Request Refund Approval

Retrieve Product ID from Get Order Details API using reference. In the message field you need to pass the message why you can not complete the refund via regular refund flow, better to add the exact message received while trying to refund with the regular endpoint.

Endpoint

```
POST https://demo-api.frontpayment.no/api/v1/orders/refund/request/approval/{{ORDER_UUID}}
```

Authentication

Include a **Bearer Token** in the `Authorization` header. You can obtain this token from **Front Payment**.

Example:

```
Authorization: Bearer YOUR_FRONTPAYMENT_BEARER_TOKEN
```

Request Payload

Send the following parameters as a JSON object in the request body:

```
{
   "type": "reservation",
   "grandTotal": 10,
   "products": [
      {
         "id": 540,
         "amount": 10
      }
   ],
   "message": "refundRejectionForWeeklyThresholdExceed",
   "source": "charged",
```

```
    "reference": "CHA3852658817",
}
```

Validation Rules

Make sure your request meets the following requirements:

Field	Type	Description
`type`	`string`	`Required` Using type. Available types are `regular`, `invoiced` and `reservation`
`grandTotal`	`numeric`	`Required` Grand total of the refunded amount.
`products.*id`	`numeric`	**Required** Order product id. From which product you want to refund.
`products.*.amount`	`numeric`	**Required** Refund amount for the product.
`message`	`string`	**Required** Refund rejection message.
`source`	`string`	**Conditional Required** This field is **required** when the type is **reservation**. Available sources are `captured` and `charged`
`reference`	`string`	**Conditional Required** This field is **required** when the type is **reservation**. If `source` is `captured`, use the `uuid` from the captured payment. If `source` is `charged`, use the `uuid` from the charged payment.

Response

A successful request will return a `201 OK` status with the following JSON payload:

```
{
    "status_code": 201,
    "status_message": "OK",
    "message": "requestedOrderRefundSuccessfully",
    "is_data": false,
    "data": null
}
```

Error Response

API returns a `404` error, it means requested order with `ORDER_UUID` could not be found in our system.


```
{
   "status_code": 404,
   "status_message": "Not Found",
   "message": "orderNotFound",
   "is_error": false,
   "errors": null
}
```


API return a `417` error, it means request payload validation failed.


```
{
   "status_code": 417,
   "status_message": "Client Error",
   "message": "payloadValidationErrors",
   "is_error": true,
   "errors": "Array"
}
```


API returns a `510` error, it means something failed on the server side


```
{
   "status_code": 510,
   "status_message": "Execution Exception Occurred",
   "message": "somethingWentWrong",
   "is_error": true,
   "errors": "Array"
}
```


Others refund rejections errors


```
{
   "status_code": 400,
   "status_message": "Conflict of Business Logic",
   "message": "requestProductIdNotAvailable",
```

```
    "is_data": false,
    "data": null
}
```

```
{
    "status_code": 400,
    "status_message": "Conflict of Business Logic",
    "message": "orderRefundRequestAlreadySubmitted",
    "is_data": false,
    "data": null
}
```

```
{
    "status_code": 400,
    "status_message": "Conflict of Business Logic",
    "message": "refundRejectionForRefundedCancelledInvoicedOrderParamRefunded",
    "is_data": false,
    "data": null
}
```

```
{
    "status_code": 400,
    "status_message": "Conflict of Business Logic",
    "message": "refundRejectionForRefundedCancelledInvoicedOrderParamCancelled",
    "is_data": false,
    "data": null
}
```


Revision #8
Created 29 January 2024 10:41:49 by Admin
Updated 8 September 2025 10:31:49 by Admin