RIVERTY

# Riverty – Recommended Integration Guide

November 28, 2022

RIVERTY

# Agenda

# 1 Introduction

This document explains how to implement the payment methods of Riverty for Germany, Austria, and Switzerland through our REST-API. The direct integration of Riverty's Payment API allows merchants to tailor the way they enable Riverty. Direct integration provides access to full functionality and makes it possible to use the API's rich meta-data. We only provide the most important details for the direct integration of Riverty in this document. If you want to get a full overview of all our functionality, visit developer.riverty.com or get in touch with our team.

## 1.1 Payment API

The REST-based Payment API lets merchants integrate Riverty into their shopping experience. The Payment API displays methods for checkout services and order management using a familiar structure similar to traditional processing of card transactions, with requests to authorize and capture an order. In addition, the Riverty API supports more metadata than a conventional Payment API. To support a better shopper experience, Riverty API helps merchants maintain consistent branding throughout the full payment experience by using the product and invoice header images (hero images). Moreover, Riverty makes it possible for merchants to leverage data in their analytics and digital marketing by supporting Google Analytics integration as well as adding structured data as Google Shopping Feed information.

## 1.2 Main components for implementation

Three different calls must be implemented in your system. Authorize checks the customer's details (including a credit check) and creates a reservation if the response is positive. Capture creates a previously authorized order invoice, usually done when the items have been shipped and the payment period begins. Refund refunds the customer his money in case of a return or a subsequent credit note.

## 1.3 Recommended Riverty Integration (Best Practice)

The following implementation of the Riverty API is based on best practices and highly recommended for best results.

Mandatory order of main calls of the Riverty integration

RIVERTY

# 2 API-Components

## 2.1 Reservation (authorizePayment)

Approves the payment for a specified customer and basket. Full fraud and credit scoring applied and responses final feedback of the customer acceptance. If the online-shop customer enters incorrect address data, the system delivers a suggestion for a corrected address.

**Riverty:** /api/v3/checkout/authorize (POST)

- Payment
  - Type (Invoice, Installment)
  - DirectDebit[1]
    - bankCode
    - bankAccount
- Customer
  - salutation
  - customerCategory (Company, Person)
  - customerNumber
  - firstName
  - lastName
  - email
  - birthdate
  - address
    - street
    - streetNumber
    - postalCode
    - postalPlace
    - countryCode
  - mobilPhone
  - riskData
    - ProfilTrackingId
    - IpAddress
- deliveryCustomer
  - same fields as customer
- Order
  - orderNumber
  - totalGrossAmount (maximum 2 numbers after the comma)
  - totalNetAmount (maximum 2 numbers after the comma)
  - currency
  - items
    - productid
    - description
    - grossUnitPrice (maximum 2 numbers after the comma)
    - netUnitPrice (maximum 2 numbers after the comma)
    - quantity
    - VatPercent
    - VatAmount (maximum 2 numbers after the comma)
    - ImageUrl
    - GroupId

## Example - Request 1: Create Reservation (same billing / delivery address)


```
{
    "payment": {
        "type": "Invoice"
    },
    "customer": {
        "salutation": "Mr",
        "firstName": "Max",
        "lastName": "Mustermann",
        "email": "maxmuster@arvatotest.de",
        "birthDate": "1990-12-31",
        "customerCategory": "Person",
        "customerNumber": "123456",
        "mobilePhone": "+49 1234 567890",
        "address": {
            "street": "Gütersloher Str.",
            "streetNumber": "123",
            "postalCode": "33415",
            "postalPlace": "Verl",
            "countryCode": "DE"
        }
    },
    "order": {
        "number": "orderNoIndividual001",
        "totalGrossAmount": 20.0,
        "totalNetAmount": 16.40,
        "currency": "EUR",
        "items": [
            {
                "productId": "0001",
                "description": "Testproduct 1",
                "grossUnitPrice": 10.0,
                "netUnitPrice": 8.4,
                "quantity": 2,
                "GroupId": "21",
                "VatPecent": 19.0,
                "VatAmount": 1.6,
                "ImageUrl": "http://i.imgur.com/r9tKrfM.jpg"
            },
            {
                "description": "Shipping fee",
                "grossUnitPrice": 0.0,
                "productId": "shippingFees",
                "quantity": 1,
                "type": "ShippingFee"
            }
        ],
        "riskData": {
            "ProfileTrackingId": "12456789",
            "IpAddress": "127.0.0.1"
        }
    }
}
```

## Example - Request 2: Create Reservation with discount / voucher


```
{
    "payment": {
        "type": "Invoice"
    },
    "customer": {
        "salutation": "Mr",
        "firstName": "Max",
        "lastName": "Mustermann",
        "email": "maxmuster@arvatotest.de",
        "birthDate": "1990-12-31",
        "customerCategory": "Person",
        "customerNumber": "123456",
        "mobilPhone": "#49 1234 567890",
        "address": {
            "street": "Gütersloher Str.",
            "streetNumber": "123",
            "postalCode": "33415",
            "postalPlace": "Verl",
            "countryCode": "DE"
        }
    },
    "order": {
        "number": "orderNoIndividual002",
        "totalGrossAmount": 15.0,
        "totalNetAmount": 12.60,
        "currency": "EUR",
        "items": [
            {
                "productId": "0001",
                "description": "Testproduct 1",
                "grossUnitPrice": 10.0,
                "netUnitPrice": 8.4,
                "quantity": 2,
                "GroupId": "21",
                "VatPecent": 19.0,
                "VatAmount": 1.6,
                "ImageUrl": "http://i.imgur.com/r9tKrfM.jpg"
            },
            {
                "productId": "voucher",
                "description": "-25% Voucher",
                "grossUnitPrice": -5.0,
                "netUnitPrice": -4.2,
                "quantity": 1,
                "VatPercent": 19.0,
                "VatAmount": -0.8,
                "ImageUrl": "http://i.imgur.com/r9tKrfM.jpg"
            }
        ],
        "riskData": {
            "ProfileTrackingId": "12456789",
            "IpAddress": "127.0.0.1"
        }
    }
}
```

## Example - Request 3: Different delivery address


```
{
    "payment": {
        "type": "Invoice"
    },
    "customer": {
        "salutation": "Mr",
        "firstName": "Max",
        "lastName": "Mustermann",
        "email": "maxmuster@arvatotest.de",
        "birthDate": "1990-12-31",
        "customerCategory": "Person",
        "customerNumber": "123456",
        "mobilPhone": "+49 1234 567890",
        "address": {
            "street": "Gütersloher Str.",
            "streetNumber": "123",
            "postalCode": "33415",
            "postalPlace": "Verl",
            "countryCode": "DE"
        }
    },
    "deliveryCustomer": {
        "salutation": "Mr",
        "firstName": "Max",
        "lastName": "Mustermann",
        "email": "maxmuster@arvatotest.de",
        "birthDate": "1990-12-31",
        "customerCategory": "Person",
        "address": {
            "street": "Gütersloher Str.",
            "streetNumber": "123",
            "postalCode": "33415",
            "postalPlace": "Verl",
            "countryCode": "DE"
        }
    },
    "order": {
        "number": "orderNoIndividual003",
        "totalGrossAmount": 20.0,
        "totalNetAmount": 16.40,
        "currency": "EUR",
        "items": [
            {
                "productId": "0001",
                "description": "Testproduct 1",
                "grossUnitPrice": 10.0,
                "netUnitPrice": 8.4,
                "quantity": 2,
                "GroupId": "21",
                "VatPecent": 19.0,
                "VatAmount": 1.6,
                "ImageUrl": "http://i.imgur.com/r9tKrfM.jpg"
            }
        ],
        "riskData": {
            "ProfileTrackingId": "12456789",
            "IpAddress": "127.0.0.1"
        }
    }
}
```

## Example - Request 4: Delivery to DHL Packstation (Pick-Up-Point)


```
{
    "payment": {
        "type": "Invoice"
    },
    "customer": {
        "salutation": "Mr",
        "firstName": "Max",
        "lastName": "Mustermann",
        "email": "maxmuster@arvatotest.de",
        "birthDate": "1990-12-31",
        "customerCategory": "Person",
        "customerNumber": "123456",
        "mobilPhone": "+49 1234 567890",
        "address": {
            "street": "Gütersloher Str.",
            "streetNumber": "123",
            "postalCode": "33415",
            "postalPlace": "Verl",
            "countryCode": "DE"
        }
    },
    "deliveryCustomer": {
        "salutation": "Mr",
        "firstName": "Max",
        "lastName": "Mustermann",
        "email": "maxmuster@arvatotest.de",
        "birthDate": "1990-12-31",
        "customerCategory": "Person",
        "address": {
            "street": "Packstation",
            "streetNumber": "250",
            "careOf": "123456789",
            "posalCode": "33415",
            "postalPlace": "Verl",
            "countryCode": "DE"
        }
    },
    "order": {
        "number": "orderNoIndividual004",
        "totalGrossAmount": 20.0,
        "totalNetAmount": 16.40,
        "currency": "EUR",
        "items": [
            {
                "productId": "0001",
                "description": "Testproduct 1",
                "grossUnitPrice": 10.0,
                "netUnitPrice": 8.4,
                "quantity": 2,
                "GroupId": "21",
                "VatPecent": 19.0,
                "VatAmount": 1.6,
                "ImageUrl": "http://i.imgur.com/r9tKrfM.jpg"
            }
        ],
        "riskData": {
            "ProfileTrackingId": "12456789",
            "IpAddress": "127.0.0.1"
        }
    }
}
```

## Example - Response – Customer accepted


```
{
    "outcome": "Accepted",
    "customer": {
        "customerNumber": "118836897",
        "firstName": "Max",
        "lastName": "Mustermann",
        "addressList": [
            {
                "street": "Gütersloher Str.",
                "streetNumber": "123",
                "postalCode": "33415",
                "postalPlace": "Verl",
                "countryCode": "DE"
            }
        ]
    },
    "reservationId": "74179665-e06f-4697-c0a1896a836b",
    "checkoutId": "87ab98ff-d944-4b56-9522-9bc56335030",
    "expirationDate": "2020-09-26"
}
```


## Example - Response – Customer rejected


```
{
    "outcome": "Rejected",
    "customer": {
        "customerNumber": "118836897",
        "firstName": "Max",
        "lastName": "Mustermann",
        "addressList": [
            {
                "street": "Gütersloher Str.",
                "streetNumber": "123",
                "postalCode": "33415",
                "postalPlace": "Verl",
                "countryCode": "DE"
            }
        ]
    },
    "reservationId": "00000000-0000-0000-0000-000000000000",
    "checkoutId": "000000000-0000-0000-0000-000000000000",
    "riskCheckMessages": [
        {
            "type": "BusinessError",
            "code": "200.903",
            "message": "Customer Limit reached",
            "customerFacingMessage": "Die Zahlungsmethode ist leider derzeit nicht
verfügbar. Bitte eine andere Zahlungsmethode auswählen.",
            "actionCode": "OfferSecurePaymentMethods"
        }
    ]
}
```

## 2.2 Void (voidPayment)

If an item purchased by the customer cannot be delivered, the authorize cannot be captured. For this situation the merchant can cancel the authorize.

**Riverty:** /api/v3/orders/{orderNumber}/voids

| cancellationDetails |
|---|
| totalGrossAmount (maximum 2 numbers after the comma) |
| totalNetAmount (maximum 2 numbers after the comma) |
| items |
| productid |
| groupId |
| description |
| netUnitPrice (maximum 2 numbers after the comma) |
| grossUnitPrice (maximum 2 numbers after the comma) |
| quantity |
| VatPercent |
| VatAmount (maximum 2 numbers after the comma) |
| ImageUrl |
| ProductUrl |
| lineNumber |

| Void Response |
|---|
| totalCapturedAmount |
| totalAuthorizedAmount |
| remainingAuthorizedAmount |

## Example 1: Full-Void


```
{
    "cancellationDetails": {
        "totalGrossAmount": 79.98,
        "totalNetAmount": 67.21,
        "items": [
            {
                "productId": "0001",
                "groupId": "21",
                "description": "Testproduct 1",
                "netUnitPrice": 25.20,
                "grossUnitPrice": 29.99,
                "quantity": 1,
                "vatPercent": 19.00,
                "vatAmount": 4.79,
                "imageUrl": "http://i.imgur.com/r9tKrFm.jpg",
                "product_url": "http://i.imgur.com/r9tKrFm.jpg",
                "lineNumber": 1
            },
            {
                "productId": "0002",
                "groupId": "21",
                "description": "Testproduct 2",
                "netUnitPrice": 42.01,
                "grossUnitPrice": 49.99,
                "quantity": 1,
                "vatPercent": 19.00,
                "vatAmount": 7.98,
                "imageUrl": "http://i.imgur.com/r9tKrFm.jpg",
                "product_url": "http://i.imgur.com/r9tKrFm.jpg",
                "lineNumber": 2
            }
        ]
    }
}
```


## Example 1: Void Response (Full-Void)


```
{
    "totalCapturedAmount": 0.00,
    "totalAuthorizedAmount": 79.98,
    "remainingAuthorizedAmount": 0.00
}
```

## Example 2: Partial-Void


```
{
    "cancellationDetails": {
        "totalGrossAmount": 29.99,
        "totalNetAmount": 25.20,
        "items": [
            {
                "productId": "0001",
                "groupId": "21",
                "description": "Testproduct 1",
                "netUnitPrice": 25.20,
                "grossUnitPrice": 29.99,
                "quantity": 1,
                "vatPercent": 19.00,
                "vatAmount": 4.79,
                "imageUrl": "http://i.imgur.com/r9tKrFm.jpg",
                "product_url": "http://i.imgur.com/r9tKrFm.jpg",
                "lineNumber": 1
            }
        ]
    }
}
```


## Example 2: Void Response (Partial-Void)


```
{
    "totalCapturedAmount": 49.99,
    "totalAuthorizedAmount": 79.98,
    "remainingAuthorizedAmount": 29.99
}
```

## 2.3 Capture (capturePayment)

Completes the payment that has been authorized before. Typically done when the order is shipped. It can be a full or partial capture of the order amount. The customer receives one invoice per capture and the payment period begins.

**Riverty:** /api/v3/orders/{orderNumber}/captures

| orderDetails |
|---|
| totalGrossAmount (maximum 2 numbers after the comma)<br>totalNetAmount (maximum 2 numbers after the comma)<br>currency<br>items<br>productId<br>description<br>grossUnitPrice (maximum 2 numbers after the comma)<br>netUnitPrice (maximum 2 numbers after the comma)<br>quantity<br>VatPercent<br>VatAmount (maximum 2 numbers after the comma)<br>GroupId<br>ImageUrl<br>LineNumber |

invoiceNumber -> used as captureNumber if given (generated by Riverty if not)

| shippingDetails |
|---|
| type (Shipment, Return)<br>shippingCompany<br>trackingId |

| CaptureResponse |
|---|
| capturedAmount |
| authorizedAmount |
| remainingAuthorizedAmount |
| captureNumber |

## Example 1: Full-Capture

Previously the following articles have been submitted to our API – OrderNo 88371221


```
{
    "orderDetails": {
        "totalGrossAmount": 23.95,
        "totalNetAmount": 20.12,
        "currency": "EUR",
        "items": [
            {
                "productId": "0001",
                "description": "Testproduct 1",
                "grossUnitPrice": 10.0,
                "netUnitPrice": 8.4,
                "quantity": 2,
                "GroupId": "21",
                "VatPercent": 19.0,
                "VatAmount": 1.6,
                "ImageUrl": "http://i.imgur.com/r9tKrfM.jpg",
                "lineNumber": 1
            },
            {
                "productId": "shippingFee",
                "description": "Shipping Fee",
                "grossUnitPrice": 3.95,
                "netUnitPrice": 3.32,
                "quantity": 1.0,
                "VatPercent": 19.0,
                "VatAmount": 0.63
            }
        ]
    },
    "invoiceNumber": "800871770"
}
```


## Example 1: Capture Response (Full-Capture)


```
{
    "capturedAmount": 23.95,
    "authorizedAmount": 23.95,
    "remainingAuhtorizedAmount": 0.00,
    "captureNumber": "800871770"
}
```

## Example 2: Partial-Capture

```
{
    "orderDetails": {
        "totalGrossAmount": 49.99,
        "totalNetAmount": 42.01,
        "currency": "EUR",
        "items": [
            {
                "productId": "0002",
                "description": "Testproduct 2",
                "grossUnitPrice": 49.9,
                "netUnitPrice": 42.01,
                "quantity": 1,
                "VatPercent": 19.0,
                "VatAmount": 7.98,
                "ImageUrl": "http://i.imgur.com/r9tKrfM.jpg",
                "lineNumber": 1
            }
        ]
    },
    "invoiceNumber": "800871770"
}
```

## Example 2: Capture Response (Partial-Capture)

```
{
    "capturedAmount": 49.99,
    "authorizedAmount": 79.98,
    "remainingAuhtorizedAmount": 29.99,
    "captureNumber": "800871770"
}
```

→ The response of the full capture is identical, only the remainingAuthorizedAmount value is zero
→ captureNumber is required for returns or subsequent credit notes

## 2.4 Refund (refundPayment)

If a shopper returns an order or the merchant wants to compensate the shopper for an order related issue. A refund can be either for the full payment or for a part of the payment. A partial refund is typically issued by order line items and amount, but you do not have to stick to the structure of the capture passed before as long as the original invoice amount is not exceeded.

**Request**: /api/v3/orders/{orderNumber}/refunds

| | |
|---|---|
| captureNumber | |
| creditNoteNumber → | used as refundNumber if given (generated by Riverty if not) |
| Items | |
| | productid |
| | groupId |
| | description |
| | grossUnitPrice (maximum 2 numbers after the comma) |
| | netUnitPrice (maximum 2 numbers after the comma) |
| | quantity |
| | vatPercent |
| | vatAmount (maximum 2 numbers after the comma) |
| | imageUrl |
| | productUrl |
| | lineNumber |

| RefundResponse |
|---|
| totalCaptureNumber |
| totalAuthorizedAmount |
| refundNumbers<br>array of refundNumbers |
| totalRefundedAmount |

## Example 1: Full-Refund

Previously the following articles have been captured –OrderNo 88371221

```
"items": [
    {
        "productId": "0001",
        "descrption": "Testproduct 1",
        "grossUnitPrice": 29.99,
        "netUnitPrice": 25.20,
        "quantity": 1,
        "GroupId": "21"
        "VatPercent": 19.0,
        "VatAmount": 4.79,
        "ImageUrl": "http://i.mgur.com/r9tKrFm.jpg",
        "ProductUrl": "http://i.mgur.com/r9tKrFm.jpg",
        "lineNumber": 1
    },
    {
        "productId": "0002",
        "descrption": "Testproduct 2",
        "grossUnitPrice": 49.99,
        "netUnitPrice": 42.01,
        "quantity": 1,
        "VatPercent": 19.0,
        "VatAmount": 7.98,
        "ImageUrl": "http://i.mgur.com/r9tKrFm.jpg"
    }
]
```

## Example 1: Refund Response (Full-Refund)

```
{
    "totalCapturedAmount": 49.99,
    "totalAuthorizedAmount": 79.98,
    "refundNumbers": [
        "800871770"
    ],
    "totalRefundedAmount": 49.99
}
```

## Example 2: Partial-Refund

Previously the following articles have been captured –OrderNo 88371222

```
{
    "captureNumber": "800871770", //captureNumber from capture response
    "orderitems": [
        {
            "productId": "0002",
            "description": "Testproduct 2",
            "grossUnitPrice": 49.99,
            "netUnitPrice": 42.01,
            "quantity": 1,
            "GroupId": "21",
            "VatPercent": 19.0,
            "VatAmount": 7.98,
            "ImageUrl": "http://i.imgur.com/r9tKrfM.jpg",
            "ProductUrl": "http://i.imgur.com/r9tKrfM.jpg",
            "lineNumber": 1
        }
    ]
}
```

## Example 2: Refund Response (Partial-Refund)

```
{
    "totalCapturedAmount": 49.99,
    "totalAuthorizedAmount": 79.98,
    "refundNumbers": [
        "800871770"
    ],
    "totalRefundedAmount": 49.99
}
```

## Example 3: Subsequent discount (no return)

We have no restrictions for returns. You can use any reason for a discount. Whether it is a subsequent discount, a refund of shipping costs or anything else -as long as the original invoice amount is not exceeded.

```
{
    "captureNumber": "{{CAPTURE_NUMBER}}", //captureNumber from response
    "orderItems": [
        {
            "productId": "discount",
            "description": "Subsequent discount (10€)",
            "grossUnitPrice": 10.00,
            "netUnitPrice": 8.40,
            "quantity": 1,
            "VatPErcent": 19.0,
            "VatAmount": 1.60
        }
    ]
}
```

## Response (Partial-Request)

```
{
    "totalCapturedAmount": 49.99,
    "totalAuthorizedAmount": 79.98,
    "refundNumbers": [
        "800871771"
    ],
    "totalRefundedAmount": 49.99
}
```

# 3 Riverty Error Handling

To ensure that no errors occur when the data is transferred to us, our API includes an error-feedback system. If we receive a call incorrectly, our system reacts and returns an error message with the appropriate action code. Using these codes, the customer can be asked to re-enter a value instead of having to reject the order. We distinguish our error messages between validation errors (e.g. wrong birthdate) and business errors (e.g., Customer limit reached).

| Code | Description | Possible action code |
|---|---|---|
| 400.000 | Unknown error | Unavailable |
| 400.001 | Request body missing | Unavailable |
| 400.002 | Value is required | Unavailable / AskConsumerToReEnterData |
| 400.003 | Value out of range | Unavailable / AskConsumerToReEnterData |
| 400.004 | Value format is incorrect | Unavailable / AskConsumerToReEnterData |
| 400.005 | A string value exceeds maximum length of {value} | Unavailable / AskConsumerToReEnterData |
| 400.006 | A string value with maximum length {value} is required | Unavailable / AskConsumerToReEnterData |
| 400.007 | A decimal value has more than allowed decimal digits | Unavailable / AskConsumerToReEnterData |

### Example authorize response – 400.004


```
[
    {
        "type": "BusinessError",
        "code": "400.004",
        "message": "value format is incorrect",
        "customerFacingMessage": "Value format is incorrect",
        "actionCode": "AskConsumerToReEnterData",
        "fieldReference": "customer.birthDate"
    }
]
```

Example authorize response – 400.005


```
[
    {
        "type": "BusinessError",
        "code": "400.005",
        "message": "A string value exceeds maximum length of 10. ",
        "customerFacingMessage": "A string value exceeds maximum length of 10. ",
        "actionCode": "AskConsumerToReEnterData",
        "fieldReference": "customer.address.streetNumberAdditional"
    }
]
```


## 3.1 Customer facing error messages to API

These are the human-readable, descriptive messages returned by the eCOM API for common business errors. They are usually due to missing mandatory information, or information provided in the incorrect format. When integrating Riverty in your checkout page, please make sure the contents of these messages are displayed to the customer. We provide the description with language code and full text in our response. All languages and errors are shown on developer.riverty.com.