  "code": "12345678"
 },
  "paymentInformation": {
    "customer": {
      "id": "F45FB3E443AC3C57E053A2598D0A9CFF"
    }

  },
  "orderInformation": {
    "amountDetails": {
      "currency": "USD",
      "totalAmount": "10.00"
    }
  }
}
```

Response to a Successful Request

The request response returns the payment instrument and shipping address IDs that are used as the customer's defaults.

```
{
 "_links": {
  "authReversal": {
   "method": "POST",
   "href": "/pts/v2/payments/7055928871556818104953/reversals"
  },
  "self": {
   "method": "GET",
   "href": "/pts/v2/payments/7055928871556818104953"
  },
  "capture": {
      "method": "POST",
      "href": "/pts/v2/payments/7055928871556818104953/captures"
    }
  },
  "clientReferenceInformation": {
    "code": "12345678"
  },
  "id": "7055928871556818104953",
  "orderInformation": {
    "amountDetails": {
      "authorizedAmount": "10.00",
      "currency": "USD"
    }
  },
  "paymentAccountInformation": {
    "card": {
      "type": "001"
    }
  },
  "paymentInformation": {
    "tokenizedCard": {
      "type": "001"
    },
    "instrumentIdentifier": {
      "id": "7010000000016241111",
      "state": "ACTIVE"
    },
    "shippingAddress": {
      "id": "0F35F0D99AD088B5E063A2598D0AE066"
    },
    "paymentInstrument": {
      "id": "0F35E9CFEA463E34E063A2598D0A3FC2"
    },
    "card": {
      "type": "001"
    },
    "customer": {
      "id": "B21E6717A6F03479E05341588E0A303F"
    }
  },
  "pointOfSaleInformation": {
    "terminalId": "111111"
  },
  "processorInformation": {
    "approvalCode": "888888",
    "networkTransactionId": "123456789619999",
    "transactionId": "123456789619999",
    "responseCode": "100",
    "avs": {
      "code": "X",
      "codeRaw": "I1"
    }
  },
  "reconciliationId": "67467352CRIISD1G",
  "status": "AUTHORIZED",
  "submitTimeUtc": "2024-01-18T15:48:07Z"
```

```
}
```

REST Example: Authorizing a Payment Using a Customer Token Linked to a Network Token

Request

```
{
  "clientReferenceInformation": {
    "code": "12345678"
  },
  "paymentInformation": {
    "customer": {
      "id": "F60328413BAB09A4E053AF598E0A33DB"
    }
  },
  "orderInformation": {
    "amountDetails": {
      "totalAmount": "102.21",
      "currency": "USD"
    }
  }
}
```

Response to a Successful Request

The request response returns the payment instrument and shipping address IDs that are used as the customer's defaults.

```
{
  "_links": {
    "authReversal": {
      "method": "POST",
      "href": "/pts/v2/payments/6778647071126384904953/reversals"
    },
    "self": {
      "method": "GET",
      "href": "/pts/v2/payments/6778647071126384904953"
    },
    "capture": {
      "method": "POST",
      "href": "/pts/v2/payments/6778647071126384904953/captures"
    }
  },
  "clientReferenceInformation": {
    "code": "TC50171_3"
  },
  "id": "6778647071126384904953",
  "issuerInformation": {
    "responseRaw": "0110322000000E100002000....."
  },
  "orderInformation": {
    "amountDetails": {
      "authorizedAmount": "102.21",
      "currency": "USD"
    }
  },
  "paymentAccountInformation": {
    "card": {
      "type": "002"
    }
  },
  "paymentInformation": {
    "tokenizedCard": {
      "type": "002"
    },
    "instrumentIdentifier": {
      "id": "7020000000010603216",
      "state": "ACTIVE"
    },
    "shippingAddress": {
      "id": "F60328413BAE09A4E053AF598E0A33DB"
    },
    "paymentInstrument": {
      "id": "F6032841BE33098EE053AF598E0AB0A5"
    },
    "card": {
      "type": "002"
    },
    "customer": {
      "id": "F60328413BAB09A4E053AF598E0A33DB"
    }
  },
  "pointOfSaleInformation": {
    "terminalId": "08244117"
  },  "processingInformation": {    "paymentSolution": "014"  },
  "processorInformation": {
    "paymentAccountReferenceNumber": "50015OU4U5UYXLV127XTONYN49CL1",
    "merchantNumber": "000844028303882",
    "approvalCode": "831000",
    "networkTransactionId": "0602MCC603474",
    "transactionId": "0602MCC603474",
    "responseCode": "00",
    "avs": {
      "code": "Y",
      "codeRaw": "Y"
    }
  },
  "reconciliationId": "EUHW1EMHIZ3O",
  "status": "AUTHORIZED",
  "submitTimeUtc": "2023-03-03T17:31:48Z"
}
```

Authorizing a Payment with a Non-Default Shipping Address

This section provides the information you need in order to make a payment with a non-default shipping address.

Endpoint

Production: `POST https://api.cybersource.com/pts/v2/payments`
Test: `POST https://apitest.cybersource.com/pts/v2/payments`
Production in India: `POST https://api.in.cybersource.com/pts/v2/payments`

Required Fields for Authorizing a Payment with a Non-Default Shipping Address

clientReferenceInformation.code

orderInformation.amountDetails.currency

orderInformation.amountDetails.totalAmount

paymentInformation.customer.id	**Set to the ID of the customer token you want to use.**
paymentInformation.shippingAddress.id	**Set to the ID of the shipping address token you want to use.**

Related Information

- *API field reference guide for the REST API*

REST Example: Authorizing a Payment with a Non-Default Shipping Address

Request

```
{
 "clientReferenceInformation": {
  "code": "12345678"
 },
  "paymentInformation": {
    "customer": {
      "id": "F45FB3E443AC3C57E053A2598D0A9CFF"
    },
    "shippingAddress": {
      "id": "F45FD8DE51B99E9CE053A2598D0AFDFA"
    }
  },
  "orderInformation": {
    "amountDetails": {
      "currency": "USD",
      "totalAmount": "10.00"
    }
  }
}
```

Response to a Successful Request

```
{
  "_links": {
    "authReversal": {
      "method": "POST",
      "href": "/pts/v2/payments/7055949037316786904953/reversals"
    },
    "self": {
      "method": "GET",
      "href": "/pts/v2/payments/7055949037316786904953"
    },
    "capture": {
      "method": "POST",
      "href": "/pts/v2/payments/7055949037316786904953/captures"
    }
  },
  "clientReferenceInformation": {
    "code": "12345678"
  },
  "id": "7055949037316786904953",
  "orderInformation": {
    "amountDetails": {
      "authorizedAmount": "10.00",
      "currency": "USD"
    }
  },
  "paymentAccountInformation": {
    "card": {
      "type": "001"
    }
  },
  "paymentInformation": {
    "tokenizedCard": {
      "type": "001"
    },
    "instrumentIdentifier": {
      "id": "7030000000014831523",
      "state": "ACTIVE"
    },
    "shippingAddress": {
      "id": "F45FD8DE51B99E9CE053A2598D0AFDFA"
    },
    "paymentInstrument": {
      "id": "F45FE45E7993C7DBE053A2598D0AED19"
    },
    "card": {
      "type": "001"
    },
    "customer": {
      "id": "F45FB3E443AC3C57E053A2598D0A9CFF"
    }
  },
  "pointOfSaleInformation": {
    "terminalId": "111111"
  },
  "processorInformation": {
    "approvalCode": "888888",
    "networkTransactionId": "123456789619999",
    "transactionId": "123456789619999",
    "responseCode": "100",
    "avs": {
      "code": "X",
      "codeRaw": "I1"
    }
  },
  "reconciliationId": "674679208RIKQ52K",
  "status": "AUTHORIZED",
  "submitTimeUtc": "2024-01-18T16:21:44Z"
}
```

Authorizing a Payment with a Non-Default Payment Instrument

This section provides the information you need in order to authorize a payment with a non-default payment instrument.

Endpoint

Production: `POST https://api.cybersource.com/pts/v2/payments`
Test: `POST https://apitest.cybersource.com/pts/v2/payments`
Production in India: `POST https://api.in.cybersource.com/pts/v2/payments`

Required Fields for Authorizing a Payment with a Non-Default Payment Instrument

clientReferenceInformation.code	
orderInformation.amountDetails.currency	
orderInformation.amountDetails.totalAmount	
paymentInformation.paymentInstrument.id	**Set to the ID of the payment instrument token you want to use.**

Related Information

- *API field reference guide for the REST API*

Optional Fields for Authorizing a Payment with a Non-Default Payment Instrument

You can use these optional fields to include additional information when authorizing a payment with a non-default payment instrument.

orderInformation.amountDetails.currency

orderInformation.amountDetails.totalAmount

orderInformation.billTo.address1

orderInformation.billTo.administrativeArea

orderInformation.billTo.country

orderInformation.billTo.email

orderInformation.billTo.firstName

orderInformation.billTo.lastName

orderInformation.billTo.locality

orderInformation.billTo.postalCode

paymentInformation.card.expirationMonth

paymentInformation.card.expirationYear

paymentInformation.card.number

paymentInformation.card.type

Related Information

- *API field reference guide for the REST API*

REST Example: Authorizing a Payment with a Non-Default Payment Instrument

Request

```
{
  "clientReferenceInformation": {
    "code": "12345678"
  },
    "paymentInformation": {
      "paymentInstrument": {
        "id": "0F3BB131F8143A58E063A2598D0AB921"
      }
    },
    "orderInformation": {
      "amountDetails": {
        "currency": "USD",
        "totalAmount": "10.00"
      }
  }
}
```

Response to a Successful Request

```
{
 "_links": {
  "authReversal": {
   "method": "POST",
   "href": "/pts/v2/payments/7055952648586653304951/reversals"
  },
  "self": {
   "method": "GET",
   "href": "/pts/v2/payments/7055952648586653304951"
  },
  "capture": {
   "method": "POST",
   "href": "/pts/v2/payments/7055952648586653304951/captures"
  }
 },
 "clientReferenceInformation": {
  "code": "12345678"
 },
 "id": "7055952648586653304951",
 "orderInformation": {
  "amountDetails": {
   "authorizedAmount": "10.00",
   "currency": "USD"
  }
 },
 "paymentAccountInformation": {
  "card": {
   "type": "001"
  }
 },
 "paymentInformation": {
  "tokenizedCard": {
   "type": "001"
  },
  "instrumentIdentifier": {
   "id": "7010000000016241111",
   "state": "ACTIVE"
  },
  "paymentInstrument": {
   "id": "0F3BB131F8143A58E063A2598D0AB921"
  },
  "card": {
   "type": "001"
  }
 },
 "pointOfSaleInformation": {
  "terminalId": "111111"
 },
 "processorInformation": {
  "approvalCode": "888888",
  "networkTransactionId": "123456789619999",
    "transactionId": "123456789619999",
    "responseCode": "100",
    "avs": {
      "code": "X",
      "codeRaw": "I1"
    }
  },
  "reconciliationId": "67468244CRIL0U0Y",
  "status": "AUTHORIZED",
  "submitTimeUtc": "2024-01-18T16:27:45Z"
}
```

Authorizing a Payment with a Payment Instrument

This section provides the information you need in order to authorize a payment with a payment instrument.

Endpoint

Production: `POST https://api.cybersource.com/pts/v2/payments`
Test: `POST https://apitest.cybersource.com/pts/v2/payments`
Production in India: `POST https://api.in.cybersource.com/pts/v2/payments`

Required Fields for Authorizing a Payment with a Payment Instrument

clientReferenceInformation.code	
orderInformation.amountDetails.currency	
orderInformation.amountDetails.totalAmount	
paymentInformation.paymentInstrument.id	**Set to the ID of the payment instrument token you want to use.**

Related Information

- *API field reference guide for the REST API*

Optional Fields for Authorizing a Payment with a Payment Instrument

You can use these optional fields to include additional information when authorizing a payment with a payment instrument.

orderInformation.amountDetails.currency *orderInformation.amountDetails.totalAmount*

orderInformation.billTo.address1

orderInformation.billTo.administrativeArea

orderInformation.billTo.country

orderInformation.billTo.email

orderInformation.billTo.firstName

orderInformation.billTo.lastName

orderInformation.billTo.locality

orderInformation.billTo.postalCode

paymentInformation.card.expirationMonth

paymentInformation.card.expirationYear

paymentInformation.card.number

paymentInformation.card.type

Related Information

- *API field reference guide for the REST API*

REST Example: Authorizing a Payment with a Payment Instrument

Request

```
{
 "clientReferenceInformation": {
  "code": "12345678"
 },
  "paymentInformation": {
    "paymentInstrument": {
      "id": "F4D5E715F7BD9910E053A2598D0A7278"
    }
  },
  "orderInformation": {
    "amountDetails": {
      "currency": "USD",
      "totalAmount": "10.00"
    }
  }
}
```

Response to a Successful Request

```
{
  "_links": {
    "authReversal": {
      "method": "POST",
      "href": "/pts/v2/payments/6765713628736138103955/reversals"
    },
    "self": {
      "method": "GET",
      "href": "/pts/v2/payments/6765713628736138103955"
    },
    "capture": {
      "method": "POST",
      "href": "/pts/v2/payments/6765713628736138103955/captures"
    }
  },
  "clientReferenceInformation": {
    "code": "12345678"
  },
  "id": "6765713628736138103955",
  "orderInformation": {
    "amountDetails": {
      "authorizedAmount": "10.00",
      "currency": "USD"
    }
  },
  "paymentAccountInformation": {
    "card": {
      "type": "001"
    }
  },
  "paymentInformation": {
    "tokenizedCard": {
      "type": "001"
    },
    "instrumentIdentifier": {
      "id": "7010000000016241111",
      "state": "ACTIVE"
    },
    "paymentInstrument": {
      "id": "F4D5E715F7BD9910E053A2598D0A7278"
    },
    "card": {
      "type": "001"
    },
    "customer": {
      "id": "F4D5E715F75E9910E053A2598D0A7278"
    }
  },
  "pointOfSaleInformation": {
    "terminalId": "111111"
  },
  "processorInformation": {
    "approvalCode": "888888",
    "networkTransactionId": "123456789619999",
    "transactionId": "123456789619999",
    "responseCode": "100",
    "avs": {
      "code": "X",
      "codeRaw": "I1"
    }
  },
    "reconciliationId": "60561224BE37KN5W",
    "status": "AUTHORIZED",
    "submitTimeUtc": "2023-02-16T18:16:03Z"
}
```

Authorize a Payment with an Instrument Identifier

This section provides the information you need in order to authorize a payment with an instrument identifier token.

Endpoint

Production: `POST https://api.cybersource.com/pts/v2/payments`
Test: `POST https://apitest.cybersource.com/pts/v2/payments`
Production in India: `POST https://api.in.cybersource.com/pts/v2/payments`

Required Fields for Authorizing a Payment with an Instrument Identifier

clientReferenceInformation.code	
orderInformation.amountDetails.currency	
orderInformation.amountDetails.totalAmount	
paymentInformation.instrumentIdentifier.id	**Set to the ID of the instrument identifier token you want to use.**

Related Information

- *API field reference guide for the REST API*

REST Example: Authorizing a Payment with an Instrument Identifier

Request

```
{
  "clientReferenceInformation": {
    "code": "12345678"
  },
    "paymentInformation": {
      "instrumentIdentifier": {
        "id": "7010000000016241111"
      }
    },
    "orderInformation": {
      "amountDetails": {
        "currency": "USD",
        "totalAmount": "10.00"
      }
    }
}
```

Response to a Successful Request

```
{
  "_links": {
    "authReversal": {
      "method": "POST",
      "href": "/pts/v2/payments/7055955288186053404953/reversals"
    },
    "self": {
      "method": "GET",
      "href": "/pts/v2/payments/7055955288186053404953"
    },
    "capture": {
      "method": "POST",
      "href": "/pts/v2/payments/7055955288186053404953/captures"
    }
  },
  "clientReferenceInformation": {
    "code": "12345678"
  },
  "id": "7055955288186053404953",
  "orderInformation": {
    "amountDetails": {
      "authorizedAmount": "10.00",
      "currency": "USD"
    }
  },
  "paymentAccountInformation": {
    "card": {
      "type": "001"
    }
  },
  "paymentInformation": {
    "tokenizedCard": {
      "type": "001"
    },
    "instrumentIdentifier": {
      "id": "7010000000016241111",
      "state": "ACTIVE"
    },
    "card": {
      "type": "001"
    }
  },
  "pointOfSaleInformation": {
    "terminalId": "111111"
  },
  "processorInformation": {
    "approvalCode": "888888",
    "networkTransactionId": "123456789619999",
    "transactionId": "123456789619999",
    "responseCode": "100",
    "avs": {
      "code": "1"
    }
  },
  "reconciliationId": "67468271CRIL0U24",
  "status": "AUTHORIZED",
  "submitTimeUtc": "2024-01-18T16:32:09Z"
}
```

REST Example: Authorizing a Payment with an Instrument Identifier While Creating TMS Tokens

Request

```
{
  "clientReferenceInformation": {
    "code": "TC50171_3"
  },
  "processingInformation": {
    "actionList": [
      "TOKEN_CREATE"
    ],
    "actionTokenTypes": [
      "customer",
      "paymentInstrument",
      "shippingAddress"
    ]
  },
  "paymentInformation": {
    "instrumentIdentifier": {
      "id": "7010000000016241111"
    }
  },
  "orderInformation": {
    "amountDetails": {
      "totalAmount": "102.21",
      "currency": "USD"
    },
    "billTo": {
      "firstName": "John",
      "lastName": "Doe",
      "address1": "1 Market St",
      "locality": "san francisco",
      "administrativeArea": "CA",
      "postalCode": "94105",
      "country": "US",
      "email": "test@cybs.com",
      "phoneNumber": "4158880000"
    },
    "shipTo": {
      "firstName": "John",
      "lastName": "Doe",
      "address1": "1 Market St",
      "locality": "san francisco",
      "administrativeArea": "CA",
      "postalCode": "94105",
      "country": "US"
    }
  }
}
```

Response to a Successful Request

```
{
  "_links": {
    "authReversal": {
      "method": "POST",
      "href": "/pts/v2/payments/7114679840376687203955/reversals"
    },
    "self": {
      "method": "GET",
      "href": "/pts/v2/payments/7114679840376687203955"
    },
    "capture": {
      "method": "POST",
      "href": "/pts/v2/payments/7114679840376687203955/captures"
    }
  },
  "clientReferenceInformation": {
    "code": "TC50171_3"
  },
  "id": "7114679840376687203955",
  "orderInformation": {
    "amountDetails": {
      "authorizedAmount": "102.21",
      "currency": "USD"
    }
  },
  "paymentAccountInformation": {
    "card": {
      "type": "001"
    }
  },
  "paymentInformation": {
    "tokenizedCard": {
      "type": "001"
    },
    "instrumentIdentifier": {
      "id": "7010000000016241111",
      "state": "ACTIVE"
    },
    "card": {
      "type": "001"
    }
  },
  "pointOfSaleInformation": {
    "terminalId": "111111"
  },
  "processorInformation": {
    "approvalCode": "888888",
    "networkTransactionId": "123456789619999",
    "transactionId": "123456789619999",
    "responseCode": "100",
    "avs": {
      "code": "X",
      "codeRaw": "I1"
    }
  },
  "reconciliationId": "623971212U7PN4IU",
  "status": "AUTHORIZED",
  "submitTimeUtc": "2024-03-26T15:46:24Z",
  "tokenInformation": {
    "shippingAddress": {
      "id": "14930C904FC4D97BE063A2598D0AE0F1"
    },
    "paymentInstrument": {
      "id": "149310A4A924E911E063A2598D0A47AD"
    },
    "customer": {
      "id": "14930C904FC1D97BE063A2598D0AE0F1"
    }
  }
}
```

Authorize a Payment While Ignoring Network Token

This section shows you how to authorize a payment ignoring a network token.

Endpoint

Test: `POST https://apitest.cybersource.com/pts/v2/payments`
Production: `POST https://api.cybersource.com/pts/v2/payments`
Production in India: `POST https://api.in.cybersource.com/pts/v2/payments`

Required Fields for Authorizing a Payment While Ignoring Network Token

clientReferenceInformation.code

paymentInformation.customer.id

paymentInformation.paymentInformation.id

paymentInformation.shippingAddress.id

orderInformation.amountDetails.currency

orderInformation.amountDetails.totalAmount

processingInformation.capture

processingInformation.commerceIndicator	
tokenInformation.networkTokenOption	**Set value to `ignore`.**

Related Information

- *API Field Reference for the REST API*

REST Example: Authorizing a Payment While Ignoring Network Token

Request

```
{
  "clientReferenceInformation": {
    "code": "RTS-Auth"
  },
  "paymentInformation": {
    "card": {
      "expirationYear": "2031",
      "expirationMonth": "12",
      "type": "001"
    },
    "instrumentIdentifier": {
      "id": "7010000000016241111"
    }
  },
  "orderInformation": {
    "amountDetails": {
      "currency": "USD",
      "totalAmount": "1.00"
    }
  },
  "processingInformation": {
    "capture": "false",
    "commerceIndicator": "internet"
  },
  "tokenInformation": {
    "networkTokenOption": "ignore"
  }
}
```

Response to a Successful Request

```
{
  "_links": {
    "authReversal": {
      "method": "POST",
      "href": "/pts/v2/payments/6769913443166412604951/reversals"
    },
    "self": {
      "method": "GET",
      "href": "/pts/v2/payments/6769913443166412604951"
    },
    "capture": {
      "method": "POST",
        "href": "/pts/v2/payments/6769913443166412604951/captures"
      }
    },
    "clientReferenceInformation": {
      "code": "RTS-Auth"
    },
    "id": "6769913443166412604951",
    "orderInformation": {
      "amountDetails": {
        "authorizedAmount": "1.00",
        "currency": "USD"
      }
    },
    "paymentAccountInformation": {
      "card": {
        "type": "001"
      }
    },
    "paymentInformation": {
      "tokenizedCard": {
        "type": "001"
      },
      "instrumentIdentifier": {
        "id": "7030000000014911515",
        "state": "ACTIVE"
      },
      "shippingAddress": {
        "id": "F537CE8DBA2F032CE053AF598E0A64F2"
      },
      "paymentInstrument": {
        "id": "F537E3D12322416EE053AF598E0AD771"
      },
      "card": {
        "type": "001"
      },
      "customer": {
        "id": "F537CE8DBA2C032CE053AF598E0A64F2"
      }
    },
    "pointOfSaleInformation": {
      "terminalId": "111111"
    },
    "processorInformation": {
      "paymentAccountReferenceNumber": "V0010013019326121174070050420",
      "approvalCode": "888888",
      "networkTransactionId": "123456789619999",
      "transactionId": "123456789619999",
      "responseCode": "100",
      "avs": {
        "code": "X",
        "codeRaw": "I1"
      }
    },
    "reconciliationId": "744295942E2LY3F8",
    "status": "AUTHORIZED",
    "submitTimeUtc": "2023-02-21T14:55:44Z"
```

```
}
```

Authorizing a Payment with a Legacy Token

This section shows you how to authorize a payment with a legacy token.

Endpoint

Production: `POST https://api.cybersource.com/pts/v2/payments`
Test: `POST https://apitest.cybersource.com/pts/v2/payments`

Required Fields for Authorizing a Payment with a Legacy Token

clientReferenceInformation.code	
paymentInformation.legacyToken.id	**Include the ID of the legacy token you want to use to authorize a payment.**
orderInformation.amountDetails.currency	
orderInformation.amountDetails.totalAmount	

Related Information

- *API field reference guide for the REST API*

REST Example: Authorizing a Payment with a Legacy Token

Request

```
{
 "clientReferenceInformation": {
  "code": "12345678"
 },
 "paymentInformation": {
  "legacyToken": {
   "id": "B21E6717A6F03479E05341588E0A303F"
  }
 },
 "orderInformation": {
  "amountDetails": {
   "totalAmount": "22.00",
   "currency": "USD"
  }
 }
}
```

Response to a Successful Request

```
{
 "_links": {
  "authReversal": {
      "method": "POST",
      "href": "/pts/v2/payments/7055956342476789004951/reversals"
    },
    "self": {
      "method": "GET",
      "href": "/pts/v2/payments/7055956342476789004951"
    },
    "capture": {
      "method": "POST",
      "href": "/pts/v2/payments/7055956342476789004951/captures"
    }
  },
  "clientReferenceInformation": {
    "code": "12345678"
  },
  "id": "7055956342476789004951",
  "orderInformation": {
    "amountDetails": {
      "authorizedAmount": "22.00",
      "currency": "USD"
    }
  },
  "paymentAccountInformation": {
    "card": {
      "type": "001"
    }
  },
  "paymentInformation": {
    "tokenizedCard": {
      "type": "001"
    },
    "card": {
      "type": "001"
    }
  },
  "pointOfSaleInformation": {
    "terminalId": "111111"
  },
  "processorInformation": {
    "approvalCode": "888888",
    "networkTransactionId": "123456789619999",
    "transactionId": "123456789619999",
    "responseCode": "100",
    "avs": {
      "code": "X",
      "codeRaw": "I1"
    }
  },
  "reconciliationId": "67468431FRIIS246",
  "status": "AUTHORIZED",
  "submitTimeUtc": "2024-01-18T16:33:54Z"
}
```

Making a Credit with a Customer Token

This section shows you how to make a credit with a customer token.

Endpoint

Test: `POST https://apitest.cybersource.com/pts/v2/credits`
Production: `POST https://api.cybersource.com/pts/v2/credits`
Production in India: `POST https://api.in.cybersource.com/pts/v2/credits`

Required Fields for Making a Credit with a Customer Token

clientReferenceInformation.code	
orderInformation.amountDetails.currency	
orderInformation.amountDetails.totalAmount	
paymentInformation.customer.id	**Set to the ID of the customer token you want to use.**

Related Information

- *API field reference guide for the REST API*

REST Example: Making a Credit with a Customer Token

Request

```
{
 "clientReferenceInformation": {
  "code": "12345678"
 },
  "paymentInformation": {
    "customer": {
      "id": "F45FB3E443AC3C57E053A2598D0A9CFF"
    }
  },
  "orderInformation": {
    "amountDetails": {
      "currency": "USD",
      "totalAmount": "10.00"
    }
  }
}
```

Response to a Successful Request

```
{
 "_links": {
  "void": {
   "method": "POST",
   "href": "/pts/v2/credits/7055967677826132904951/voids"
  },
  "self": {
   "method": "GET",
   "href": "/pts/v2/credits/7055967677826132904951"
  }
 },
 "clientReferenceInformation": {
  "code": "12345678"
 },
 "creditAmountDetails": {
  "currency": "USD",
  "creditAmount": "10.00"
 },
 "id": "7055967677826132904951",
 "orderInformation": {
  "amountDetails": {
   "currency": "USD"
  }
 },
 "paymentAccountInformation": {
  "card": {
   "type": "001"
  }
 },
 "paymentInformation": {
  "tokenizedCard": {
   "type": "001"
  },
  "instrumentIdentifier": {
   "id": "7030000000014831523",
   "state": "ACTIVE"
  },
  "shippingAddress": {
   "id": "F45FD8DE51B99E9CE053A2598D0AFDFA"
  },
  "paymentInstrument": {
   "id": "F45FE45E7993C7DBE053A2598D0AED19"
  },
  "card": {
   "type": "001"
  },
  "customer": {
   "id": "F45FB3E443AC3C57E053A2598D0A9CFF"
  }
 },
 "processorInformation": {
  "paymentAccountReferenceNumber": "V0010013019326121538313096266",
  "approvalCode": "888888",
  "responseCode": "100"
 },
 "reconciliationId": "67444961BRIL0BB8",
 "status": "PENDING",
 "submitTimeUtc": "2024-01-18T16:52:48Z"
}
```

Making a Credit with a Non-Default Payment Instrument

This section shows you how to make a credit with a non-default payment instrument.

Endpoint

Test: `POST https://apitest.cybersource.com/pts/v2/credits`
Production: `POST https://api.cybersource.compts/v2/credits`
Production in India: `POST https://api.in.cybersource.compts/v2/credits`

Required Fields for Making a Credit with a Non-Default Payment Instrument

clientReferenceInformation.code	
orderInformation.amountDetails.currency	
orderInformation.amountDetails.totalAmount	
paymentInformation.paymentInstrument.id	**Set to the ID of the payment instrument token that you want to use.**

Related Information

- *API field reference guide for the REST API*

Optional Fields for Making a Credit with a Non-Default Payment Instrument

You can use these optional fields to include additional information when making a credit with a non-default payment instrument.

orderInformation.amountDetails.currency

orderInformation.amountDetails.totalAmount

orderInformation.billTo.address1

orderInformation.billTo.administrativeArea

orderInformation.billTo.country

orderInformation.billTo.email

orderInformation.billTo.firstName

orderInformation.billTo.lastName

orderInformation.billTo.locality

orderInformation.billTo.postalCode

paymentInformation.card.expirationMonth

paymentInformation.card.expirationYear

paymentInformation.card.number

paymentInformation.card.type

Related Information

- *API field reference guide for the REST API*

REST Example: Making a Credit with a Non-Default Payment Instrument

Request

```
{
 "clientReferenceInformation": {
  "code": "12345678"
 },
  "paymentInformation": {
    "paymentInstrument": {
      "id": "0F3BB131F8143A58E063A2598D0AB921"
    }
  },
  "orderInformation": {
    "amountDetails": {
      "currency": "USD",
      "totalAmount": "10.00"
    }
  }
}
```

Response to a Successful Request

```
{
 "_links": {
  "void": {
   "method": "POST",
   "href": "/pts/v2/credits/7055968581386446104953/voids"
  },
  "self": {
   "method": "GET",
   "href": "/pts/v2/credits/7055968581386446104953"
  }
 },
 "clientReferenceInformation": {
  "code": "12345678"
 },
 "creditAmountDetails": {
  "currency": "USD",
  "creditAmount": "10.00"
  },
  "id": "7055968581386446104953",
  "orderInformation": {
    "amountDetails": {
      "currency": "USD"
    }
  },
  "paymentAccountInformation": {
    "card": {
      "type": "001"
    }
  },
  "paymentInformation": {
    "tokenizedCard": {
      "type": "001"
    },
    "instrumentIdentifier": {
      "id": "7010000000016241111",
      "state": "ACTIVE"
    },
    "paymentInstrument": {
      "id": "0F3BB131F8143A58E063A2598D0AB921"
    },
    "card": {
      "type": "001"
    }
  },
  "processorInformation": {
    "approvalCode": "888888",
    "responseCode": "100"
  },
  "reconciliationId": "67445196PRILCQCN",
  "status": "PENDING",
  "submitTimeUtc": "2024-01-18T16:54:18Z"
}
```

Making a Credit with a Payment Instrument

This section shows you how to make a credit with a payment instrument.

Endpoint

Test: `POST https://apitest.cybersource.com/pts/v2/credits`
Production: `POST https://api.cybersource.compts/v2/credits`
Production in India: `POST https://api.in.cybersource.compts/v2/credits`

Required Fields for Making a Credit with a Payment Instrument

clientReferenceInformation.code

orderInformation.amountDetails.currency

orderInformation.amountDetails.totalAmount	
paymentInformation.paymentInstrument.id	**Set to the ID of the payment instrument token you want to use.**

Related Information

- *API field reference guide for the REST API*

REST Example: Making a Credit with a Payment Instrument

Request

```
{
  "clientReferenceInformation": {
    "code": "12345678"
  },
    "paymentInformation": {
      "paymentInstrument": {
        "id": "F4D5E715F7BD9910E053A2598D0A7278"
      }
    },
    "orderInformation": {
      "amountDetails": {
        "currency": "USD",
        "totalAmount": "10.00"
      }
    }
}
```

Response to a Successful Request

```
{
  "_links": {
    "void": {
      "method": "POST",
      "href": "/pts/v2/credits/7055969586686467104953/voids"
    },
    "self": {
      "method": "GET",
      "href": "/pts/v2/credits/7055969586686467104953"
    }
  },
  "clientReferenceInformation": {
    "code": "12345678"
  },
  "creditAmountDetails": {
    "currency": "USD",
    "creditAmount": "10.00"
  },
  "id": "7055969586686467104953",
  "orderInformation": {
    "amountDetails": {
      "currency": "USD"
    }
  },
  "paymentAccountInformation": {
   "card": {
    "type": "001"
   }
  },
  "paymentInformation": {
   "tokenizedCard": {
    "type": "001"
   },
   "instrumentIdentifier": {
    "id": "7010000000016241111",
    "state": "ACTIVE"
   },
   "paymentInstrument": {
    "id": "F4D5E715F7BD9910E053A2598D0A7278"
   },
   "card": {
    "type": "001"
   }
  },
  "processorInformation": {
   "approvalCode": "888888",
   "responseCode": "100"
  },
  "reconciliationId": "67446174JRIKXXHB",
  "status": "PENDING",
  "submitTimeUtc": "2024-01-18T16:55:59Z"
}
```

Making a Credit with an Instrument Identifier

This section shows you how to make a credit with an instrument identifier token.

Endpoint

Test: `POST https://apitest.cybersource.com/pts/v2/credits`
Production: `POST https://api.cybersource.compts/v2/credits`
Production in India: `POST https://api.in.cybersource.compts/v2/credits`

Required Fields for Making a Credit with an Instrument Identifier

clientReferenceInformation.code	
orderInformation.amountDetails.currency	
orderInformation.amountDetails.totalAmount	
paymentInformation.paymentInstrument.id	**Set to the ID of the payment instrument token you want to use.**

Related Information

- *API field reference guide for the REST API*

REST Example: Making a Credit with an Instrument Identifier

Request

```
{
  "clientReferenceInformation": {
    "code": "12345678"
  },
    "paymentInformation": {
      "instrumentIdentifier": {
        "id": "7010000000016241111"
      }
    },
    "orderInformation": {
      "amountDetails": {
        "currency": "USD",
        "totalAmount": "10.00"
      }
    }
}
```

Response to a Successful Request

```
{
  "_links": {
    "void": {
      "method": "POST",
      "href": "/pts/v2/credits/7055970261066212404951/voids"
    },
    "self": {
      "method": "GET",
      "href": "/pts/v2/credits/7055970261066212404951"
    }
  },
  "clientReferenceInformation": {
    "code": "12345678"
  },
  "creditAmountDetails": {
    "currency": "USD",
    "creditAmount": "10.00"
  },
  "id": "7055970261066212404951",
  "orderInformation": {
    "amountDetails": {
      "currency": "USD"
    }
  },
  "paymentAccountInformation": {
    "card": {
      "type": "001"
    }
  },
  "paymentInformation": {
    "tokenizedCard": {
      "type": "001"
    },
    "instrumentIdentifier": {
      "id": "7010000000016241111",
      "state": "ACTIVE"
    },
    "card": {
      "type": "001"
    }
  },
  "processorInformation": {
    "approvalCode": "888888",
    "responseCode": "100"
  },
  "reconciliationId": "67445198PRILCQCQ",
  "status": "PENDING",
  "submitTimeUtc": "2024-01-18T16:57:06Z"
}
```

Making a Credit with a Legacy Token

This section shows you how to make a credit with a legacy token.

Endpoint

Test: `POST https://apitest.cybersource.com/pts/v2/credits`
Production: `POST https://api.cybersource.com/pts/v2/credits`
Production in India: `POST https://api.in.cybersource.com/pts/v2/credits`

Required Fields for Making a Credit with a Legacy Token

clientReferenceInformation.code

orderInformation.amountDetails.currency

orderInformation.amountDetails.totalAmount

paymentInformation.legacyToken.id	**Include the ID of the legacy token that you want to use to authorize a payment.**

Related Information

- *API field reference guide for the REST API*

REST Example: Making a Credit with a Legacy Token

Request

```
{
  "clientReferenceInformation": {
    "code": "12345678"
  },
  "paymentInformation": {
    "legacyToken": {
      "id": "B21E6717A6F03479E05341588E0A303F"
    }
  },
  "orderInformation": {
    "amountDetails": {
      "totalAmount": "22.00",
      "currency": "USD"
    }
  }
}
```

Response to a Successful Request

```
{
  "_links": {
    "void": {
      "method": "POST",
      "href": "/pts/v2/credits/7055970562096509704953/voids"
    },
    "self": {
      "method": "GET",
      "href": "/pts/v2/credits/7055970562096509704953"
    }
  },
  "clientReferenceInformation": {
    "code": "12345678"
  },
  "creditAmountDetails": {
    "currency": "USD",
    "creditAmount": "22.00"
  },
  "id": "7055970562096509704953",
  "orderInformation": {
    "amountDetails": {
      "currency": "USD"
    }
  },
  "paymentAccountInformation": {
    "card": {
      "type": "001"
    }
  },
  "paymentInformation": {
    "tokenizedCard": {
      "type": "001"
    },
    "card": {
      "type": "001"
    }
  },
  "processorInformation": {
    "approvalCode": "888888",
    "responseCode": "100"
  },
  "reconciliationId": "67444779FRILJT84",
  "status": "PENDING",
  "submitTimeUtc": "2024-01-18T16:57:36Z"
}
```