worldpay

worldpay + commercetools

User Guide

Version 1 April 2024

Table of Contents

1 Introduction

The worldpay-RAFT-commercetools-connector module provides integration between commercetools and the Worldpay Native RAFT API, to minimize implementation timescales, and client-side configuration required.

The Native RAFT API allows merchants of all verticals to process payments transactions directly to RAFT - Worldpay's core authorization processing platform.

The API is a set of functions used to invoke the transaction processing capabilities of RAFT and utilizes the modern, widely used REST design and the JSON data-interchange format. Messages are received via HTTP protocol over the internet or through dedicated circuits (extranet). In addition to this, the Worldpay product eProtect is used to encrypt payment details to reduce your PCI footprint by using fully hosted fields for PCI-sensitive data.

This document enables business and technical users to understand what scenarios the connector caters for and the technical flows that support them.

1.1 Overview of Features

This user guide will support you with implementing the following payment flow use cases using the worldpay-RAFT-commercetools-connector:

- Payment Authorization through the creditAuth API using eProtect for the following payment methods:
  - Credit Card
  - Google Pay
  - Apple Pay
  - PINLess Debit
- Payment Cancellations (part and full) through the creditAuth API specifying reversal attributes.
- Payment Completions (part and full) through the creditCompletion API.
- Refunds (part and full) through the creditRefund API.
- Gift Card Transactions through the GiftCard API:
  - Gift Card Inquiry
  - Gift Card Pre Authorization
  - Gift Card Pre Authorization Reversal
  - Gift Card Completion
  - Gift Card Refund

In addition to the above use cases, the module source code repository provides:

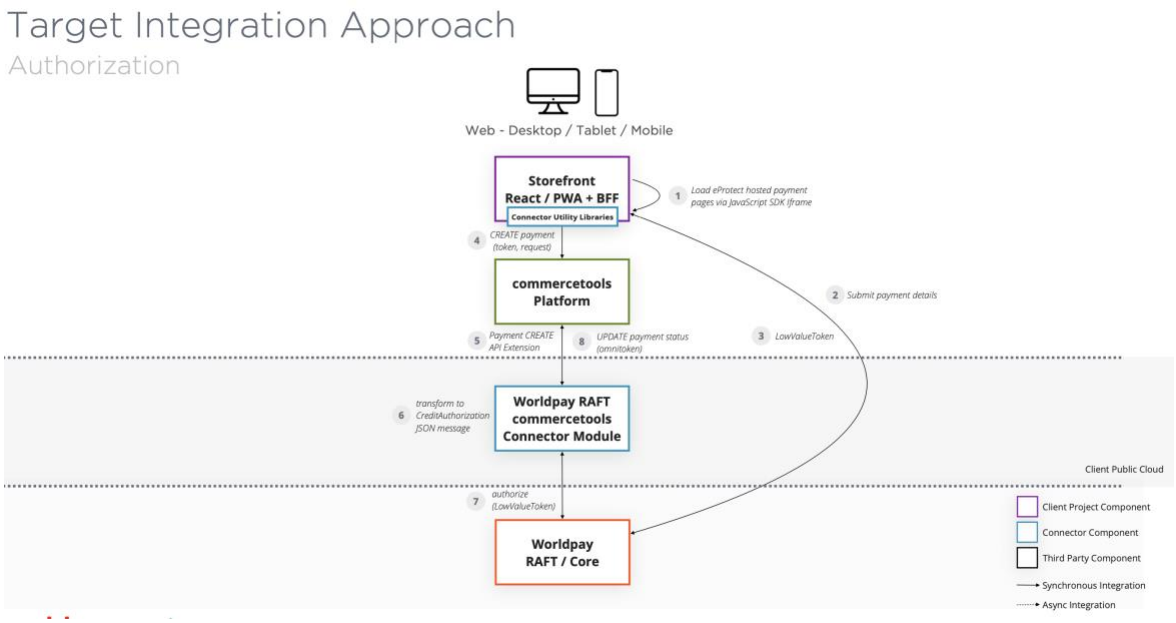
- Infrastructure-as-code (IaC) cloud deployment templates for easy provisioning into a client's cloud infrastructure.
- A suite of unit and integration test components.

1.2 Integration Approach – how it works.

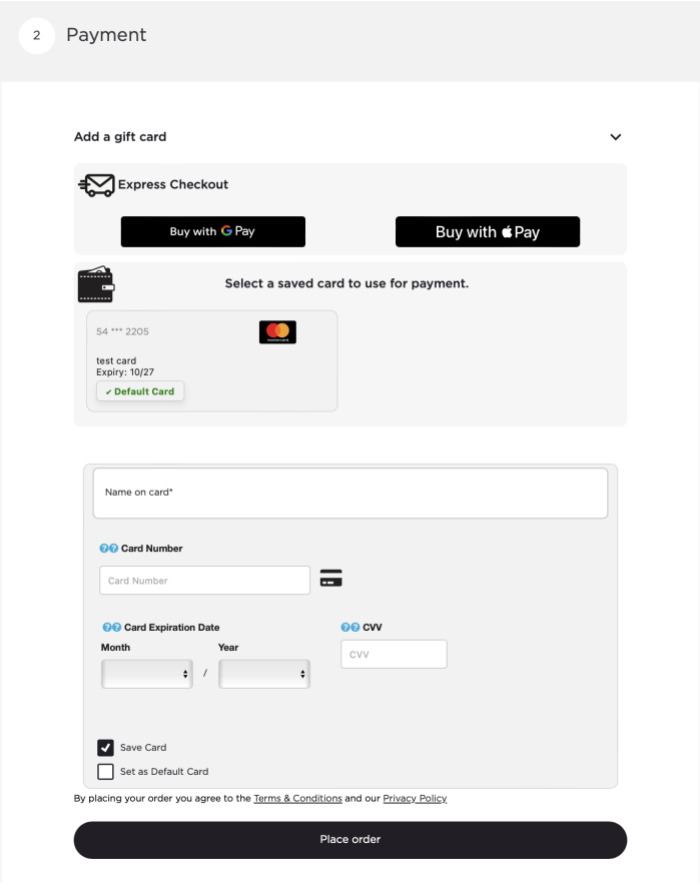
The integration approach leverages commercetools API Extensions to transform payment create / update API calls via the Worldpay RAFT Connector module between commercetools and Worldpay RAFT.

The below diagram shows an overview of the steps within the journey to create / update a payment, this will form the backbone of the infrastructure covered within this document.

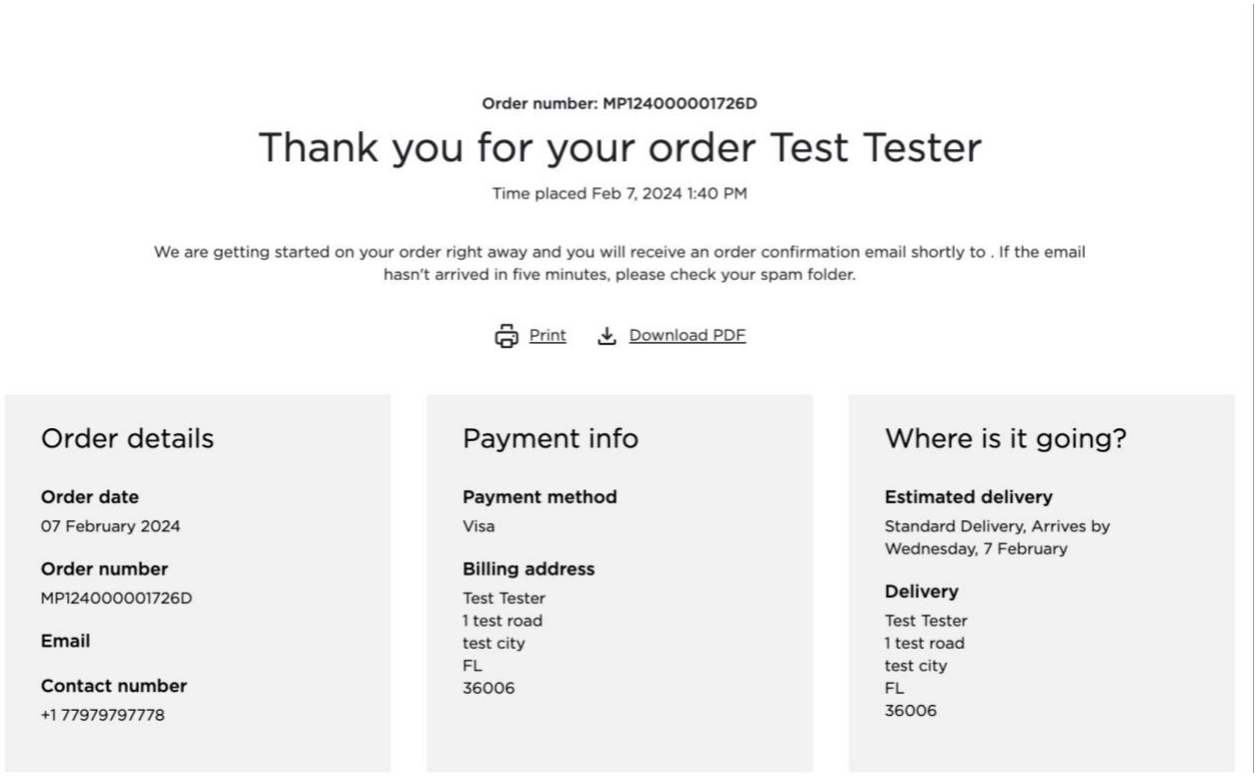
1.2.1 Payment Authorization



- **Step 1**: The customer navigates to the payment page on the checkout, where all applicable payment methods will be displayed, including rendering the eProtect iFrame to allow the customer to pay using a new card.



- **Step 2**: When card details are entered, they are securely sent from eProtect to Worldpay (without passing through your servers).
- **Step 3**: The encrypted fields received by Worldpay are converted into a registrationID (which will be sent as the LowValueToken on the PaymentRequest).
- **Step 4:** The front-end application (or API layer) calls the commercetools HTTP API to create a payment object.
- **Step 5:** A Payment Create commercetools API Extension is invoked to pass on the payment data to the Extension Module.
- **Step 6**: The payment data (including amount, currency, LowValueToken and sequence information) is transformed into the Worldpay creditAuth message format.
- **Step 7**: The creditAuth API call is sent to the Worldpay RAFT to complete the payment authorization.
- **Step 8**: The creditAuth response is received synchronously to update the payment status through the lifecycle of the payment transaction.
- **Step 9**: The payment status is updated in commercetools.
- **Step 10:** The outcome of the payment in commercetools is then passed pack to the Storefront UI which causes the customer to be directed to the relevant response page in the storefront checkout based on the payment result.



Order number: MP124000001726D

Thank you for your order Test Tester

Time placed Feb 7, 2024 1:40 PM

We are getting started on your order right away and you will receive an order confirmation email shortly to . If the email hasn't arrived in five minutes, please check your spam folder.

Print Download PDF

Order details

Order date
07 February 2024

Order number
MP124000001726D

Email

Contact number
+1 77979797778

Payment info

Payment method
Visa

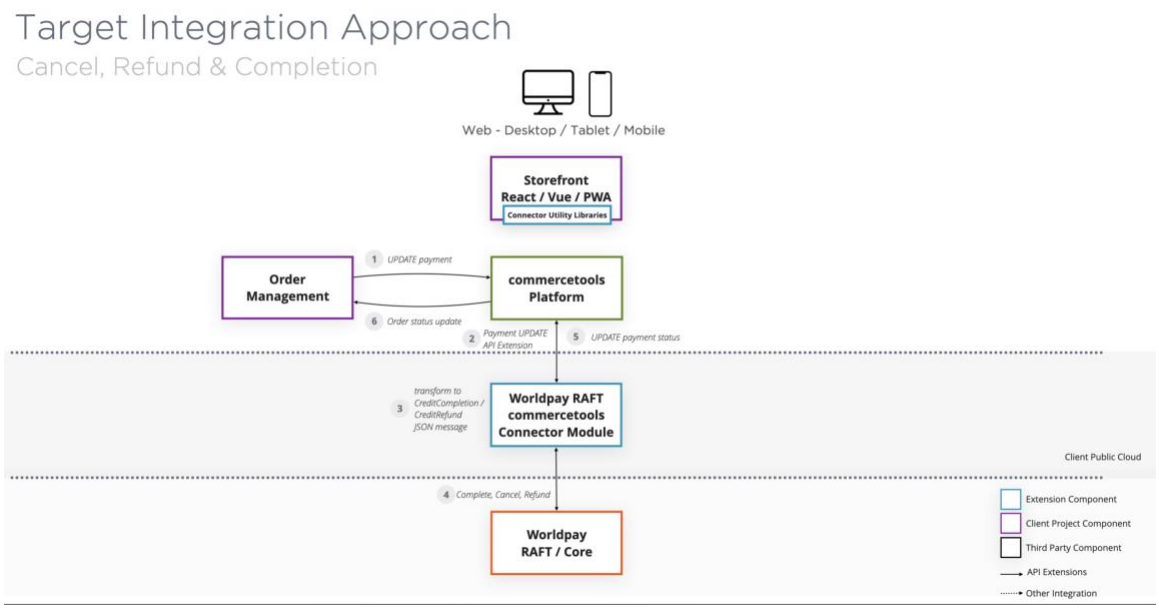
Billing address
Test Tester
1 test road
test city
FL
36006

Where is it going?

Estimated delivery
Standard Delivery, Arrives by Wednesday, 7 February

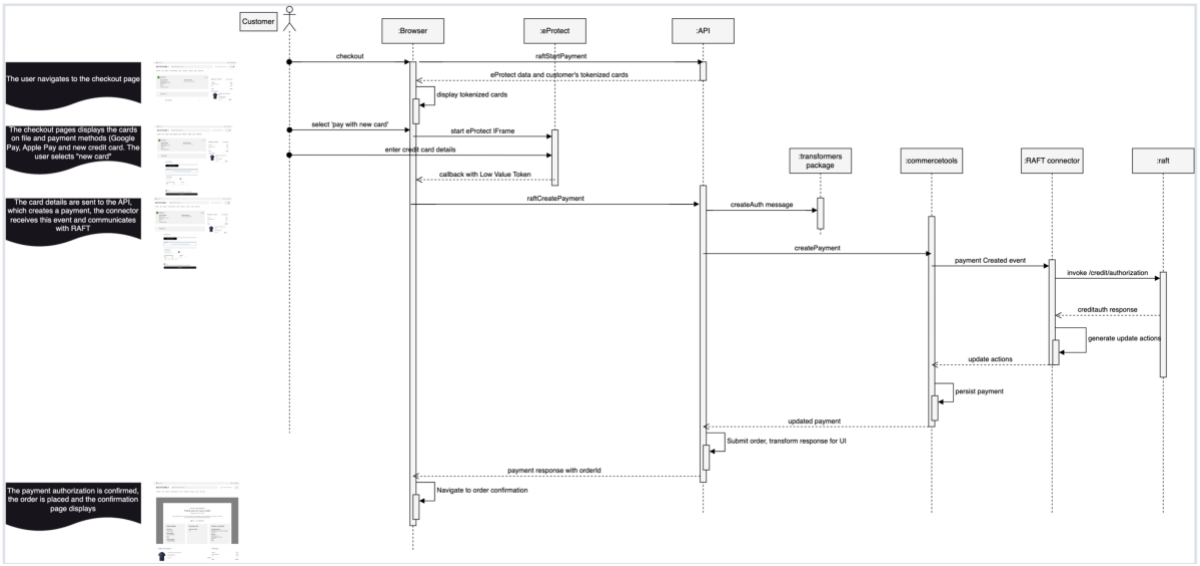
Delivery
Test Tester
1 test road
test city
FL
36006

1.2.2 Payment Modifications

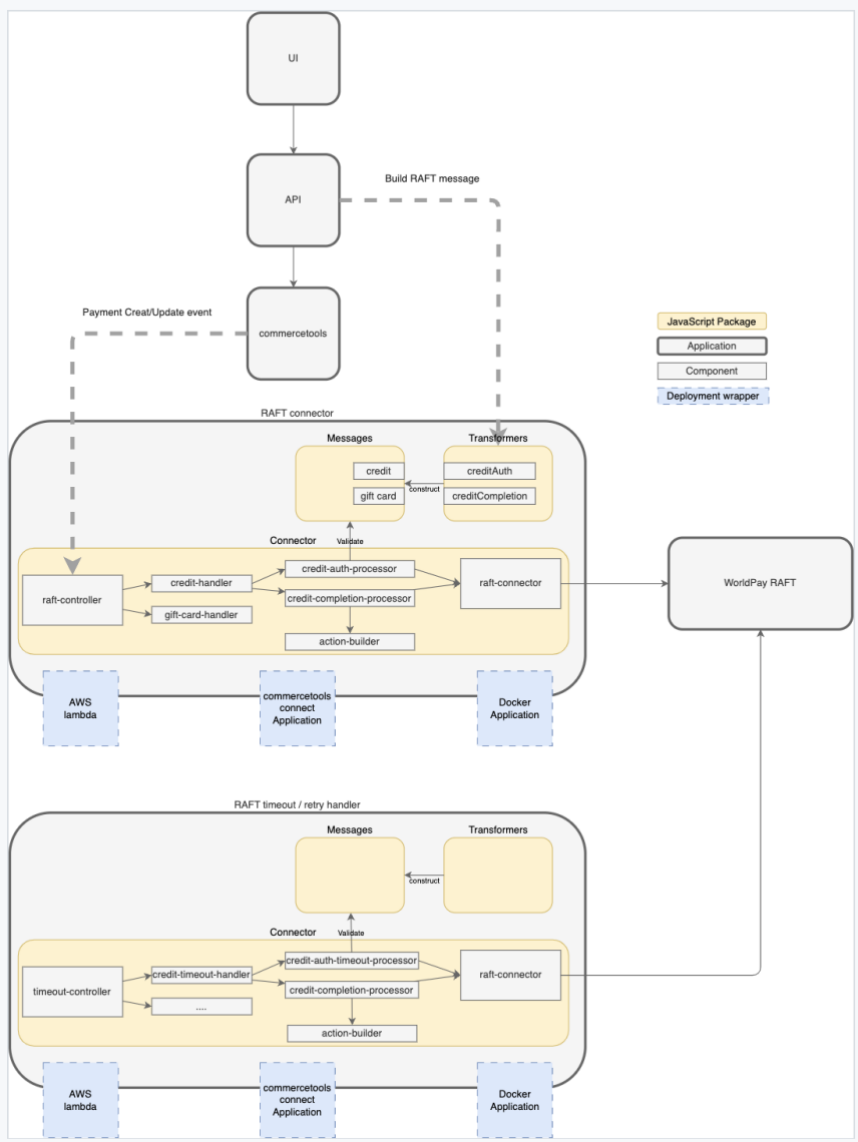


- **Step 1:** When an order update is received from the Order Management System that results in a new transaction against a payment (either an authorization reversal, completion, or refund), a new transaction is created in commercetools.
- **Step 2**: A Payment Update commercetools API Extension is invoked to pass on the payment data to the Extension Module.
- **Step 3**: The payment data (including amount, TokenizedPan and sequence information) is transformed into the Worldpay message format applicable to the required transaction.
- **Step 4**: The relevant modification API call (creditCompletion, creditRefund or creditAuth – Reversal) is sent to the Worldpay RAFT to complete the request.
- **Step 5**: The response is received synchronously to update the payment status through the lifecycle of the payment transaction and the payment status is updated in commercetools.
- **Step 6:** The outcome of the payment in commercetools is then passed pack to the Order Management System to continue the order lifecycle.

1.3 Payment Sequence Flow

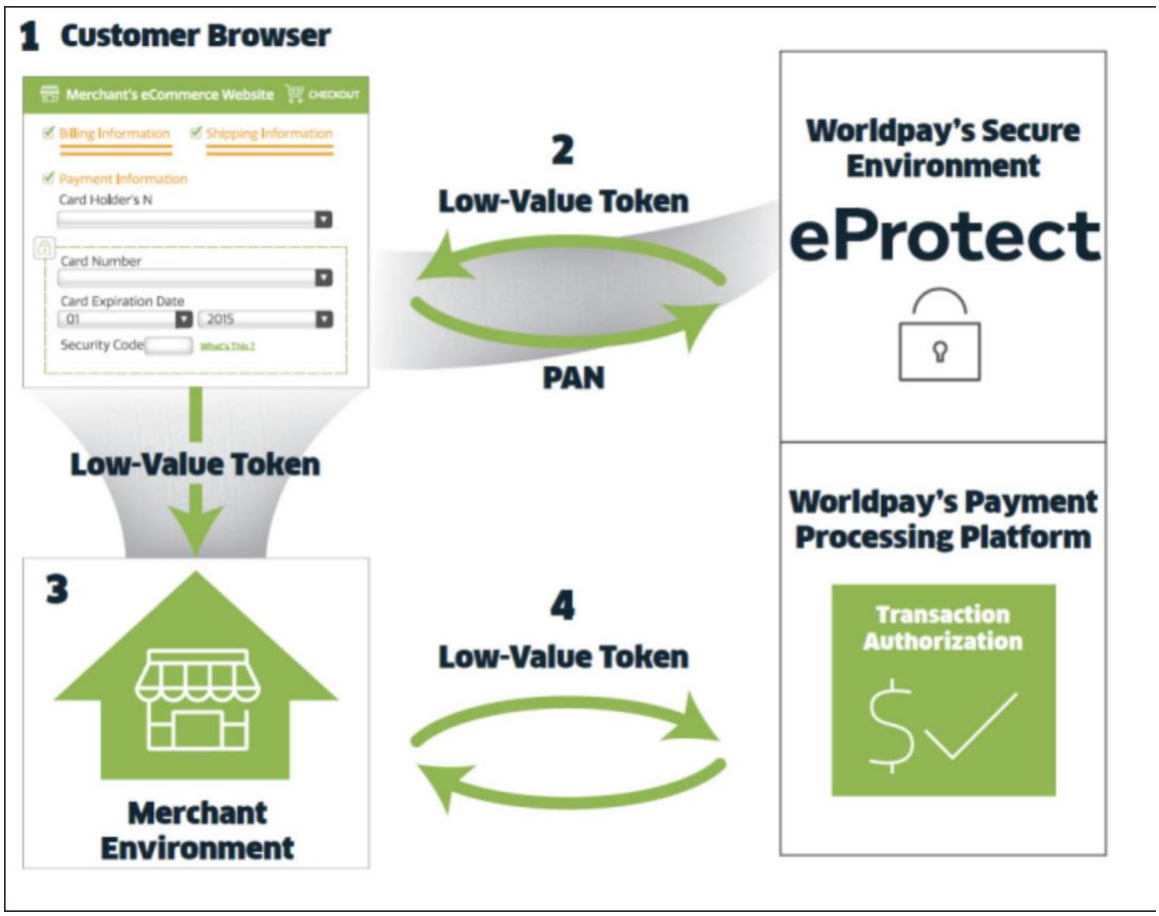


It should be noted that although the sequence flow depicts multiple elements, only those contained within the 'RAFT Connector' and 'RAFT timeout / retry handler' boxes (in the diagram below) are included within the connector, with the assumption that the remaining elements exist within your overall technical solution.



1.4 Worldpay RAFT Integration using eProtect.

Worldpay eProtect is a feature that helps merchants reduce their exposure to sensitive cardholder data and the risk of payment data theft. It works by encrypting the card data at the point of entry and sending it to Worldpay, where it is stored in a secure vault and replaced by a token. The token can then be used for subsequent transactions without revealing the original card data. This is visually depicted in the image below:



By using eProtect with Worldpay RAFT, you can reduce your PCI compliance scope, enhance your security, and offer a seamless payment experience to your customers, all while no card data is transmitted via your web server.

Worldpay provides multiple integration options for eProtect, however the connector has been designed to be used in conjunction with the iFrame API for card and the JavaScript Customer Browser API for Apple pay.

Further developer documentation on eProtect can be found here.

1.4.1 Worldpay eProtect prerequisites

Before you start using the eProtect feature, you must complete the following:

- Obtain a PayPage ID from your eProtect Implementation Consultant
- create a Cascading Style Sheet (CSS) to customize the look and feel of the iFrame to match your checkout page, then submit the Style Sheet to Worldpay for verification. See Creating a Customized CSS for iFrame for more information.

1.5 Extensibility

The connector is designed to be flexible and easy to customize. It consists of Node.js applications that can be provisioned in your chosen public cloud, for example as serverless functions in AWS or Azure, or on any system using Docker containers.

As well as this, due to the composable commerce architecture approach of constructing the message on the client API layer side, there are also additional extensibility hooks:

- The message can be amended with any additional attribute on the API side - which is client code - to extend the information being passed.
- The extension module does not need to retrieve additional information (i.e. cart, customer) which reduces the time it takes for the request to send.
- There is no need to store the message attributes somewhere in the payment, cart, or customer just to pass them over to the connector.

To add a new property in the message, you can just populate the property in your API layer, then the connector will validate the message and pass it on (no data amended in the extension).

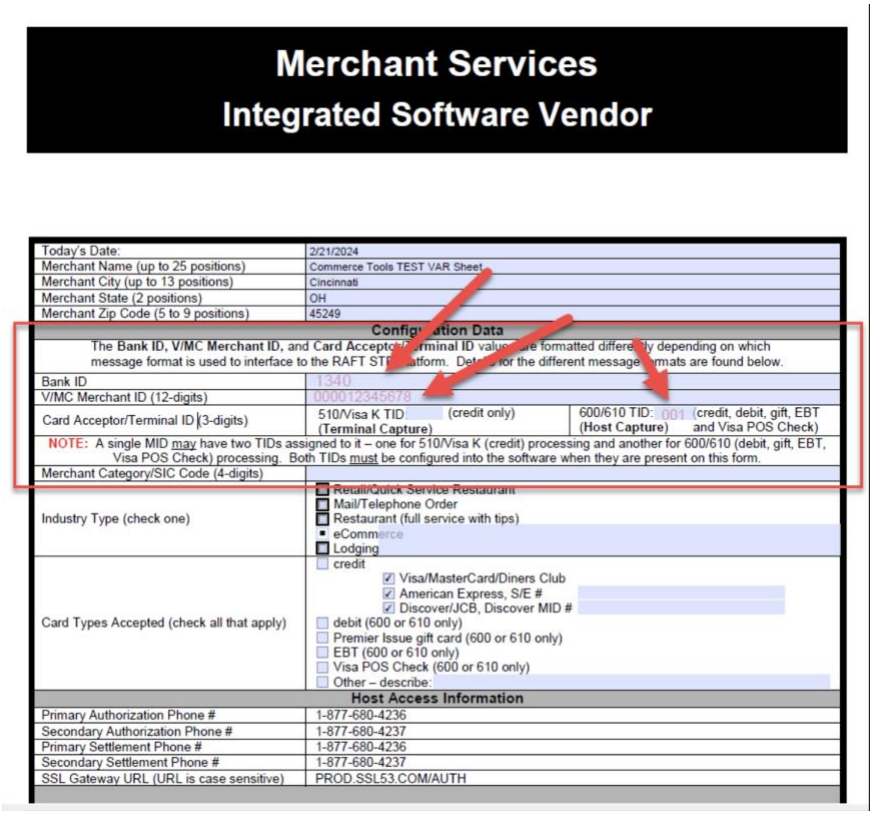
Please Note: adding a new property in the request will lead to additional information in the response that needs storing, the extension will need to be changed to ensure this can be stored.

1.6 Before you start

1.6.1 Worldpay prerequisites

Make sure your Implementation Manager or support contact has provided you with a Worldpay VAR sheet (as below) which should include:

- A Worldpay Merchant ID – configured to accept your contracted payment methods.
- A STP Bank ID
- A STP Terminal ID



Merchant Services
Integrated Software Vendor

Today's Date:	2/21/2024
Merchant Name (up to 25 positions)	Commerce Tools TEST VAR Sheet
Merchant City (up to 13 positions)	Cincinnati
Merchant State (2 positions)	OH
Merchant Zip Code (5 to 9 positions)	45249
Configuration Data	
The Bank ID, V/MC Merchant ID, and Card Acceptor/Terminal ID values are formatted differently depending on which message format is used to interface to the RAFT STIP platform. Details for the different message formats are found below.	
Bank ID	1340
V/MC Merchant ID (12-digits)	000012345678
Card Acceptor/Terminal ID (3-digits)	510/Visa K TID: (credit only) (Terminal Capture) / 600/610 TID: 001 (credit, debit, gift, EBT (Host Capture) and Visa POS Check)
NOTE: A single MID may have two TIDs assigned to it – one for 510/Visa K (credit) processing and another for 600/610 (debit, gift, EBT, Visa POS Check) processing. Both TIDs must be configured into the software when they are present on this form.	
Merchant Category/SIC Code (4-digits)	
Industry Type (check one)	☐ Retail/Quick Service Restaurant ☐ Mail/Telephone Order ☐ Restaurant (full service with tips) ▪ eCommerce ☐ Lodging
Card Types Accepted (check all that apply)	☐ credit ☑ Visa/MasterCard/Diners Club ☑ American Express, S/E # ☑ Discover/JCB, Discover MID # ☐ debit (600 or 610 only) ☐ Premier Issue gift card (600 or 610 only) ☐ EBT (600 or 610 only) ☐ Visa POS Check (600 or 610 only) ☐ Other – describe:
Host Access Information	
Primary Authorization Phone #	1-877-680-4236
Secondary Authorization Phone #	1-877-680-4237
Primary Settlement Phone #	1-877-680-4236
Secondary Settlement Phone #	1-877-680-4237
SSL Gateway URL (URL is case sensitive)	PROD.SSL53.COM/AUTH

In addition to this, you will be provided with a RAFT API license and an eProtect PayPage ID.

All these values will be required on every interaction with RAFT and are covered in more detail in Section 3.1.1.

1.6.1.1 eProtect

The eProtect PayPage ID is a critical component that is NOT a configuration requirement for the connector itself but is integral to utilizing eProtect from the client-side logic. It is your responsibility to ensure the secure storage and management of the PayPage ID where you will be leveraging eProtect.

1.6.2 Commercetools prerequisites

As well as configuring Worldpay, before the connector can be used, commercetools configuration and deployment of the connector into the cloud is required.

When using the connector in a commercetools project, a few prerequisites exist for the project setup:

- Data Model Extensions:
  - Customer attribute `raftTokenizedCards`, to store cards on file to a customer.
  - Payment type to store the request for RAFT and any response attribute to be kept.
  - Transaction type for the `authorizationNumber` to be used in subsequent transactions, and `STPReferenceNUM` for reference.
  - Payment states: states that the payment will have after the connector completes processing: Open, Paid, Failed, Refunded, Cancelled.

- Extensions:
  - The RAFT payment connector needs to be configured to receive payment Create and Update events.

The resources/commercetools folder holds files to add these extensions.

Detail on how to do this can be found in the Developer Guide.

1.7 If you need support

Please get in touch with your Worldpay support contact. If they're not available, call our customer service team on 0800 096 3997.

2 Installation

As well as the Worldpay and commercetools prerequisites, configuration, and deployment of the connector into you chosen public cloud is required. Detail on how to do this can be found in the Deployment Guide.

It is essential to have all required configuration parameters set up for the connector to run correctly.

3 Basic Configurations

3.1 Configuring your Environment

When starting the connector, configuration needs to be passed in to control how the connector behaves. The configuration will be validated at start-up time, to check if all required parameters for the application are in place. Missing configuration will cause the connector to fail.

3.1.1 How do I provide configuration parameters?

Configuration parameters are user-definable settings that control various aspects of a server's behavior, you can use configuration parameters to tailor a server for an installation's particular needs. The simplest way to set up the configuration parameters is by using environment variables.

Important: Make sure that you securely store these environment variables in your cloud infrastructure (i.e., within AWS Secrets Manager or Azure Key Vault) and ensure that the AWS resources have requisite Identity and Access Management (IAM) permissions to read these secrets.

Note: If you try to use the connector without having set up the configuration parameters, then you will receive an error and the server will not start up.

The table below describes the main configuration parameters that are required in each individual message sent to RAFT and can be used for multiple accounts:

Parameter	Type	Description
worldpayRaftLicense	String, required	The RAFT license number that will be used in the request header to authenticate with the RAFT service.
MerchantID	String, Required	The merchantID which identifies the merchant with the RAFT service.
STPBankID	String, Required	The BankID used in each request to the RAFT service, identifying the merchant's bank.
STPTerminalID	String, Required	The TerminalID sent to RAFT in each communication, identifying the merchant.

The table below contains optional parameters that can be sent, along with the default value that will be used if they are not available in the configuration:

Parameter	Type	Description	Default Value
worldpayRaftURL	String, optional	The URL of the server used for the RAFT communication defaulting if unavailable in the configuration.	`https://ws-cert.vantiv.com`
worldpayRaftPath	String, optional	The path on the RAFT server, defaulting if unavailable in the	`/merchant/servicing/apitransactions/NativeRaftApi/v1`

		configuration. The path is concatenated to the URL.	

3.2 Exposing Endpoints

The connector is invoked by payment API Extension requests from commercetools. These are configured against commercetools. By default, the connector expects the commercetools payment API extension requests to be configured to the destination /payments path from the base url of your server, and the order notifications from Worldpay to be configured at the /notifications path.

For example, you start up the local server on http://localhost:3000, the default endpoint for processing incoming commercetools payment API extension requests will be http://localhost:3000/payments.

3.3 Deployment Templates

Information regarding the deployment templates can be found in the Technical Documentation.

3.4 Test Components

To replicate the certification testing conducted, a full suite of test scripts can be found within the Postman Collection.

4 Main Configurations

4.1 Payment Authorization, Order Modification flows

4.1.1 Payment Authorization

The authorization transaction can be used to confirm that a customer has submitted a valid payment method with their order and has sufficient funds to purchase the goods or services they ordered. An approved Credit Card Authorization reduces the customer's credit limit (or bank balance, in the case of a debit card) by the amount of the approval by placing the amount on hold. By default, a credit card authorization is a preauthorization (preauth), as there will always be a second piece to finalize the authorization: a completion request.

The use case for payment authorization is as follows:

As a: **System**

I want: **to create a Payment Request (`creditAuth`) to Worldpay RAFT**

So that: **Worldpay can initiate a payment authorization.**

To initiate the authorization, your API layer initiates the `PaymentRequest` to a transformers package, this is to construct the payment authorization message, which is then stored in the `custom.fields.request` object in commercetools. When the connector receives an update to the Create Payment API Extension, it then retrieves this object and sends it to RAFT to initiate the `creditAuth` transaction.

The full overview of the creditAuth message can be found here, but the fields that must be sent (and are expected by the connector) are:

Field	Description
TransactionAmount	Value submitted for authorization.
AVSZIPCode	Cardholders Zip code (up to 9 characters so can handle zip+4)
AVSAddress	Cardholders address
WorldPayMerchantID	The full Merchant ID assigned by Worldpay (i.e. MID).
APITransactionID	Worldpay uses this for transaction matching (reversals, completions, and so on) and tracking. If you are initiating a subsequent message that ties back to an original request, provide the `APItransactionID` of the original transaction. Cross-format matching is also supported in the ISO and 610 formats, matching to field Native RAFT API Transaction ID.
STPBankId	Contains the 4-digit bank id assigned by Worldpay during boarding.
STPTerminalId	Contains the 3-digit terminal id assigned by Worldpay during boarding.
E-commerceIndicator	All electronic commerce transactions must include this field to indicate the type of transaction being performed. It can also be used to distinguish various types of Bill Payment transactions. Valid values are: • `01` - Single transaction - default for Bill Payments • `02` - Recurring Transaction

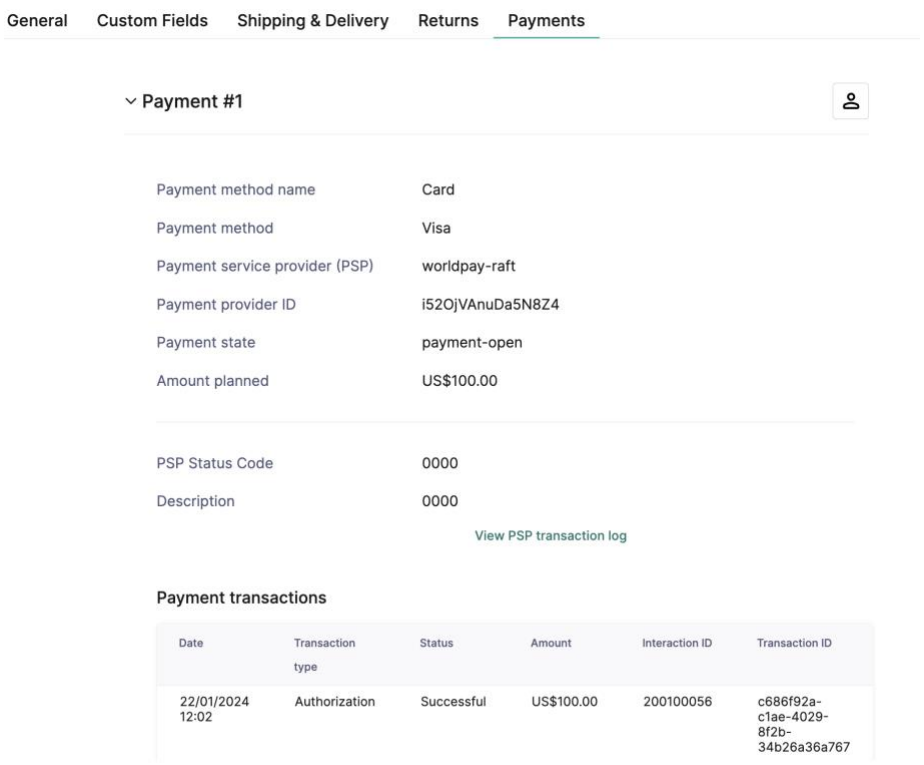
	• 03 - Installment Payment • 05 - Verified by Visa authenticated/MasterCard Secure Code with AAV data/Discover with CAVV data. • 06 - Verified by Visa attempts processing/MasterCard Secure Code with or without AAV data/Discover with or without CAVV data. • 07 - eCommerce, but neither Verified by Visa, nor MasterCard SecureCode. • 08 - The cardholder's payment card data was transmitted to the merchant using no security method. • 09 - Used by non-U.S. merchants to designate Secure Electronic Transaction (SET) purchases. U.S. Issuers should not receive ECI of 9, unless the value was the result of a processing error or a miscoded value. • 10 - Recurring transaction (first transaction of a recurring payment series) • 20 - Token Initiated (AMEX only)
AcquirerCurrencyCode	Acquirer Currency Code in ISO 4217 format
LocalDateTime	This field contains the merchant's local date and time in a YYYY-MM-DDTHH:mm:ss format. For example, 4:15 PM on October 31st of 2023 would be 2023-10-31T16:15:00. Any follow up type messages for gift cards (completions, reversals, etc.) should contain the original local date/time for matching purposes.
LowValueToken	The field contains a temporary token / registrationID with an expiration time used with eProtect. In a card not present online environment a low value token is presented in lieu of PAN and optional CVV information.
ExpirationDate	In YYMM format, Worldpay recommends that merchants provide expiry date with every transaction using a Token.
PinlessRequest	Y/N flag indicating whether the customer would like Worldpay to attempt a PINLess conversion from signature to debit. Set to Y
Cvv2Cvc2CIDIndicator	This field contains the CVV2, CVC2, or CID Presence Indicator. Set to 1 if LowValueCVV2Token is passed

PartialAllowed	Y/N flag indicating whether the customer allows partial authorizations.
ReturnTokenizedPan	Y/N flag indicating the merchant would like the requested card results to be returned as a tokenized PAN.
CardholderInitiatedTransaction	Y/N flag indicating that the cardholder is the entity that initiated the transaction.
SequenceNumber_00-99	Indicates which shipment is being sent. Use 00 for initial auth.
SequenceCount_01-99	Indicates the total number of shipments (set to 99 for all shipments except the final shipment, which should send the number of actual completions sent)
FinalShipment	Y/N flag indicating this transaction is the final shipment of an order. Use N for initial auth.
UserData1	<=35 Characters - Client specific data (such as Order number etc.) This is an optional field.
UserData2	<=20 Characters - Client specific data (such as Customer number etc.) This is an optional field.
UserData3	<=20 Characters - Client specific data (such as Customer email etc.) This is an optional field.

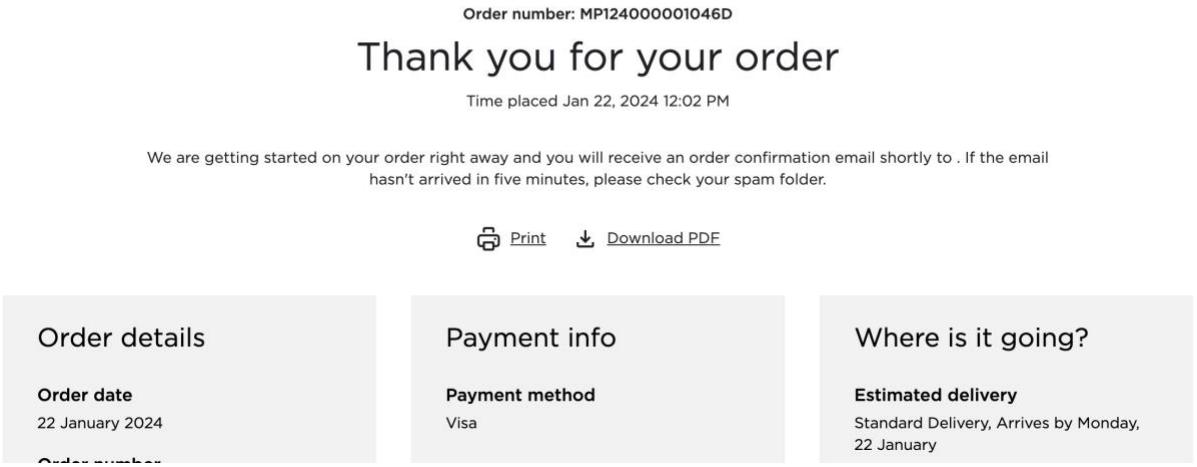
As previously mentioned in the extensibility section, the connector can be built out to include other fields available within the specification, however these need to be extended to be added in the custom object to be expected for both the request and the response.

When the payment response is received, it will contain a `ReturnCode` of `0000` that confirms the authorization was successful. Any non-zero value or fault code means the transaction has failed.

This is then updated in commercetools, updating the payment object with the response from RAFT.



Based on your project settings, commercetools should then push this payment update to your API, which should in turn direct the customer to the relevant outcome in the browser.

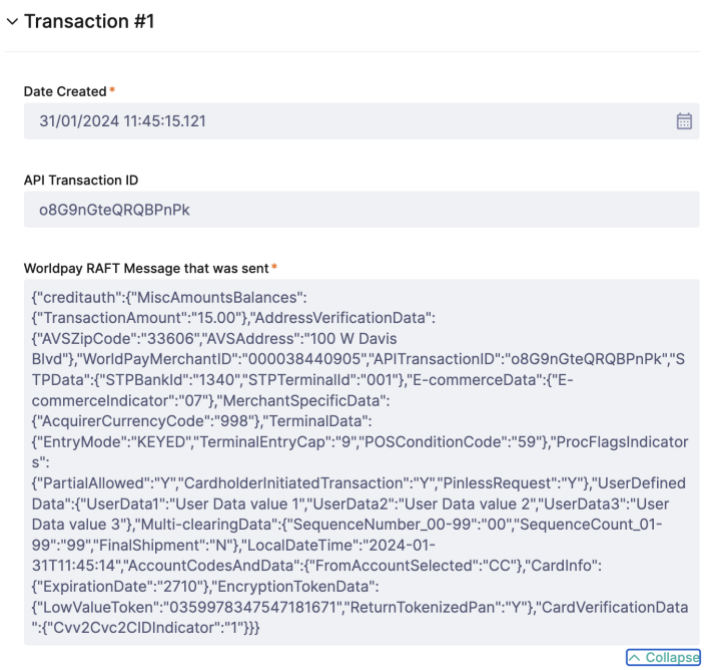


The following information should show in the response:

Field	Description
ReturnCode	This parameter indicates the success of the message. Any non-zero value means the transaction failed. Expected values are: ○ 0000 – Successful ○ 0004 – Edit error on input ○ 0008 – Logic error ○ 0012 – System issue
ReasonCode	This parameter contains a value you can use to pinpoint exactly where the error occurred.
CardProductCode	This field indicates the network brand's product for the selected BIN.
AVSResult	Worldpay returns this field in the response message if the authorization request included `AVSZipCode` or `AVSAddress`.
TokenizedPAN	This field contains the tokenized PAN that is returned to convert the Low Value Token (submitted in the request) to a High Value Token (to be used on subsequent transactions – such as `creditCompletion`.
PINLessConverted	for PINLess Debit transactions only – Y/N flag indicating whether Worldpay was able to successfully perform PINLess conversion.
CVV2fromReg-ID	Y/N flag indicating if a CVV2 was generated from a low value token.
STPReferenceNUM	Worldpay assigned reference number on each transaction that is returned to the terminal. Sent by the terminal on follow up messages to match back to the original transaction.
RetrievalREFNumber	Contains a value generated by the message originator to associate a unique identifier to a given transactions to identify it throughout the transaction's life cycle (authorization, reversal, etc.) This value will be generated by Worldpay if not sent in the request.
APITransactionID	Worldpay uses this for transaction matching, if you are initiating a subsequent message that ties back to an original request, provide the `APITransactionID` of the original transaction.

Other data will be included with the response but will be card network / brand specific.

This request and response data will be stored within the commercetools Payment Object to use on subsequent transactions.



4.1.2 Order Modifications

Following a successful authorization, subsequent requests will be required to:

- Complete on the authorized funds.
- Cancel the authorized funds.
- Refund completed funds.

Order modifications with RAFT are fully synchronous giving immediate responses to requests without the need for asynchronous notifications.

Please note, that there is no error handling built into the API extension, the assumption would be that any error's would be treated as failures and if necessary, a re-attempt should be made as a separate transaction.

4.1.2.1 Sending a Completion on an Authorized Payment

To ensure that this can be completed, multiple conditions must be met:

1. A payment has already been authorized for the order (as a result a payment object with `TransactionType=Authorization` with `TransactionState=Success` has been created in commercetools, the completion request currency, `APITransactionID` and the

`STPReferenceNUM` matches match what was originally submitted and received on the authorization request.

2. A payment transaction has been created for all or some of the original authorization value with the `TransactionType=Charge` and `TransactionState=Initial` in commercetools).

The use case for payment completion is:

As a: **System**

I want: **To send an order modification completion (`creditCompletion`) request**

So that: **I can complete funds that have been authorized and update the order.**

For the use case, the trigger to formulate the modification request will be outside the scope of the connector and need to be configured (initiating the `chargePayment` request from the API), but once the conditions have been met, the initiation of the modification will occur.

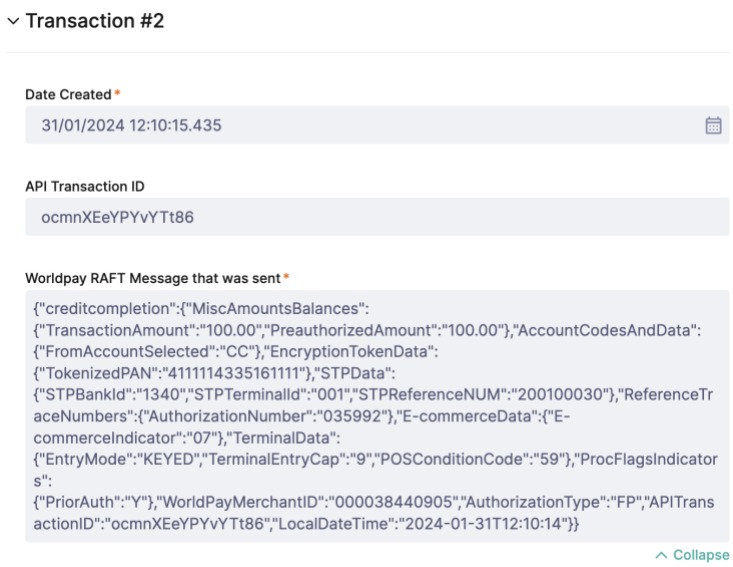
The `creditCompletion` has set mandatory fields that will be collated from the order in commercetools which include:

Field	Description
WorldPayMerchantID	The full Merchant ID assigned by Worldpay (i.e. MID).
TransactionAmount	This is the amount being requested on the completion in the payment currency.
PreauthorizedAmount	The amount that was originally pre-authorised.
APITransactionID	Worldpay uses this for transaction matching (reversals, completions, and so on) and tracking. If you are initiating a subsequent message that ties back to an original request, provide the `APItransactionID` of the original transaction.
LocalDateTime	This field contains the merchant's local date and time in a YYYY-MM-DDTHH:mm:ss format.
STPBankId	Contains the 4-digit bank id assigned by Worldpay during boarding.
STPTerminalId	Contains the 3-digit terminal id assigned by Worldpay during boarding.

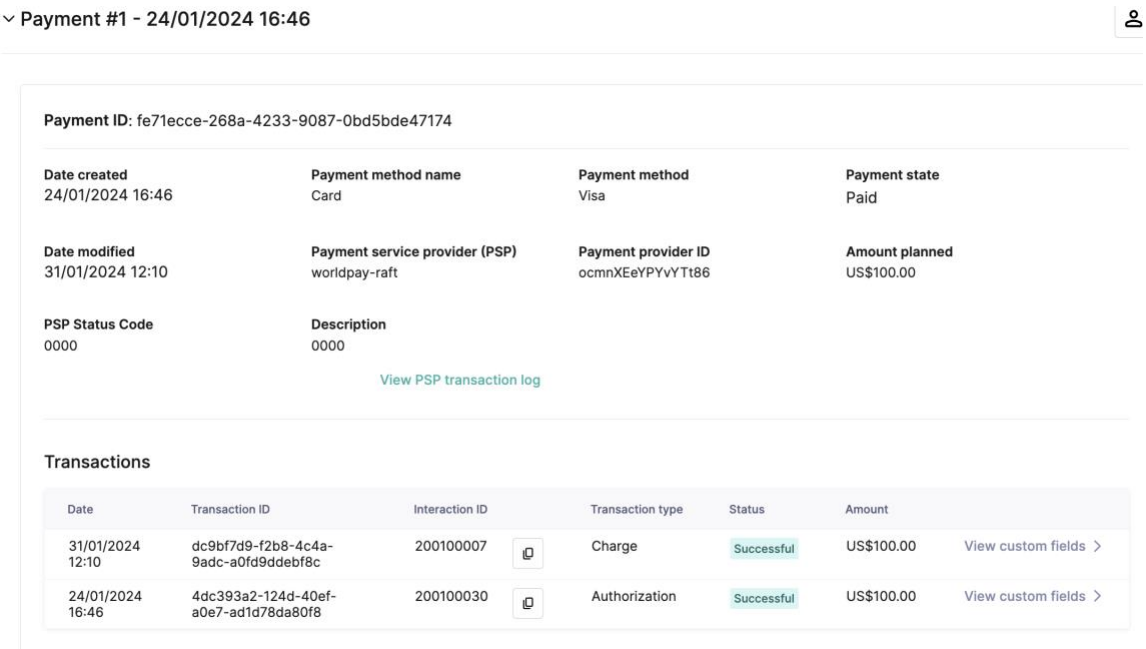
STPReferenceNUM	Worldpay assigned reference number on each transaction that is returned to the terminal in the CreditAuth. Must be sent on follow-up messages to match back to the original transaction.
TokenizedPAN	This field contains the tokenized PAN.
PriorAuth	Y/N flag indicating if the transaction is a prior authorized transaction.
RetrievalREFNum	Contains a value generated by the message originator to associate a unique identifier to a given transactions to identify it throughout the transaction's life cycle (authorization, reversal etc). This value will be generated by Worldpay if not sent in the request.

Other fields can be included, a full list of values that can be submitted can be found here.

Once constructed, the request is then sent via the connector to worldpay to process and will be logged in the `worldpay-raft-payment-interaction` object within commercetools to track the payment interaction.



Worldpay will then send a synchronous response confirming whether the request was successfully received, which based on the response would then update the `TransactionState` from `Initial` to either `Success` or `Failure`. As well updating the transaction, the response will be logged in the `worldpay-raft-payment-interaction` object within commercetools.



Please Note: Following a successful completion, you will receive a settlement for the transaction which finalises the transfer of funds from the bank to your account. It is important to note that the settlement of the funds is outside of the connector and if this fails, then will need to be addressed within your own code.

4.1.2.2 Cancelling an Authorized Payment

To ensure that this can be process correctly, multiple conditions must be met:

1. A payment has already been authorized for the order (as a result a payment object with `TransactionType=Authorization` with `TransactionState=Success` has been created in commercetools) and the completion request currency and the `RetrievalREFNum` and `APITransactionID` matches what was originally submitted and received on the authorization request.
2. A payment transaction has been created for all or some of the original authorization value with the `TransactionType=AuthorizationCanceled` and `TransactionState=Initial` in commercetools).

The use case for part or full payment cancellation is:

As a: **System**

I want: **to create a Payment Reversal Request (`creditAuth`) to Worldpay RAFT**

So that: **Worldpay can void any funds being held on an authorization.**

For the use case to be met, the trigger to formulate the modification request will be outside the scope of the connector and need to be configured (initiating the `cancelPayment` request from the API), but once the conditions have been met, the initiation of the modification will occur.

Please note, the same request format will be used regardless of whether a full or partial reversal is being submitted, with the `TransactionAmount` specifying the amount of the reversal.

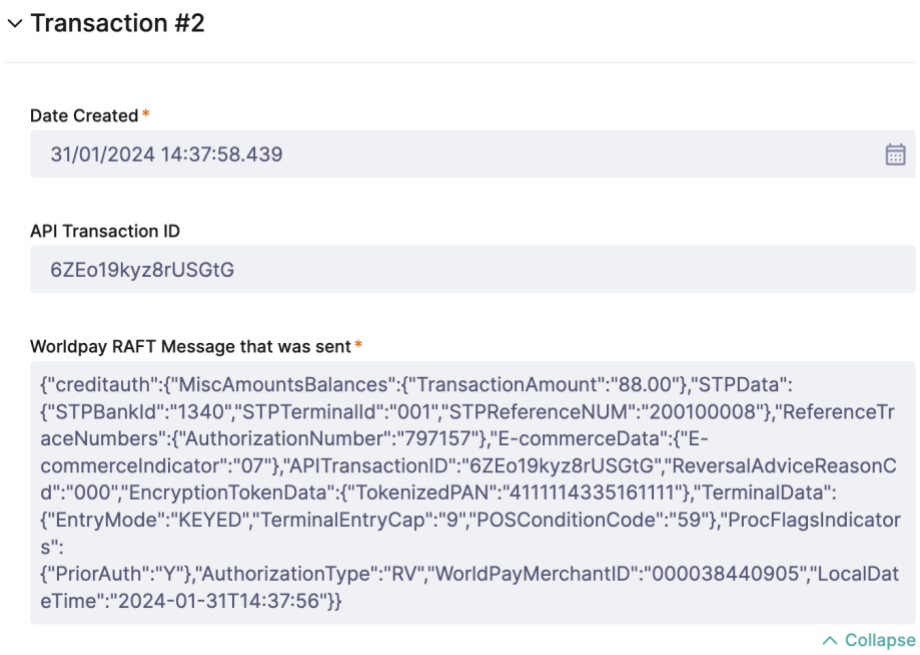
The `creditAuth` has set mandatory fields that will be collated from the order in commercetools for a reversal which include:

Field	Description
TransactionAmount	Value submitted for reversal.
AcquirerCurrencyCode	Acquirer Currency Code
STPBankId	Contains the 4-digit bank id assigned by Worldpay during boarding.
STPTerminalId	Contains the 3-digit terminal id assigned by Worldpay during boarding.
STPReferenceNUM	Worldpay assigned reference number on each transaction that is returned to the terminal in the CreditAuth. Must be sent on follow-up messages to match back to the original transaction.
WorldPayMerchantID	The full Merchant ID assigned by Worldpay (i.e. MID).
ReversalAdviceReasonCd	For reversal messages, this field contains the reason the reversal was generated. For advice messages, it contains the reason or nature of the advice. If a value is not provided, Worldpay will use default values. Valid values are: • 000 - Normal Reversal • 002 - Timeout Reversal • 003 – Syntax • 005 - Clerk Cancel • 006 - Customer Cancel • 010 - Previously Authorized
AuthorizationType	The value RV should be provided here. This field provides a means for the transaction disposition to be changed from standard authorization to forced conditions.
APITransactionID	Worldpay uses this for transaction matching (reversals, completions, and so on) and tracking. If you are initiating a subsequent message that ties back to an original request,

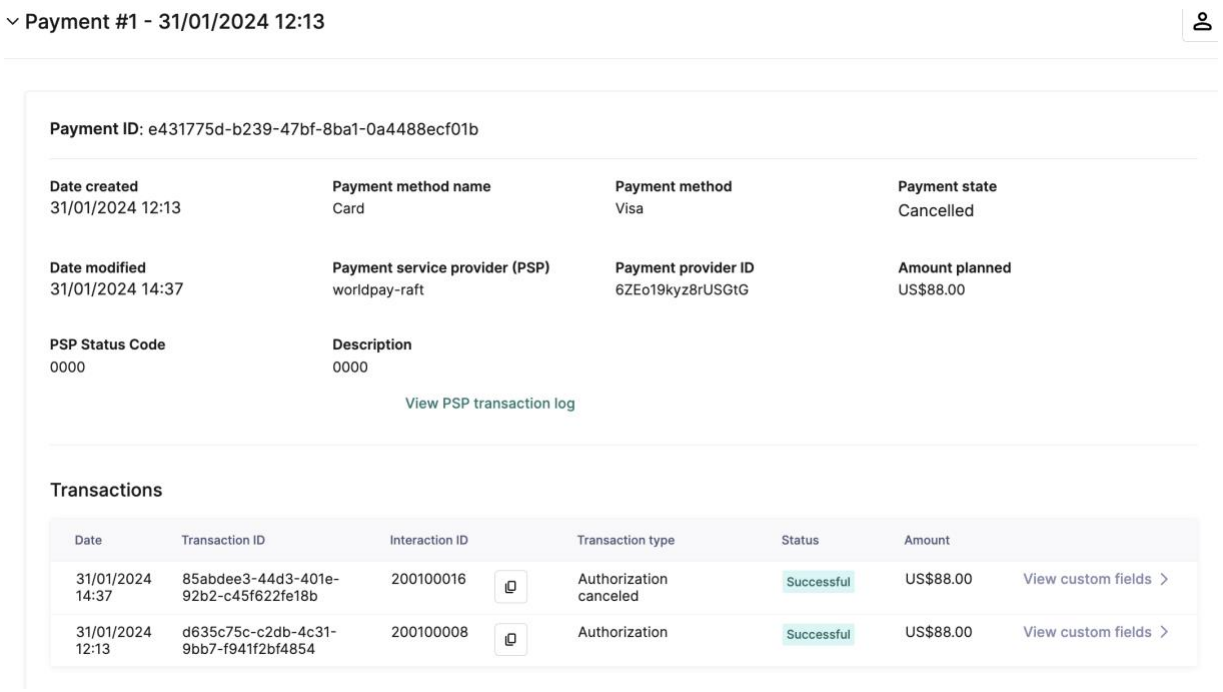
	provide the `APItransactionID` of the original transaction. Cross-format matching is also supported in the ISO and 610 formats, matching to field Native RAFT API Transaction ID.
TokenizedPAN	This field contains the tokenized PAN.
PriorAuth	Y/N flag indicating if the transaction is a prior authorized transaction.
LocalDateTime	This field contains the merchant's local date and time in a YYYY-MM-DDTHH:mm:ss format. For example, 4:15 PM on October 31st of 2023 would be 2023-10-31T16:15:00. Any follow up type messages for gift cards (completions, reversals, etc.) should contain the original local date/time for matching purposes.
RetrievalREFNum	Contains a value generated by the message originator to associate a unique identifier to a given transactions to identify it throughout the transaction's life cycle (authorization, reversal etc.) This value will be generated by Worldpay if not sent in the request.

Other optional fields can be found here.

Once constructed, the request is then sent via the connector to worldpay to process and will be logged in the `worldpay-raft-payment-interaction` object within commercetools to track the payment interaction.



Worldpay will then send a synchronous response confirming whether the request was successfully received, which based on the response would then update the `TransactionState` from `Initial` to either `Success` or `Failure`. As well updating the transaction, the response will be logged in the `worldpay-raft-payment-interaction` object within commercetools.



4.1.2.3 Refund Request

The use case for a refund request is:

As a: **System**

I want: **To send an order modification Refund request**

So that: **I can refund funds that have been completed and update the order**

The difference between the modifications for a cancellation and a refund request is where the cancellation request requires funds to be available on a successful authorization to process, on a refund, the modification occurs against a transaction that has already been completed (charged) successfully.

When a refund request is submitted, a transaction is created with the `TransactionType` set to `Refund` and the `TransactionState` set to `Initial`.

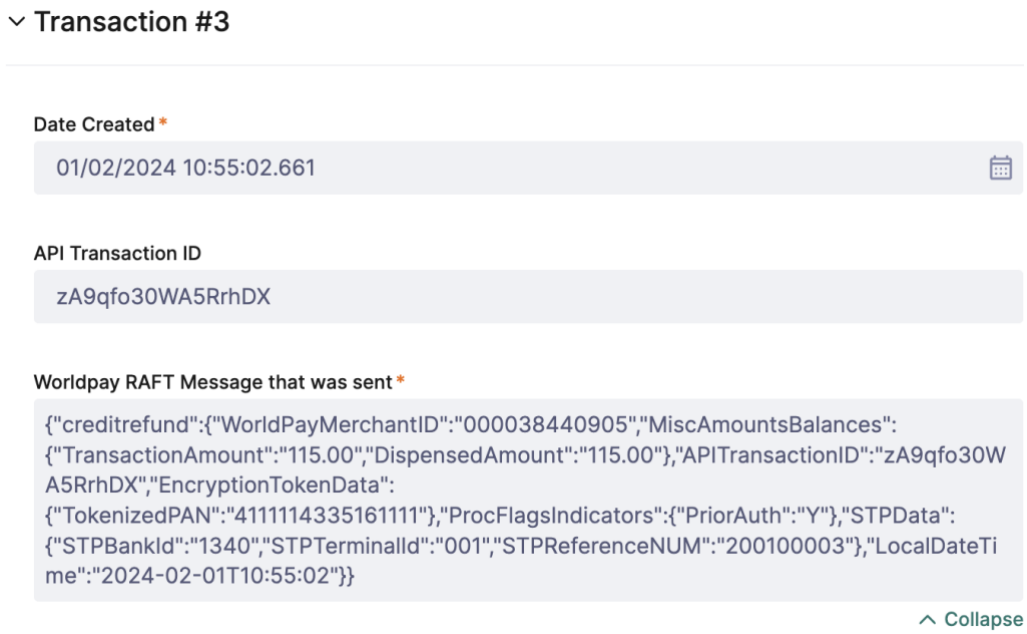
As with the cancellation request, the same request format will be used regardless of whether a full or partial refund is being submitted, with the refund value being specified in the `TransactionValue` field.

The fields that need to be included on the `creditRefund` request include:

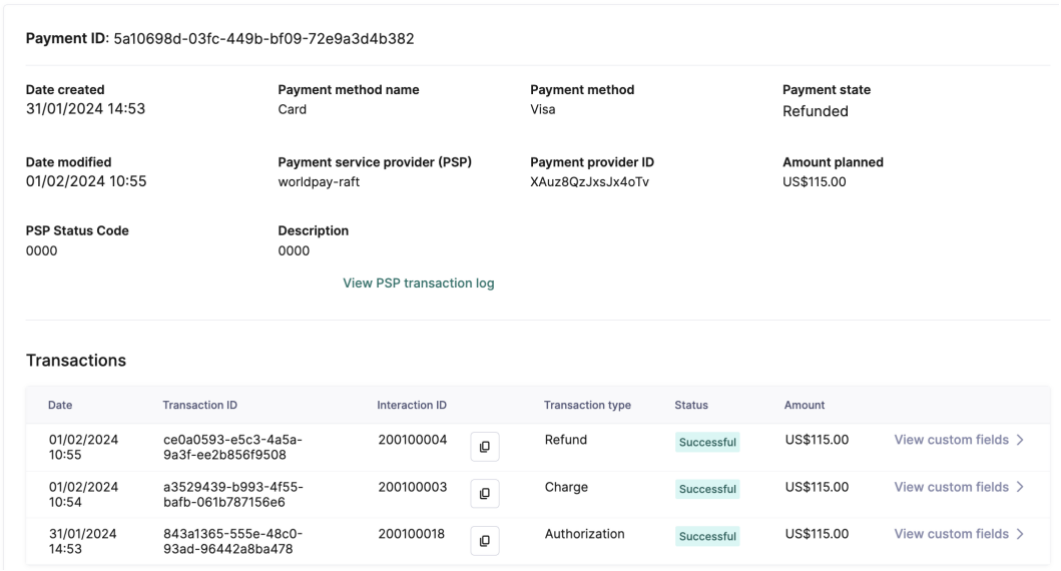
Field	Description
WorldPayMerchantID	The full Merchant ID assigned by Worldpay (i.e. MID).
TransactionAmount	This is the amount being refunded in the payment currency.
PreauthorizedAmount	The amount that was originally pre-authorized.
APITransactionID	Worldpay uses this for transaction matching (reversals, completions, and so on) and tracking. If you are initiating a subsequent message that ties back to an original request, provide the `APItransactionID` of the original transaction.
TokenizedPAN	This field contains the tokenized PAN.
PriorAuth	Y/N flag indicating if the transaction is a prior authorized transaction.
STPBankId	Contains the 4-digit bank id assigned by Worldpay during boarding.
STPTerminalId	Contains the 3-digit terminal id assigned by Worldpay during boarding.
STPReferenceNUM	Worldpay assigned reference number on each transaction that is returned to the terminal in the CreditCompletion. Must be sent on follow-up messages to match back to the original transaction.
LocalDateTime	This field contains the merchant's local date and time in a YYYY-MM-DDTHH:mm:ss format.

Other optional fields can be found here.

As with the other modifications, once constructed, the request is then sent via the connector to worldpay to process and will be logged in the `worldpay-raft-payment-interaction` object within commercetools to track the payment interaction.



Worldpay will then send a synchronous response confirming whether the request was successfully received, which based on the response would then update the `TransactionState` from `Initial` to either `Success` or `Failure`. As well updating the transaction, the response will be logged in the `worldpay-raft-payment-interaction` object within commercetools.



4.2 Wallet Payments

4.2.1 Apple Pay

Apple Pay offers a secure and seamless payment solution that can be utilized within applications, in physical stores, and on the web. Employing network tokenization, Apple Pay securely stores payment details on the user's eligible Apple Pay device, with transaction confirmation facilitated through Touch ID or Face ID authentication. Notably, if implemented in Europe, Apple Pay adheres to PSD2 SCA (Payment Services Directive 2 Strong Customer Authentication) compliance standards.

When opting for Apple Pay, the user encounters a payment sheet, enabling card selection and the input of contact details and shipping information. Subsequently, the user is prompted to authenticate the transaction using either Face ID or Touch ID.

Apple Pay provides liability shift support for American Express, Discover, JCB, Mastercard, and Visa cards. For Visa, liability shift applies to transactions involving cards issued in the Europe region.

4.2.1.1 Apple Pay prerequisites.

To utilize this payment method, there are a few configurations expected, including the creation of an Apple iOS account (to provide you with an Apple Merchant ID), as well as enabling Apple Pay on your Worldpay account. From this, Worldpay can provide a Certificate Signing Request (CSR), which needs to be used as the Payment Processing Certificate to facilitate a secure connection with your Apple Pay Merchant ID.

Full instructions on how to complete these configurations can be found here.

Please note that the integration to Apple Pay is outside of the scope of the connector, and only starts to interact with the connector at the point of the `createPayment` event.

Apple Pay can work alongside eProtect, the Worldpay eProtect Customer Browser JavaScript API controls the fields on your checkout page that hold sensitive card data. When the cardholder clicks the Apple Pay button, communication is exchanged with Apple Pay via the JavaScript API to obtain the `PKPaymentToken`. From this point forward, your handling of the transaction is identical to any other eProtect transaction. The eProtect server returns a Registration ID (low-value token) and your server constructs the authorization transaction using that ID.

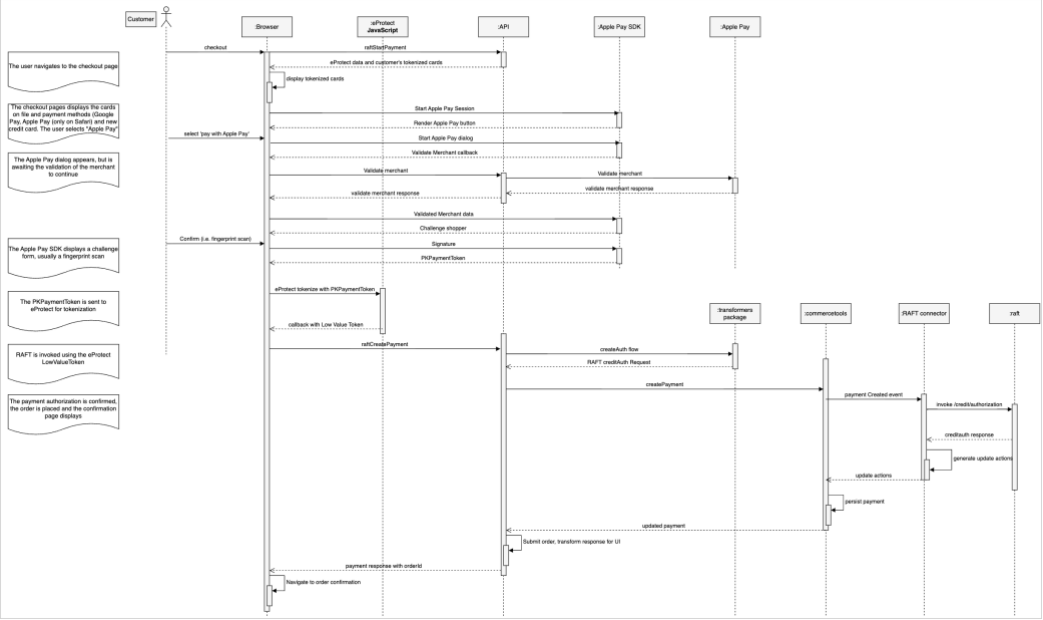
4.2.1.1.1 Project Code

The successful implementation and integration of Apple Pay with eProtect require the expertise of a software developer. As the software developer, your primary role involves the client-side coding necessary to initiate the payment process. This includes crafting the eProtect request, which must

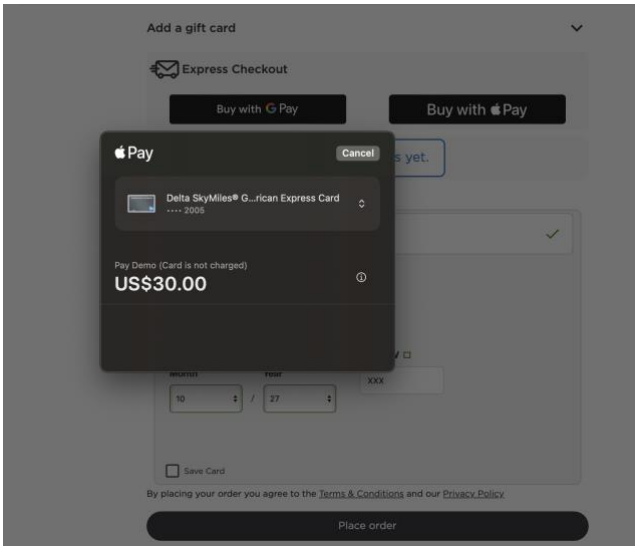
contain the `paypageId` provided by Worldpay. You will be responsible for ensuring that the Apple Pay button on the checkout page triggers the appropriate sequence of events: capturing the `PKPaymentToken` through the Apple Pay API and passing it securely to eProtect using the Worldpay JavaScript API function '`eProtect().sendToEprotect()`'. It is also your duty to manage the Payment Processing Certificate's integration and handle the communication between the client's system and the eProtect server to obtain a Registration ID (`paypageRegistrationId`). Following this, the transaction process through the connector should be seamlessly executed, mirroring any other eProtect transaction.

Documentation on using eProtect JavaScript API can be found here.

4.2.1.2 Apple Pay Shopper Experience



- **Step 1:** The customer navigates to the checkout page.
- **Step 2:** The checkout pages display the stored cards along with other payment methods available.
- **Step 3:** The option to pay via Apple Pay is selected.
- **Step 4:** Apple Pay payment sheet is displayed for the customer to select their preferred payment option.



- **Step 5:** Tokenized Apple Pay details are sent to the API, which creates a payment, the connector receives this event and communicates with RAFT.
- **Step 6:** The payment authorization is confirmed, the order is placed, and the confirmation page is displayed.

4.2.2 Google Pay

Utilizing Google Pay, customers have the convenience of making purchases with the credit or debit cards stored in their Google account (only Visa, MasterCard and American Express cards are currently supported through Worldpay). Whether shopping from Android devices, on the web, or in-store through POS terminals, users benefit from a streamlined checkout process.

During transactions on apps and websites, Google Pay presents a payment sheet, allowing users to choose their preferred card payment method and verify their purchase.

Google Pay extends liability shift support to Mastercard and Visa cards issued in the EU or the UK. However, it does not offer liability shift for American Express or Visa cards issued outside of the EU or the UK. To secure liability shift when using Google Pay, it is necessary to authenticate the transaction with 3D Secure (which is outside the connector).

4.2.2.1 Google Pay prerequisites.

To offer Google Pay, additional set up is required, which includes:

- Setting up a Google Account.
- Worldpay enabling Google Pay on your account.

Full instructions on how to complete these configurations and set up Google Pay can be found here, once opening the Google Tutorial, select the 'Vantiv' Gateway to ensure it will work alongside the RAFT connector.

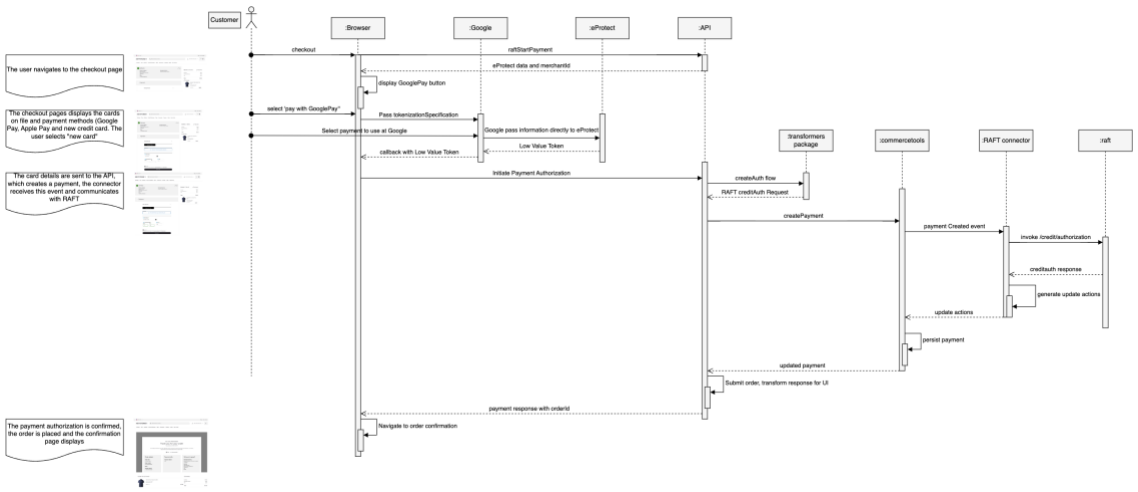
As with Apple Pay, the implementation of Google Pay is outside of the connector and will need to be configured to display as a payment option.

Once the customer has selected Google Pay as their payment method, your system integrator will send Worldpay specific parameters to Google to initialize the Google Pay button. This should then load the payment sheet for the customer to select a payment method in their wallet. Your system integrator will handle the response via the callback function passed in the initialization request. The response will contain the token required to initiate the creditAuth request.

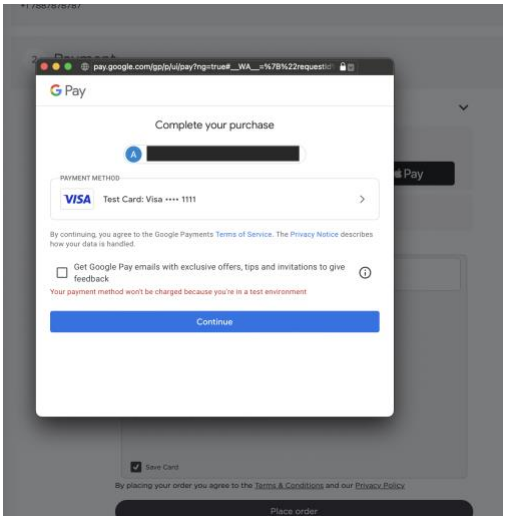
The expected parameters to be sent in the `tokenizationSpecification` to Google Pay:

Field	Description
vantiv:merchantPayPageId	A unique number assigned by eProtect Implementation
vantiv:merchantOrderId	The merchant-assigned unique value representing the order in your system (used when linking authorizations, captures, and refunds, and for retries). The field must be a text string with a maximum of 25 characters.
vantiv:merchantTransactionId	The merchant-assigned unique value representing this transaction in your system. Worldpay recommends that the values for `TransactionId` and `orderId` must be different and unique so that we can use these identifiers for reconciliation or troubleshooting.
vantiv:merchantReportGroup	Report Groups are intended to allow you to segregate transactions by logically grouping them into the different segments of your business. Use any value from 1-25 characters.

4.2.2.2 Google Pay Shopper Experience



- **Step 1:** The customer navigates to the checkout page.
- **Step 2:** The checkout pages display the stored cards along with other payment methods available.
- **Step 3:** The option to pay via Google Pay is selected.
- **Step 4:** Google Pay browser-based pop up is displayed for the customer to select their preferred payment option.



- **Step 5:** Tokenized Google Pay details are sent to the API, which creates a payment, the connector receives this event and communicates with RAFT.
- **Step 6:** The payment authorization is confirmed, the order is placed, and the confirmation page is displayed.

4.3 Gift Cards

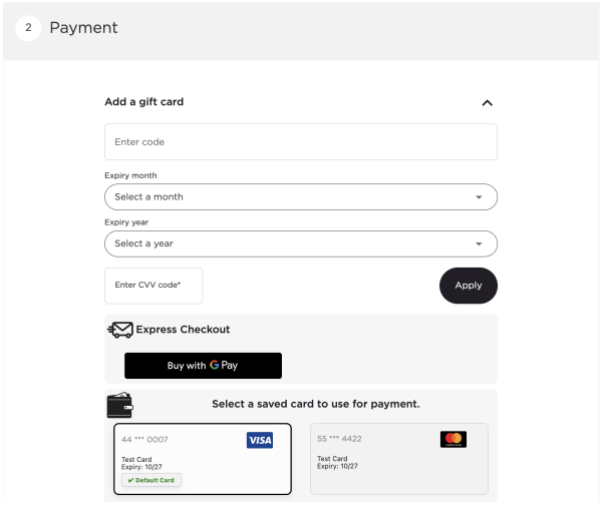
Gift card programs are more than just marketing and a new revenue stream, they're a way to start a virtuous cycle of positive feelings around your brand among both existing and hopefully future customers.

Gift card programs allow you to increase sales, whilst promoting your brand and capitalizing on holiday shopping, where a gift from your business becomes a small part of holiday giving, you're not only making a sale, but you're also making a connection between your company, the gift giver, and the recipient.

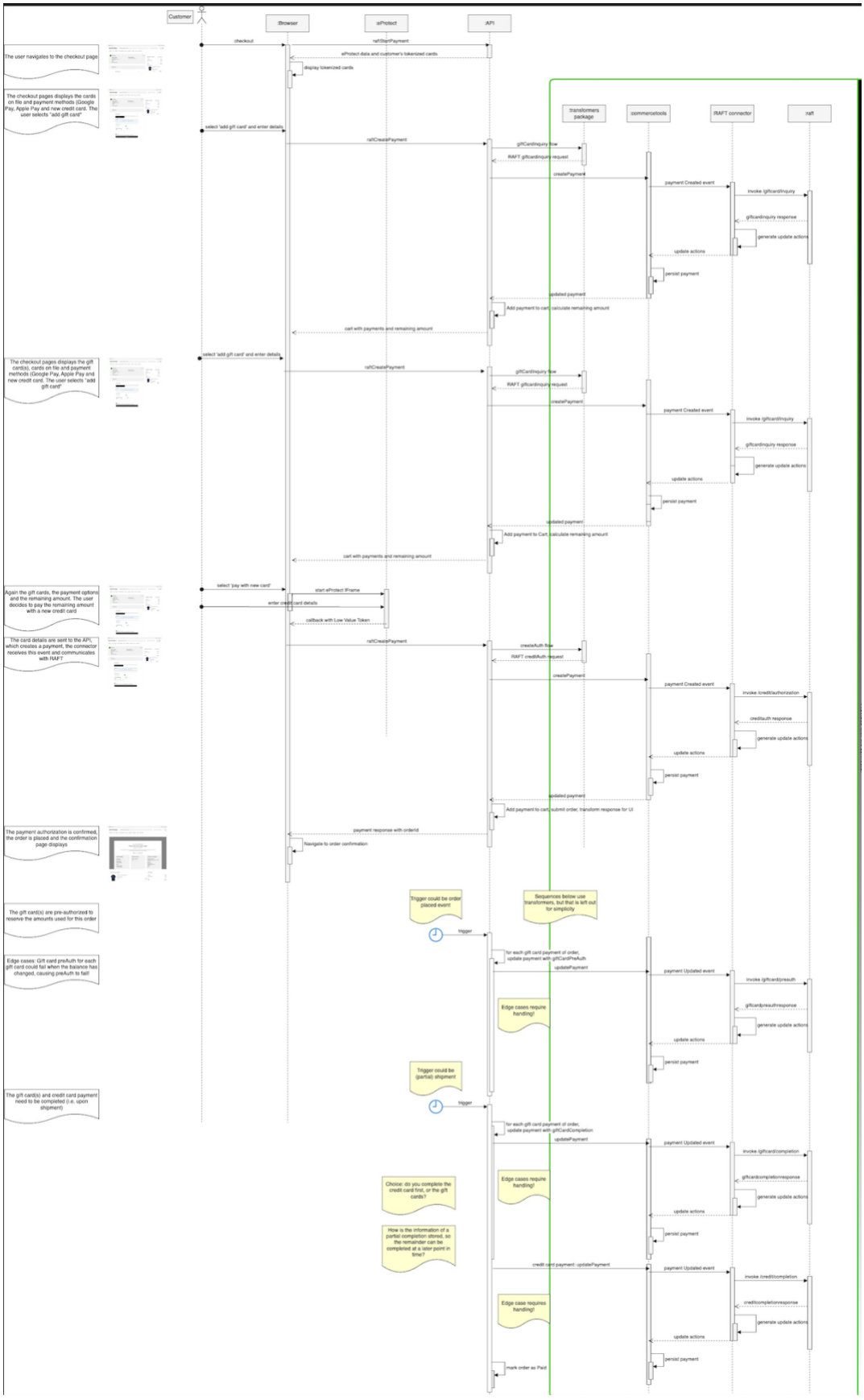
When using a gift card in commercetools in conjunction with the RAFT connector, we expect that a customer can pay with a gift card that (fully or partially) covers the order price. When a gift card does not cover the entire price, the remainder of the purchase must be paid with another payment method. Please note that it is possible that the 2nd (or subsequent) payment method is also a gift card.

4.3.1 Gift Card Shopper Experience

- **Step 1:** The customer navigates to the checkout page.
- **Step 2:** The checkout pages display the stored cards along with other payment methods available, including the option to add a gift card as a payment.



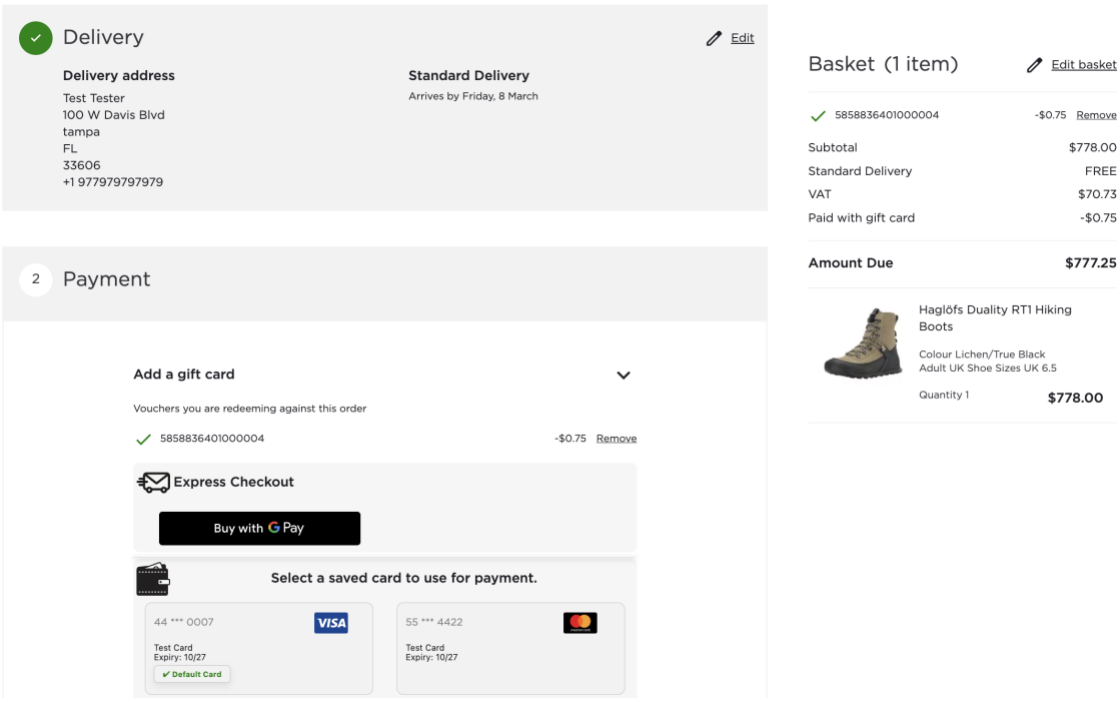
- **Step 3:** The gift card details are entered, and the customer selects 'Apply'. This causes a `giftCardInquiry` to be initiated.
- **Step 4:** The gift card applies to the basket and if any remaining balance is due additional payment information can be input, causing the order to confirm.
- **Step 5:** Once the trigger is met, your API will initiate the sequent gift card transactions (preauth, completion, preauth cancellation, refund).
- **Step 6:** A transaction is created in commercetools with the relevant transaction and the request is sent to RAFT to complete.
- **Step 7:** A response is received, and the transaction is updated in commercetools.


Customer
:Browser
:eProtect
:API
transformers package
commercetools
:RAFT connector
:raft
checkout
raftStartPayment
eProtect data and customer's tokenized cards
display tokenized cards
The user navigates to the checkout page
The checkout pages displays the cards on file and payment methods (Google Pay, Apple Pay and new credit card. The user selects "add gift card"
select 'add gift card' and enter details
raftCreatePayment
giftCardInquiry flow
RAFT giftcardinquiry request
createPayment
payment Created event
invoke /giftcard/inquiry
giftcardinquiry response
generate update actions
update actions
persist payment
updated payment
Add payment to cart, calculate remaining amount
cart with payments and remaining amount
The checkout pages displays the gift card(s), cards on file and payment methods (Google Pay, Apple Pay and new credit card. The user selects "add gift card"
select 'add gift card' and enter details
raftCreatePayment
giftCardInquiry flow
RAFT giftcardinquiry request
createPayment
payment Created event
invoke /giftcard/inquiry
giftcardinquiry response
generate update actions
update actions
persist payment
updated payment
Add payment to Cart, calculate remaining amount
cart with payments and remaining amount
Again the gift cards, the payment options and the remaining amount. The user decides to pay the remaining amount with a new credit card
select 'pay with new card'
start eProtect IFrame
enter credit card details
callback with Low Value Token
The card details are sent to the API, which creates a payment, the connector receives this event and communicates with RAFT
raftCreatePayment
createAuth flow
RAFT creditAuth request
createPayment
payment Created event
invoke /credit/authorization
creditauth response
generate update actions
update actions
persist payment
updated payment
Add payment to cart, submit order, transform response for UI
The payment authorization is confirmed, the order is placed and the confirmation page displays
payment response with orderId
Navigate to order confirmation
Trigger could be order placed event
Sequences below use transformers, but that is left out for simplicity
trigger
for each gift card payment of order, update payment with giftCardPreAuth
updatePayment
payment Updated event
invoke /giftcard/preauth
giftcardpreauthresponse
generate update actions
update actions
persist payment
The gift card(s) are pre-authorized to reserve the amounts used for this order
Edge cases: Gift card preAuth for each gift card could fail when the balance has changed, causing preAuth to fail!
Edge cases require handling!
Trigger could be (partial) shipment
trigger
for each gift card payment of order, update payment with giftCardCompletion
updatePayment
payment Updated event
invoke /giftcard/completion
giftcardcompletionresponse
generate update actions
update actions
persist payment
The gift card(s) and credit card payment need to be completed (i.e. upon shipment)
Edge cases require handling!
Choice: do you complete the credit card first, or the gift cards?
How is the information of a partial completion stored, so the remainder can be completed at a later point in time?
credit card payment: updatePayment
payment Updated event
invoke /creditcompletion
creditcompletionresponse
generate update actions
update actions
persist payment
Edge case requires handling!
mark order as Paid

4.3.2 Gift Card Inquiry

A `giftcardinquiry` allows you to ascertain the value the remaining balance on a gift card. Once this is established, it allows you to reduce the payable balance against the order until the full value is met. Each payment applied must be sent to RAFT as individual requests to be processed successfully.

Please note - as the inquiry doesn't reduce the available balance on the gift card, you should take steps to prevent the same gift card from being applied more than once.



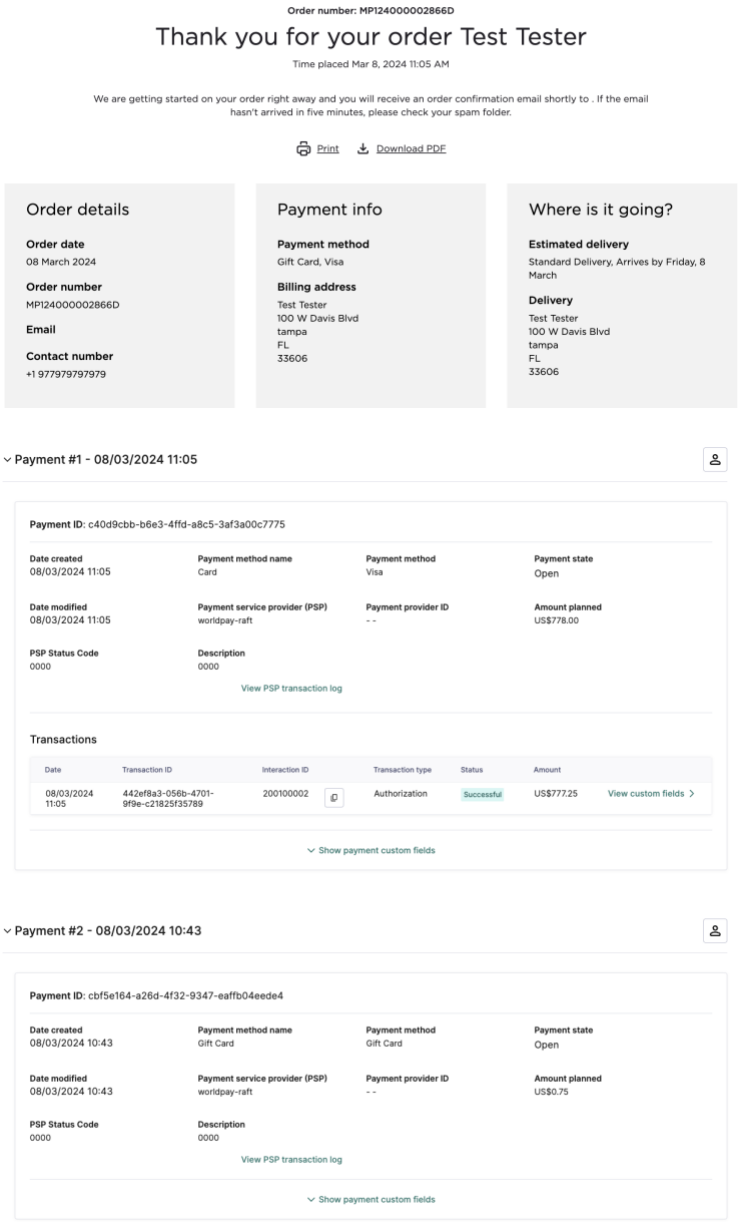
The full overview of the gift card inquiry message can be found here, but the fields that need to be included on the `giftcardinquiry` request include:

Field	Description
WorldPayMerchantID	The full Merchant ID assigned by Worldpay (i.e. MID).
APITransactionID	Worldpay uses this for transaction matching (reversals, completions, and so on) and tracking. If you are initiating a subsequent message that ties back to an original request, provide the `APItransactionID` of the original transaction.
PAN	This is the gift card number entered by the customer.

ToAccountSelected	Account selected at the terminal by the customer. This should show the value GC, it will be defaulted if not provided based on other information in the message.
GcSecurityCode	Similar to the CVV2 value, Worldpay denies the transaction if you use an invalid code. Additionally, three failed Security Code attempts place a lock on the card. You can only remove it with a successful transaction where the security code is not present and is not required. If the code is present, whether it is required or not, Worldpay validates it.
ExpirationDate	In YYMM format, Worldpay recommends that merchants provide expiry date with every transaction using a Token.
STPBankId	Contains the 4-digit bank id assigned by Worldpay during boarding.
STPTerminalId	Contains the 3-digit terminal id assigned by Worldpay during boarding.
STPReferenceNUM	Worldpay assigned reference number on each transaction that is returned to the terminal in the CreditCompletion. Must be sent on follow-up messages to match back to the original transaction.
LocalDateTime	This field contains the merchant's local date and time in a YYYY-MM-DDTHH:mm:ss format.
TransactionAmount	This field contains the amount of funds requested or remitted by the cardholder in the local currency of the transaction as represented by the acquirer currency code if provided. For the Inquiry, this should contain the value `$0.00`
EntryMode	Indicates the entry mode for the transaction.
POSConditionCode	A value that describes the overall environment in which the transaction takes place.
TerminalEntryCap	Indicates by which the card data was read.

When the inquiry response is received, it will contain a `ReturnCode` of `0000` that confirms the authorization was successful. Any non-zero value or fault code means the transaction has failed.

This is then updated in commercetools, updating the payment object with the response from RAFT.



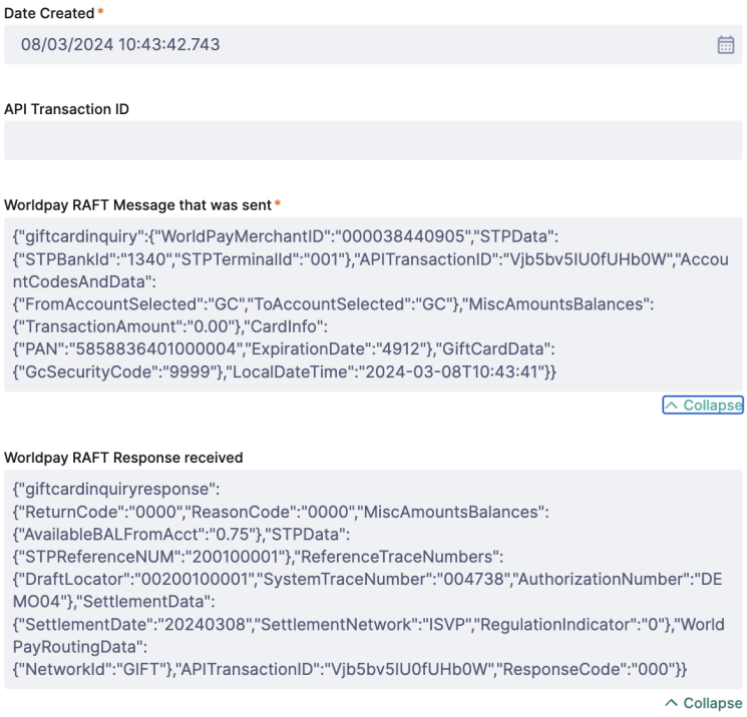
The following information should show in the response:

Field	Description
`ReturnCode`	This parameter indicates the success of the message. Any non-zero value means the transaction failed. Expected values are: ○ 0000 – Successful ○ 0004 – Edit error on input ○ 0008 – Logic error

	○ 0012 – System issue
ReasonCode	This parameter contains a value you can use to pinpoint exactly where the error occurred.
AvailableBALFromAcct	Available balance: From Account.
NetworkId	The ID for the network to which Worldpay sent the transaction.
STPReferenceNUM	Worldpay assigned reference number on each transaction that is returned to the terminal. Sent by the terminal on follow up messages to match back to the original transaction.
APITransactionID	Worldpay uses this for transaction matching, if you are initiating a subsequent message that ties back to an original request, provide the `APITransactionID` of the original transaction.

Other data could be included with the response.

The request and response data will be stored within the commercetools Payment Object to use on subsequent transactions.



4.3.3 Gift Card Pre-Auth

The difference between a gift card inquiry and a pre auth is that unlike an inquiry, a pre auth will decrease the available balance on the gift card.

When reserving the amount, it is assumed that an inquiry has previously been executed storing the PAN, expiry data and security number on the payment to prevent the need for the information to be re-entered by the customer.

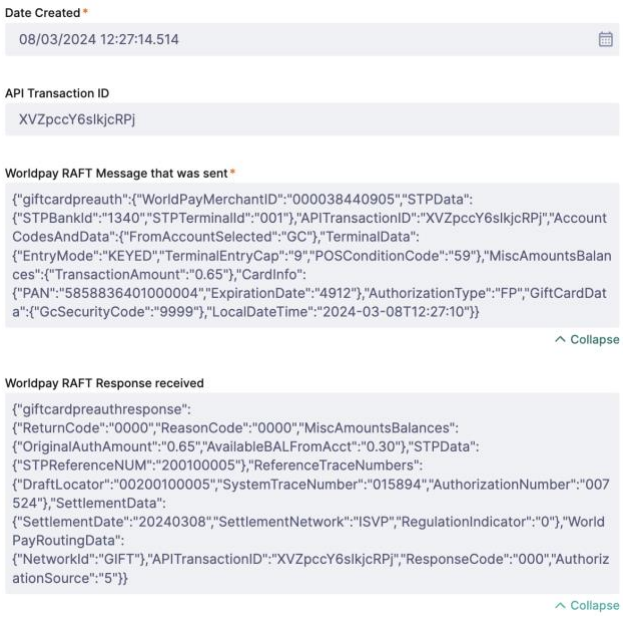
The transaction is normally initiated at the point the customer enters their final payment information and selects 'pay now', however the trigger for initiation is defined by you as the merchant.

The full overview of the gift card pre auth message can be found here, but the fields that need to be included on the `giftcardpreauth` request include:

Field	Description
WorldPayMerchantID	The full Merchant ID assigned by Worldpay (i.e. MID).
APITransactionID	Worldpay uses this for transaction matching (reversals, completions, and so on) and tracking. If you are initiating a subsequent message that ties back to an original request, provide the `APItransactionID` of the original transaction.
AuthorizationNumber	This field contains a value generated by the authorizing processor to indicate their acceptance of the transaction. If a value is not generated by either Worldpay or the network on approved transactions, Worldpay will generate one and return it to the merchant.
TransactionAmount	This field contains the amount of funds requested or remitted by the cardholder in the local currency of the transaction as represented by the acquirer currency code if provided.
PAN	This is the gift card number entered by the customer.
FromAccountSelected	Account selected at the terminal by the customer. This should show the value GC, it will be defaulted if not provided based on other information in the message.
ToAccountSelected	Account selected at the terminal by the customer. This should show the value GC, it will be defaulted if not provided based on other information in the message.
GcSecurityCode	Similar to the CVV2 value, Worldpay denies the transaction if you use an invalid code. Additionally, three failed Security Code attempts place a lock on the card. You can only remove it with a successful transaction where the security code is not present and is not required. If the code is present, whether it is required or not, Worldpay validates it.

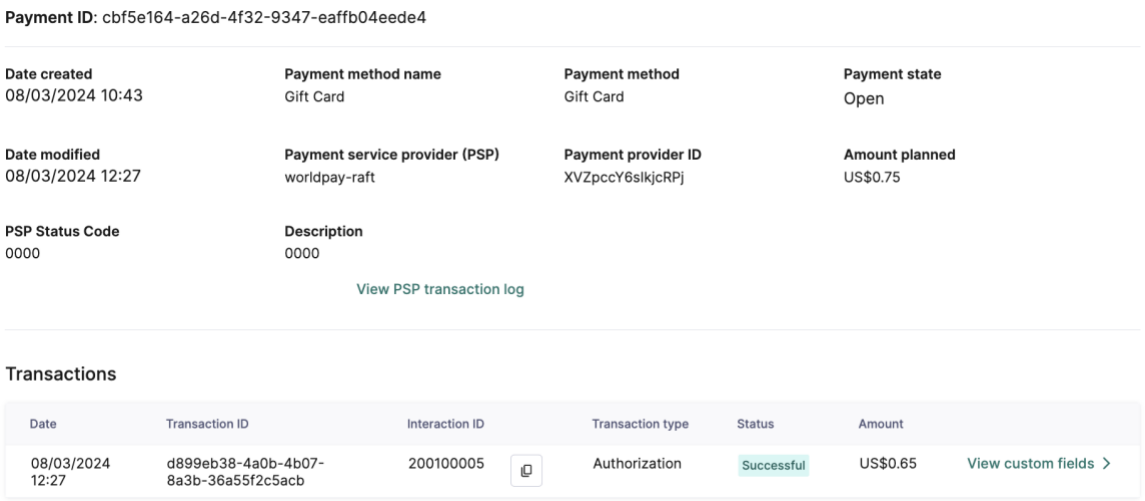
ExpirationDate	In YYMM format, Worldpay recommends that merchants provide expiry date with every transaction using a Token.
STPBankId	Contains the 4-digit bank id assigned by Worldpay during boarding.
STPTerminalId	Contains the 3-digit terminal id assigned by Worldpay during boarding.
LocalDateTime	This field contains the merchant's local date and time in a YYYY-MM-DDTHH:mm:ss format.
EntryMode	Indicates the entry mode for the transaction.
POSConditionCode	A value that describes the overall environment in which the transaction takes place.
TerminalEntryCap	Indicates by which the card data was read.

As with the inquiry, once constructed, the request is then sent via the connector to worldpay to process and will be logged in the `worldpay-raft-payment-interaction` object within commercetools to track the payment interaction.



Worldpay will then send a synchronous response confirming whether the request was successfully received, which based on the response would then update the transaction line with the `TransactionType` of `Authorization`, with the `TransactionState` from `Initial` to either

`Success` or `Failure`. As well as updating the transaction, the response will be logged in the `worldpay-raft-payment-interaction` object within commercetools.



4.3.3.1 Gift Card Pre-Auth Reversal

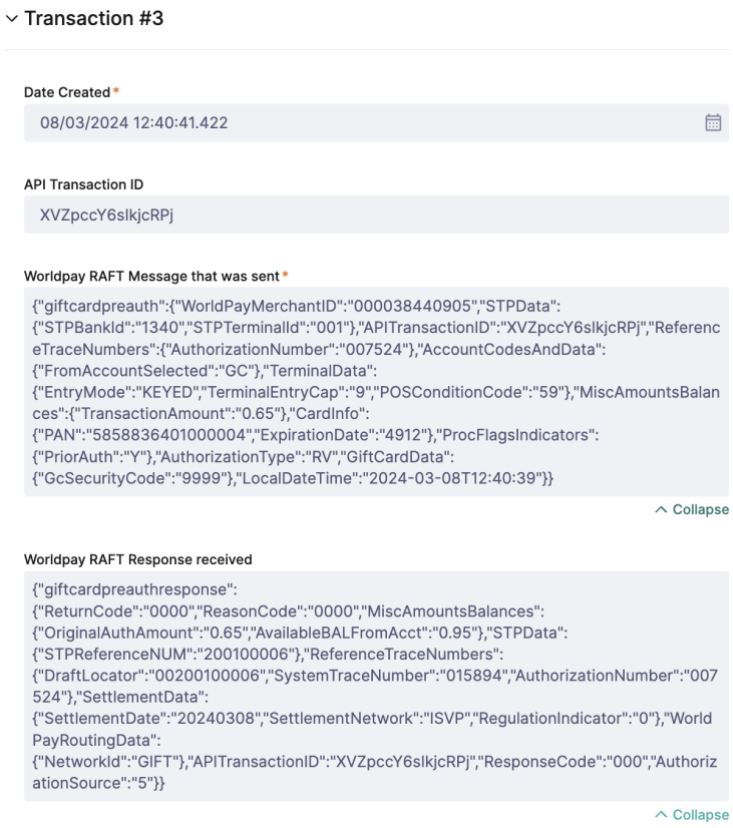
A pre auth reversal is required when there is a need to release the hold on funds provisioned on a gift card prior to a completion request being submitted.

The trigger to formulate the reversal will be outside the scope of the connector and need to be configured (initiating the `cancelPayment` request from the API), but once the conditions have been met, the initiation of the transaction will occur.

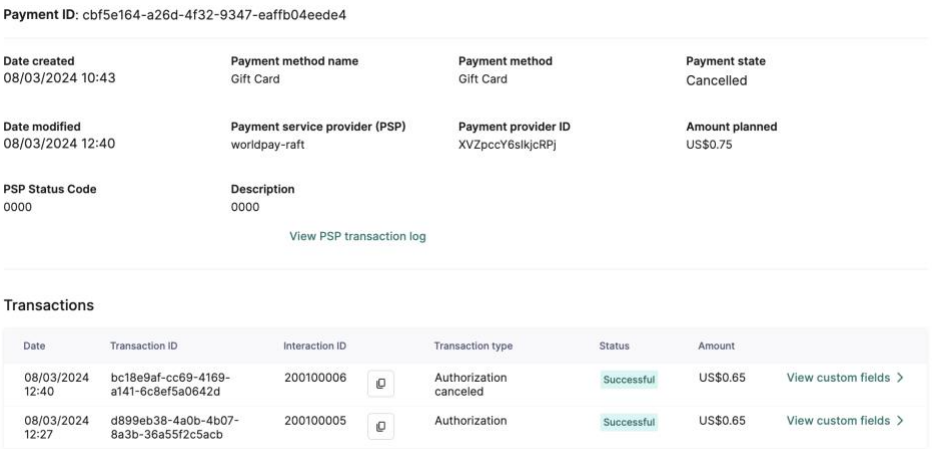
The request sent to initiate a `giftcardpreauth` reversal is the same as the pre auth request with two fields being the exception:

Field	Description
`AuthorizationType`	Unlike a standard pre-auth, for a reversal, this must be set to `RV`

As with the inquiry, once constructed, the request is then sent via the connector to worldpay to process and will be logged in the `worldpay-raft-payment-interaction` object within commercetools to track the payment interaction.



Worldpay will then send a synchronous response confirming whether the request was successfully received, which based on the response would then update the transaction line with the `TransactionType` of `Authorizationcanceled`, with the `TransactionState` from `Initial` to either `Success` or `Failure`. As well updating the transaction, the response will be logged in the `worldpay-raft-payment-interaction` object within commercetools.



4.3.4 Gift Card Completion

To ensure that the gift card can be completed, multiple conditions must be met:

1. A gift card payment has already been authorized, as a result a payment object with `TransactionType=Authorization` with `TransactionState=Success` has been created in commercetools.
2. A payment transaction has been created for all or some of the original authorization value with the `TransactionType=Charge` and `TransactionState=Initial` in commercetools).

The trigger to formulate the gift card completion request will be outside the scope of the connector and need to be configured (initiating the `chargePayment` request from the API), but once the conditions have been met, the initiation will occur.

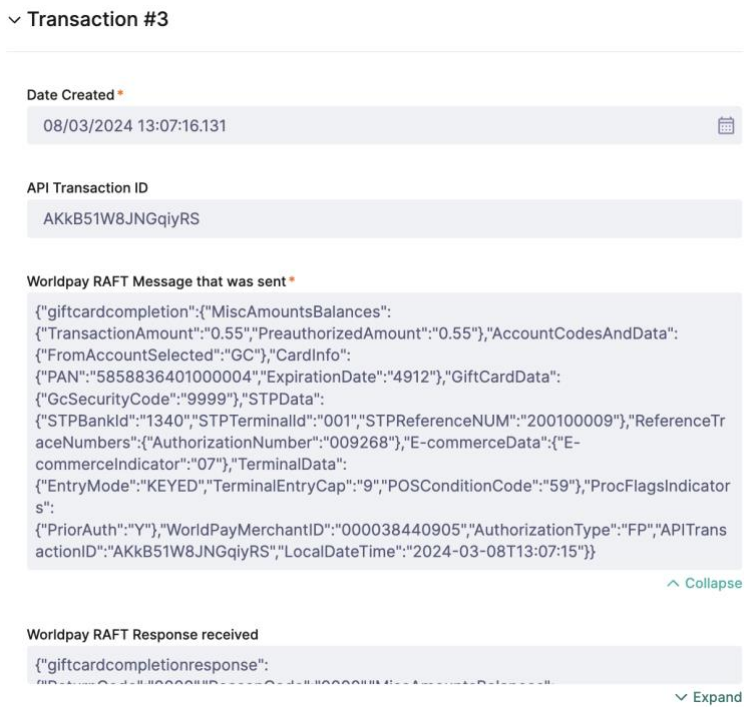
The `giftcardcompletion` has set mandatory fields that will be collated from the previous transaction against the order in commercetools which include:

Field	Description
WorldPayMerchantID	The full Merchant ID assigned by Worldpay (i.e. MID).
APITransactionID	Worldpay uses this for transaction matching (reversals, completions, and so on) and tracking. If you are initiating a subsequent message that ties back to an original request, provide the `APItransactionID` of the original transaction.
TransactionAmount	This field contains the amount of funds requested or remitted by the cardholder in the local currency of the transaction as represented by the acquirer currency code if provided.
PreauthorizedAmount	For previously authorized transactions (preauth completion, credit card completion), the acquirer places the amount that the transaction was originally authorized for in this field. For gift card purchases, this field can be sent by the acquirer to indicate the amount that was pre-authorized.
PAN	This is the gift card number entered by the customer.
FromAccountSelected	Account selected at the terminal by the customer. This should show the value GC, it will be defaulted if not provided based on other information in the message.
ToAccountSelected	Account selected at the terminal by the customer. This should show the value GC, it will be defaulted if not provided based on other information in the message.

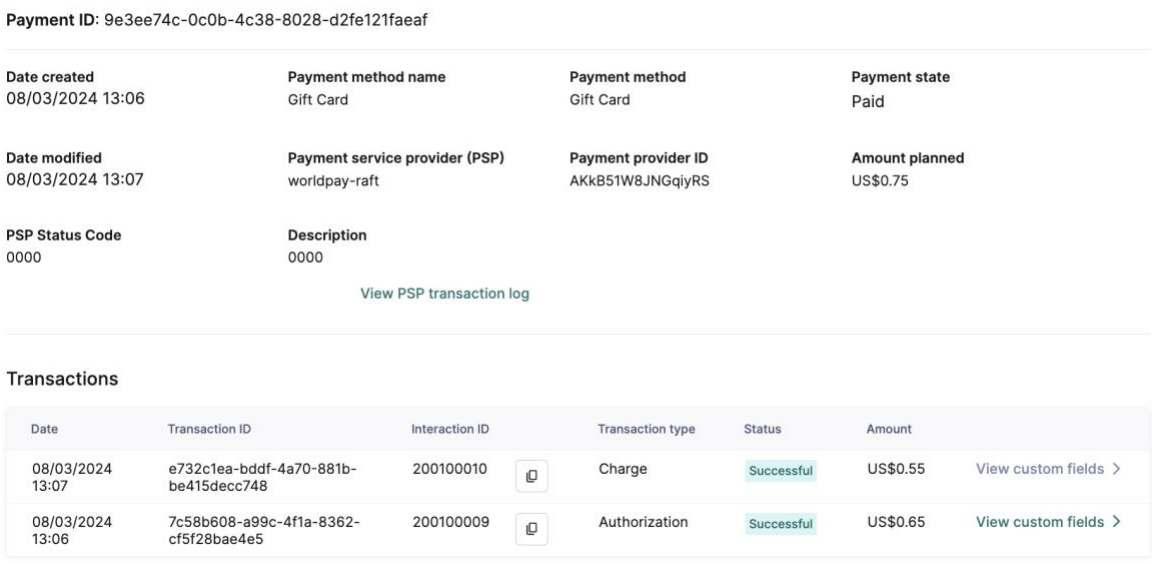
GcSecurityCode	Similar to the CVV2 value, Worldpay denies the transaction if you use an invalid code. Additionally, three failed Security Code attempts place a lock on the card. You can only remove it with a successful transaction where the security code is not present and is not required. If the code is present, whether it is required or not, Worldpay validates it.
ExpirationDate	In YYMM format, Worldpay recommends that merchants provide expiry date with every transaction using a Token.
PriorAuth	Y/N flag indicating if the transaction is a prior authorized transaction.
STPBankId	Contains the 4-digit bank id assigned by Worldpay during boarding.
STPTerminalId	Contains the 3-digit terminal id assigned by Worldpay during boarding.
STPReferenceNUM	Worldpay assigned reference number on each transaction that is returned to the terminal in the CreditCompletion. Must be sent on follow-up messages to match back to the original transaction.
AuthorizationType	Provides a means for the transaction disposition to be changed from standard authorization to forced conditions. This must be set to `FP`
LocalDateTime	This field contains the merchant's local date and time in a YYYY-MM-DDTHH:mm:ss format.
EntryMode	Indicates the entry mode for the transaction.
POSConditionCode	A value that describes the overall environment in which the transaction takes place.
TerminalEntryCap	Indicates by which the card data was read.

Full details on other fields that can be included can be found here.

Once constructed, the request is then sent via the connector to worldpay to process and will be logged in the `worldpay-raft-payment-interaction` object within commercetools to track the payment interaction.



Worldpay will then send a synchronous response confirming whether the request was successfully received, which based on the response would then update the transaction line with the `TransactionType` of `Charge`, with the `TransactionState` from `Initial` to either `Success` or `Failure`. As well updating the transaction, the response will be logged in the `worldpay-raft-payment-interaction` object within commercetools.



4.3.5 Gift Card Refund

To ensure that the gift card refund can be initiated, multiple conditions must be met:

1. A gift card payment has already been completed, as a result a payment object with `TransactionType=Charge` with `TransactionState=Success` has been created in commercetools.
2. A payment transaction has been created for all or some of the original completion value with the `TransactionType=Refund` and `TransactionState=Initial` in commercetools.

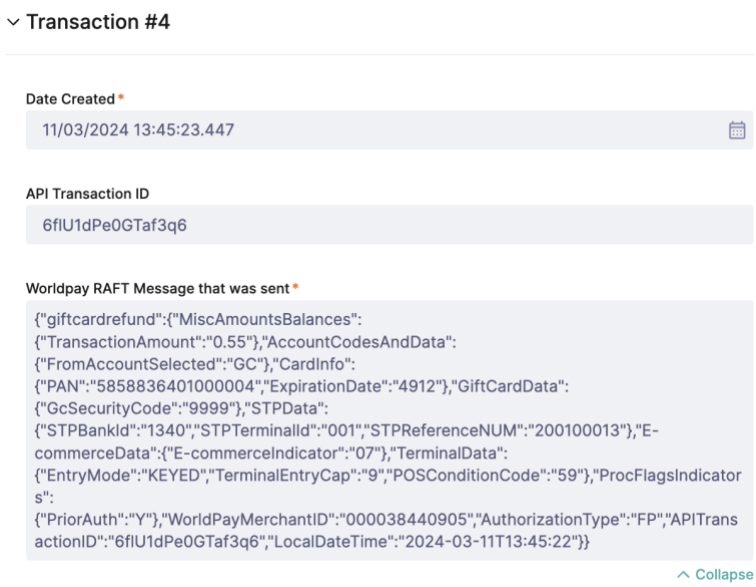
The trigger to formulate the gift card refund request will be outside the scope of the connector and need to be configured (initiating the `refundPayment` request from the API), but once the conditions have been met, the initiation will occur.

The fields expected on the `giftcardrefund` request are the same as a `giftcardcompletion`, except for the following field which should not be present in the refund request:

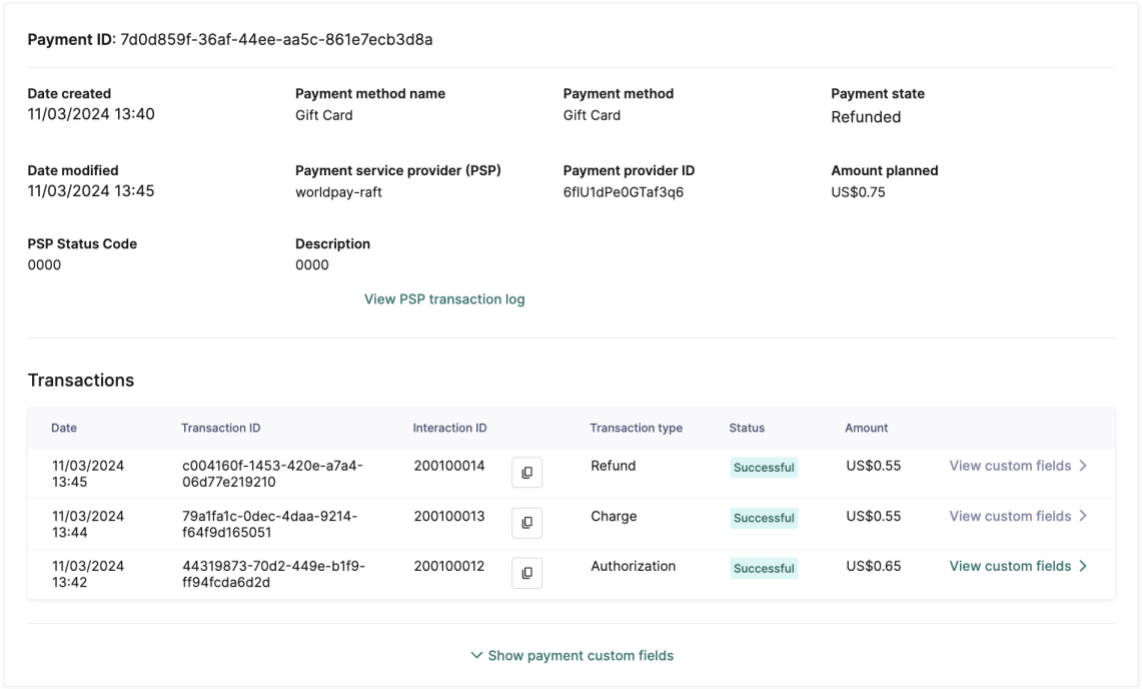
Field	Description
`PreauthorizedAmount`	For previously authorized transactions (preauth completion, credit card completion), the acquirer places the amount that the transaction was originally authorized for in this field. For gift card purchases, this field can be sent by the acquirer to indicate the amount that was pre-authorized.

Full details on other fields that can be included can be found here.

Once constructed, the request is then sent via the connector to worldpay to process and will be logged in the `worldpay-raft-payment-interaction` object within commercetools to track the payment interaction.



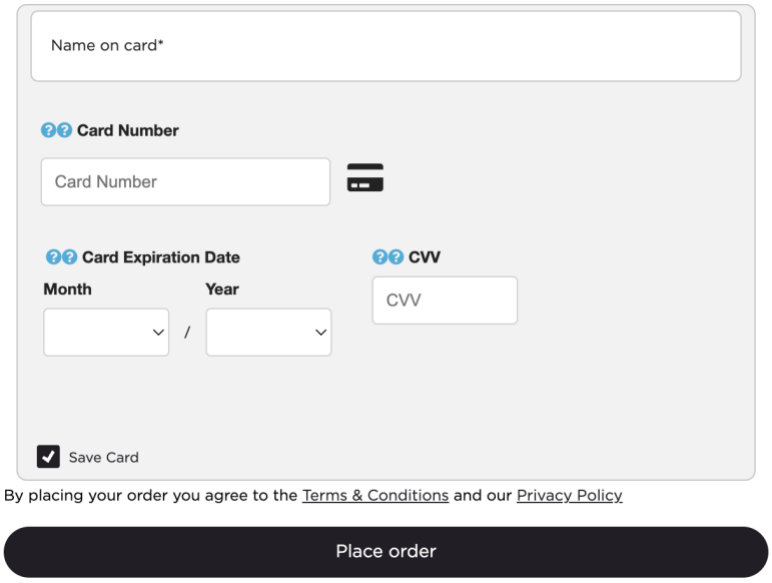
Worldpay will then send a synchronous response confirming whether the request was successfully received, which based on the response would then update the transaction line with the `TransactionType` of `Refund`, with the `TransactionState` from `Initial` to either `Success` or `Failure`. As well updating the transaction, the response will be logged in the `worldpay-raft-payment-interaction` object within commercetools.



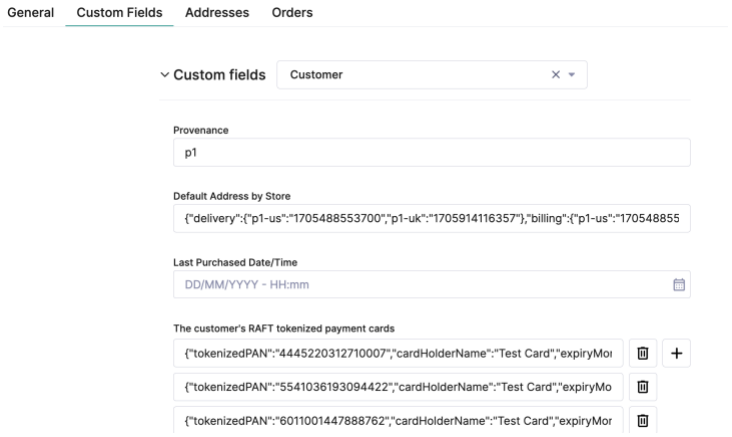
5 Supplementary Configurations

5.1 Card on File (stored cards) Transactions

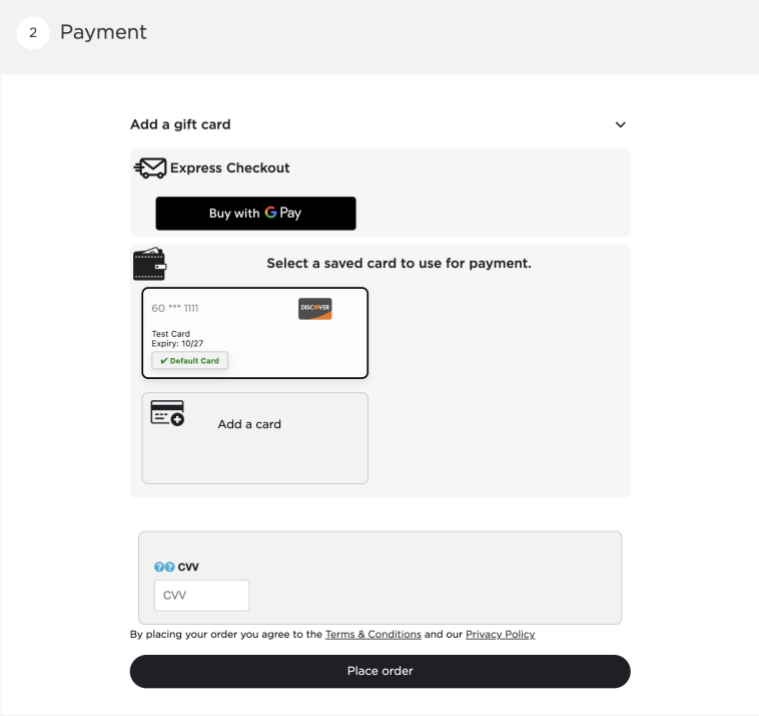
The tokenization of a customer's card PAN from the LowValueToken that occurs within the authorization (`creditAuth`) process, allows you to securely store shopper's payment details for future payments. To do this, when the customer adds a new card during checkout, you can expose the ability to 'save' the card to use on future orders.



When the customer selects this option, the `tokenizedPAN` returned in the `creditAuth` response will be stored against the customer's profile within commercetools:



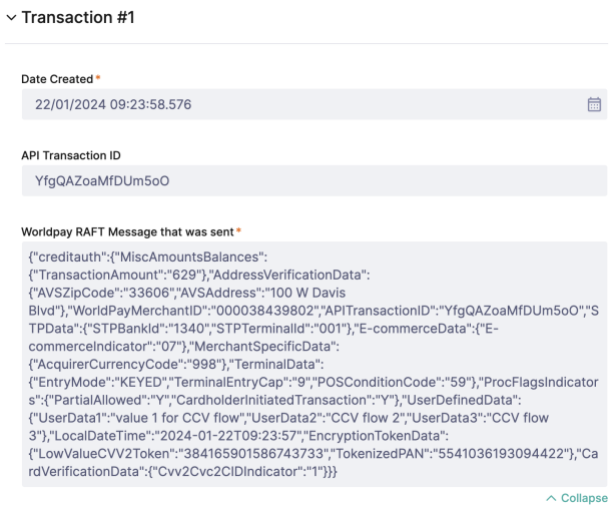
The next time the customer then returns to your checkout, they can select to pay via a previously stored card and validate the card by entering a CVV number (presented by the eProtect iFrame).



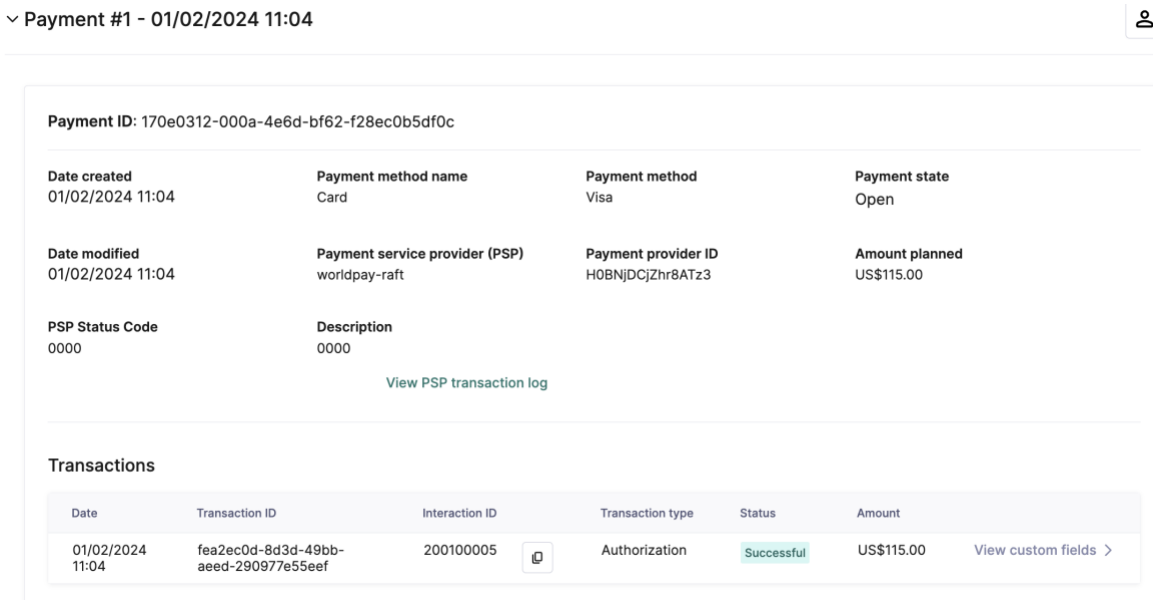
Upon entering their CVV and selecting to 'Place Order', the `creditAuth` request will be initiated.

The format of this message will be the same as outlined in the Payment Authorization section, however there will be 2 small differences:

1. Instead of sending a request containing a `LowValueToken`, the field sent will be the `tokenizedPan`. This will contain the `tokenizedPan` that was received and stored against the customer profile.
2. As well as the `tokenizedPan`, the request should contain a `lowValueCVV2Token`, this is returned from eProtect as a `CheckoutID`.



As per the standard authorization, once a successful response has been received, the transaction will store as a payment transaction against the order in commercetools.



5.2 Multi-Completion

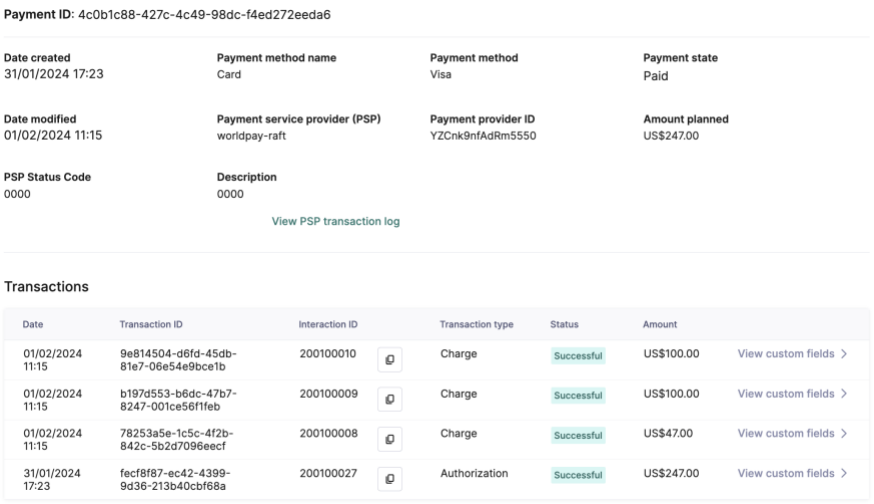
The same request format (as per the sending a completion on an authorized payment section) will be used regardless of whether a full or partial completion is being submitted, the following 3 fields should be added to the request which contain the relevant indicators as to whether one or more follow up transactions are expected:

Field	Description
`SequenceNumber_00-99`	Indicates which shipment is being sent. Use 00 for initial auth.
`SequenceCount_01-99`	Indicates the total number of shipments (set to 99 for all shipments except the final shipment, which should send the number of actual completions sent)
`FinalShipment`	Y/N flag indicating this transaction is the final shipment of an order.

With each completion request, the values in the `SequenceNumber_00-99` field should increase by 01 (i.e. / 01 ,02, 03, etc.)

Once the `FinalShipment` indicator has been sent with a `Y` value, RAFT does not expect any further requests pertaining to that original authorization.

Each completion request will be visible in commercetools as a separate ‘Charge’ transaction on the payment transactions screen within the commercetools Merchant Centre:



6 Troubleshooting

6.1 Timeouts when waiting for a response.

As with all API’s, there is a possibility that the connector will not receive the required response within the expected time. To mitigate this, the connector has a timeout-handler. A timeout-handler is an endpoint (depicted as a separate application) that deals with timeouts and retries.

To ensure the timeout-handler works as expected, the following configuration parameters are required:

Property Name	Type	Description
`commercetools.clientId`	String	A commercetools Client ID to log on to commercetools.
`commercetools.clientSecret`	String	A commercetools Client Secret, required authorization for `manage:payments`.
`commercetools.projectKey`	String	The commercetools project Key.
`commercetools.region`	String	The commercetools region.
`commercetools.timeoutMs`	Number	The number of milliseconds before a commercetools connection times out.

CTP_SCOPES	String	The commercetools scopes for the API requests. If this is left blank, that API client is given all the scopes assigned to the API client in commercetools.
worldpayRaftTimeoutMs	Number	The timeout of the call to the Worldpay RAFT service by the connector. Note that commercetools also has a timeout for invocation of the connector itself. In case the latter exceeds the configuration, commercetools will abort the transaction before the connector does and the result could be that the RAFT action was executed, but the results were never persisted in commercetools. To prevent this, please define a value in the configuration that is below the commercetools timeout to avoid unpredictable failures.
worldpayRaft.maxRetries	Number	The maximum number of attempts to re-send a timed-out message before reverting the message.
worldpayRAFT.reverseMaxRetries	Number	The maximum number of attempts to re-send a reversal message if it times out.

As described in the deployment guide, the amount of time over which a retry will be attempted depends on how often the timeout-handler is called. For example, when it runs every minute as an interval with a `retryCount` of 5, the system will retry for 5 minutes. But if the interval is changed to every 30 seconds, it will retry for only 2 and a half minutes.

A timeout is handled in the connector by moving the relevant transaction to Pending in commercetools.

The timeout-handler component checks commercetools for payments with pending transactions. For each one it finds, it triggers a retry by updating a `retryCount` property in the transaction. If a timeout happens in the connector, the request field is retained. Updating the payment transaction (by updating the `retryCount`) will cause an event, and with the retained request, the connector will send the same message to RAFT.

If a response is received from RAFT, then the transaction is updated in commercetools, where the payment status will move from `pending` to either `success` or `failure` (this change will remove the transaction from the retry cycle.

If no response is received, once the maximum number of retries (`maxTries`) is reached, the timeout-handler modifies the message in the request field by changing the `AuthorizationType`

to `RV - Reversal` and updates the `retryCount` to a negative number. This indicates the reversal of the original message has been initiated.

As with the initial request, the reversal attempts, also have a set number retries, this is configured in the `reverseMaxRetries` parameter.

When a response is received for the reversal, this should update the original transaction's `transactionState` from `pending` to `failure`. To flag that it did not complete as intended.

7 Appendix A. Related documentation

#	Document Title	Location / URL	Author
1	Native RAFT API Documentation	https://developerengine.fisglobal.com/apis/native-raft	Worldpay
2	CreditAuthorization API Document	https://developerengine.fisglobal.com/apis/native-raft/credit/api-specification#tag/Credit/operation/CreditAuthorization	Worldpay
3	CreditCompletion API Document	https://developerengine.fisglobal.com/apis/native-raft/credit/api-specification#tag/Credit/operation/CreditCompletion	Worldpay
4	CreditRefund API Document	https://developerengine.fisglobal.com/apis/native-raft/credit/api-specification#tag/Credit/operation/CreditRefund	Worldpay
5	Commercetools Configuration Setup	https://github.com/Worldpay/Worldpay-CommerceTools-RAFT/tree/feature/initial-version/resources/commercetools	Gradient Edge
6	Commercetools Project Installation	https://github.com/Worldpay/Worldpay-CommerceTools-RAFT/blob/feature/initial-version/resources/docs/TechnicalDocumentation.md#installation	Gradient Edge
7	eProtect Integration Guide	https://developerengine.fisglobal.com/apis/usecomm/eprotect	Worldpay
8	Apple Pay Configuring Your Environment	https://developer.apple.com/documentation/apple_pay_on_the_web/configuring_your_environment	Apple
9	Google Pay Tutorial	https://developers.google.com/pay/api/web/guides/tutorial	Google
10	Timeouts and Reversals	https://github.com/Worldpay/Worldpay-CommerceTools-RAFT/blob/feature/initial-version/resources/docs/DeploymentGuide.md#timeouts-and-retries	Gradient Edge

11	Commercetools HTTP API	https://docs.commercetools.com/api/projects/payments#create-payment	commercetools
12	Commercetools API Extensions	https://docs.commercetools.com/api/projects/api-extensions	commercetools
13	eProtect Integration Options	https://developerengine.fisglobal.com/apis/usecomm/eprotect/introduction/overview	Worldpay
14	Gift card inquiry API	https://developerengine.fisglobal.com/apis/native-raft/giftcard/api-specification#tag/Giftcard/operation/GiftcardInquiry	Worldpay
15	Gift card Pre-Authorization API	https://developerengine.fisglobal.com/apis/native-raft/giftcard/api-specification#tag/Giftcard/operation/GiftcardPreauth	Worldpay
16	Gift Card Completion API	https://developerengine.fisglobal.com/apis/native-raft/giftcard/api-specification#tag/Giftcard/operation/GiftcardCompletion	Worldpay
17	Gift Card Refund API	https://developerengine.fisglobal.com/apis/native-raft/giftcard/api-specification#tag/Giftcard/operation/GiftcardRefund	Worldpay
18	Architecture Diagram	https://github.com/Worldpay/Worldpay-CommerceTools-RAFT/blob/feature/initial-version/resources/docs/architecture.drawio.svg	Gradient Edge
19	Standard Payment Flow Diagram	https://github.com/Worldpay/Worldpay-CommerceTools-RAFT/blob/feature/initial-version/resources/docs/Standard%20payment%20flow.drawio.svg	Gradient Edge
20	Gift Card Flow Diagram	https://github.com/Worldpay/Worldpay-CommerceTools-RAFT/blob/feature/initial-version/resources/docs/Gift%20card%20payment%20flow.drawio.svg	Gradient Edge
21	Apple Pay Flow Diagram	https://github.com/Worldpay/Worldpay-CommerceTools-RAFT/blob/feature/initial-version/resources/docs/Apple%20Pay%20payment%20flow.drawio.svg	Gradient Edge
22	Google Pay Flow Diagram	https://github.com/Worldpay/Worldpay-CommerceTools-RAFT/blob/feature/initial-version/resources/docs/Google%20Pay%20payment%20flow.drawio.svg	Gradient Edge

23	Deployment Guide	https://github.com/Worldpay/Worldpay-CommerceTools-RAFT/blob/feature/initial-version/resources/docs/DeploymentGuide.md	Gradient Edge
24	Postman Collection	https://github.com/Worldpay/Worldpay-CommerceTools-RAFT/tree/feature/initial-version/resources/postman	Gradient Edge
25	Technical Documentation	https://github.com/Worldpay/Worldpay-CommerceTools-RAFT/blob/feature/initial-version/resources/docs/TechnicalDocumentation.md	Gradient Edge

8 Appendix B. Terminology

Term	Abbreviation	Description
commercetools API extensions	n/a	An API Extension gets called after the processing of a create or update request of an API call, but before the result is persisted. The API Extension can validate the object or apply additional updates to it.
Node.js	n/a	Node.js is a cross-platform, open-source JavaScript runtime environment that can run on Windows, Linux, Unix, macOS, and more. Node.js runs on the V8 JavaScript engine and executes JavaScript code outside a web browser. Node.js lets developers use JavaScript to write command line tools and for server-side scripting.
Amazon Web Services	AWS	Amazon Web Services is a subsidiary of Amazon that provides on-demand cloud computing platforms and APIs to individuals, companies, and governments, on a metered, pay-as-you-go basis. Clients will often use this in combination with autoscaling.
Docker Container	n/a	A container is a standard unit of software that packages up code and all its dependencies, so the application runs quickly and reliably from one computing environment to another. A Docker container image is a lightweight, standalone, executable package of software that includes everything needed to run an application: code, runtime, system tools, system libraries and settings.
Payment Card Industry Data Security Standard	PCI DSS	The Payment Card Industry Data Security Standard is an information security standard used to handle credit cards from major card brands. The standard is administered by the Payment Card Industry Security Standards Council, and its use is mandated by the card brands.
Payment Services Directive Two	PSD2	Payment Services Directive Two (PSD2) is a piece of legislation designed to force providers of payment services to improve customer authentication processes and to also bring in new regulation around third-party involvement.
Strong Customer Authentication	SCA	Strong Customer Authentication (SCA) is a new requirement of the second Payment Services Directive (PSD2), which aims to add extra layers of security to electronic payments. SCA will apply to the European Economic Area (EEA) and the United Kingdom and is likely to continue to apply in the UK after the Brexit transition period. It will require banks to perform additional checks when consumers make payments to confirm their identity. To do this, banks may ask for a combination of two forms of identification at checkout.

Infrastructure as code	IaC	Infrastructure as Code (IaC) is the managing and provisioning of infrastructure through code instead of through manual processes. With IaC, configuration files are created that contain your infrastructure specifications, which makes it easier to edit and distribute configurations. It also ensures that you provision the same environment every time.
iFrame API	n/a	An iFrame is an HTML element that is embedded naturally into your website or online shop. Using an iFrame is a great option for a hosted payment page, as it reduces the amount of redirection the customer must go through, making the overall checkout process much smoother and more seamless.
Synchronous responses	n/a	A Synchronous Response is a response given while a customer and agent are present in the same platform, at the same time. This enables both sides to have real-time responses and updates that are needed to resolve an inquiry or issue.
Asynchronous notifications	n/a	Asynchronous responses are returned to the client in subsequent connections, freeing the client to send other requests instead of blocking while waiting for the response.
Configuration Parameter	n/a	Configuration parameters are user-definable settings that control various aspects of a server's behavior. The server supplies default values for all configuration parameters. You can use configuration parameters to tailor a server for an installation's particular needs.
Serverless Function	n/a	A serverless function is essentially a piece of business logic that is both stateless (does not maintain data) and ephemeral (is used and destroyed). A serverless function potentially lasts only for seconds and is designed to be triggered by a specific condition.
AWS SecretsManager	n/a	AWS Secrets Manager is a secrets management service that helps you protect access to your applications, services, and IT resources. This service enables you to easily rotate, manage, and retrieve database credentials, API keys, and other secrets throughout their lifecycle.
Azure KeyVault	n/a	Azure Key Vault is a cloud service that provides a secure store for secrets. You can securely store keys, passwords, certificates, and other secrets.
AWS Identity and Access Management Permissions	IAM Permissions	AWS Identity and Access Management (IAM) provides you with fine-grained access control to help you establish permissions that determine who can access which AWS resources under which conditions. Use fine-grained access control to help secure your AWS resources on your journey to achieve least privilege.

Secrets	n/a	Secrets typically refer to digital authentication credentials which are access credentials like API Keys, Credential pairs, and security certificates to name a few.
Liability Shift	n/a	Liability shift refers to the responsibility of covering the losses from fraudulent transactions moving from the merchant to the issuing bank.
Certificate Signing Request	CSR	A CSR is a specially formatted encrypted message containing a public key pair in your keychain. This allows Apple to certify that the subject of this certificate holds the private key that matches the public key embedded in this certificate.
Payment Sheet	n/a	During checkout, a payment sheet can show the credit or debit card linked to Apple Pay, purchase amount (including tax and fees), shipping options, and contact information.
Browser Based Pop Up	n/a	Pops up an actual browser window for which it displays its content driven by Google's JavaScript Library.
Representational State Transfer APIs	REST APIs	Representational State Transfer (REST) is a software architecture that imposes conditions on how an API should work. REST was initially created as a guideline to manage communication on a complex network like the internet. You can use REST-based architecture to support high-performing and reliable communication at scale. You can easily implement and modify it, bringing visibility and cross-platform portability to any API system.
Representational State Transfer Design	REST Design	The REST pattern allows the client and the server to be implemented independently without the knowledge of the other entity. This means that code at either side can be modified without having to worry about the effect of the modification on the other side.
JavaScript Object Notation Data Interchange	JSON Data Interchange	JSON is a text-based, lightweight, and human-readable format for data exchange between clients and servers. JSON is derived from JavaScript and bears a close resemblance to JavaScript objects, but it is not dependent on JavaScript.
HyperText Transfer Protocol	HTTP	HTTP is an application layer protocol designed to transfer information between networked devices and runs on top of other layers of the network protocol stack. It is the foundation of the World Wide Web, used to load webpages using hypertext links.
Source Code Repository	n/a	A code repository is a storage location for code and other software development assets, such as documentation, tests, and scripts. They are often used to manage and organize a software project's codebase and collaborate with other project developers.
Extension Module	n/a	Extension modules are components which extend the standard scope of delivery of an application / product.

Card Primary Account Number	Card PAN	The term primary account number refers to a 14-, 15-, 16-, or even up to 19-digit number that serves as a unique identifier on credit cards, debit cards, and other types of payment cards. Also known as a payment card number, it is typically either embossed or laser-printed on the front of the card.
Extranet	n/a	Extranet is an intranet that can be partially accessed by authorized outside users, enabling businesses to exchange information over the internet in a secure way.
Application Programming Interface Layer	API Layer	An API layer is the part of the backend Application that encapsulates all the Programming logic required to receive and send data through an Interface (API). In other words, all external API calls a backend application relies on pass through this architectural layer.1
Data Model Extensions	n/a	A data model extension is where you extend a data models existing entities to define additional relationships, or you can create new entities that define relationships to other entities.
Javascript Customer Browser API	n/a	Browser APIs are built into your web browser and can expose data from the browser and surrounding computer environment.
Mobile API	n/a	Mobile APIs are APIs used in mobile apps to allow developers to access another platform or application.
Transformers Package	n/a	Transformers provides APIs and tools to easily download and train state-of-the-art pretrained models. Using pretrained models can reduce your compute costs, carbon footprint, and save you the time and resources required to train a model from scratch.
ISO 4217 Format	n/a	ISO 4217 is a standard published by the International Organization for Standardization that defines alpha codes and numeric codes for the representation of currencies and provides information about the relationships between individual currencies and their minor units.
Card Bank Identification Number	Card BIN	A bank identification number (BIN) represents the first four to six digits on a credit card. The first four to six digits identify the financial institution that issued the card. The BIN is a security measure to protect both consumers and merchants engaging in online transactions.
Connection Timeout	n/a	A connection timeout refers to the maximum amount of time a system or application is willing to wait for a connection to be established with another system or service. It is a predefined duration set to limit the waiting time for a connection to be established successfully.

9 Appendix C. Payment Resource Model