Reservation Management

- Submit Reservation
- Create Session for Reservation
- Get Reservation Details By UUID
- Capture Reservation
- Charge Reservation
- Cancel Reservation
- Complete Reservation
- Resend Reservation
- Refund Reservation
- Get Reservation History

Submit Reservation

The **Submit Reservation API** allows merchants to create a reservation for a customer and generate a payment link to be sent to the customer. Once created, the customer can complete payment with Visa and Mastercard debit or credit cards.

This API is designed for seamless integration into booking, order management, and e-commerce workflows where you want to lock in an order before payment is finalized.

Typical Flow

1. Submit a reservation request with customer and order details.
2. Send payment link from FrontGO with order and merchant details.
3. Redirect the customer to the provided checkout URL.
4. Receive status updates (e.g., `Reserved`, `Captured`, `Charged`) via your callback endpoint.

Prerequisites

Before integrating the **Submit Reservation API**, ensure you have the following:

- **API Access**: A valid merchant account with Front Payment and access to the API environment (Demo or Production).
- **Authentication**: A Bearer token issued by Front Payment, included in the `Authorization` header of every request.
- **Callback Endpoint**: A publicly accessible HTTPS endpoint to receive real-time payment status notifications (recommended).
- **Basic Setup**:
  - Ability to send HTTPS `POST` requests with JSON payloads.
  - Correct handling of numeric totals and validation rules (e.g., customer type, tax fields, 11-digit personal numbers).
  - Server time synchronized to ensure accurate timestamps for order and payment link expiry.
- **Testing Environment** : Access to the demo API from the following URL to validate your integration end to end before going live.

  ```
  https://demo-api.frontpayment.no
  ```

Reservation Lifecycle

Once a reservation is successfully created and its status is updated to **Reserved**, you have several options for handling the reserved funds:

- **Capture**: Capture the reserved amount either in full or partially. Capturing is typically used to secure payment when goods or services are delivered. The reservation period varies depending on your business type but generally lasts between **7 and 31 days**. See Capture API Reference
- **Partial Capture and Release**: If you capture only part of the reserved amount, the remaining balance is automatically released back to the customer within **1–3 days**.
- **Charge**: It is also possible to charge the customer up to **90 days** after the reservation has been confirmed. After this period, there is **no guarantee** that the reserved funds will still be available for charging. See Charge API Reference

---

These rules provide flexibility in aligning payment capture with your operational processes, while ensuring that customers are not left with indefinite reservations on their accounts.

Submit Reservation

Endpoint

POST https://demo-api.frontpayment.no/api/v1/connect/reservations/submit

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

```
{
  "customerDetails": {
```

```
  "uuid": "",
  "type": "private",
  "countryCode": "+880",
  "msisdn": "1724313009",
  "email": "nafees@yopmail.com",
  "name": "Nafees",
  "preferredLanguage": "en",
  "personalNumber": "19635464524",
  "organizationId": null,
  "address": {
    "street": "Dhaka",
    "zip": "3500",
    "city": "Cumilla",
    "country": "NO"
  }
},
"orderDate": "1756871295",
"dueDateForPaymentLink": "1756871295",
"referenceNo": null,
"customerReference": null,
"sendOrderBy": {
  "sms": false,
  "email": true
},
"products": {
  "0": {
    "name": "Test",
    "productId": null,
    "rate": 1,
    "tax": "0",
    "amount": 1
  }
},
"orderSummary": {
  "subTotal": 1,
  "totalTax": 0,
  "grandTotal": 1
},
"chargeValidity": "55",
"customerNotes": null,
```

```
  "tnc": null,
  "settings": {
    "secureDetails": false,
    "isChargePartiallyRefundable": false
  },
  "callback": {
    "callbackUrl": "https://wp.frontpayment.no"
  }
}
```


Validation Rules

Field	Type	Description
`customerDetails`	`array`	**Required**. Customer details information.
`customerDetails.type`	`string`	**Required**. Customer type (private or corporate).
`customerDetails.countryCode`	`string`	**Required**. Country dialing code (e.g., +880).
`customerDetails.msisdn`	`string`	**Required**. Mobile Subscriber ISDN Number (phone number).
`customerDetails.email`	`email`	**Required**. Valid customer email address.
`customerDetails.name`	`string`	**Required**. Full name of the customer.
`customerDetails.preferredLanguage`	`string`	**Required**. Preferred language (e.g., en, no).
`customerDetails.personalNumber`	`string`	**Optional**. Customer's personal identification number, must be 11 characters.
`customerDetails.organizationId`	`string`	**Required** if `customerDetails.type` is `corporate`. Must be number
`customerDetails.address`	`array`	**Required**. Customer address details.
`customerDetails.address.street`	`string`	**Required**. Street address of the customer.
`customerDetails.address.zip`	`string`	**Required**. Zip code of the customer's address.
`customerDetails.address.city`	`string`	**Required**. City of the customer's address..
`customerDetails.address.country`	`string`	**Required**. ISO Alpha-2 country code (e.g., "NO"). Custom validation IsoAlpha2Country applies.
`orderDate`	`string`	**Required**. Unix timestamp for the Date of the order.
`dueDateForPaymentLink`	`string`	**Required**.Unix timestamp for payment link expiry. Must be current or future timestamp.
`referenceNo`	`string`	**Optional**. Internal reference number.
`customerReference`	`string`	**Optional**. Customer’s reference number.

Field	Type	Description
`sendOrderBy`	`array`	**Required**. Defines communication channels (SMS, email).
`sendOrderBy.sms`	`boolean`	**Required**. Whether to send via SMS.
`sendOrderBy.email`	`boolean`	**Required**. Whether to send via Email.
`products`	`array`	**Required**. List of products in the order.
`products.*.name`	`string`	**Required**. Name of the product.
`products.*.productId`	`string`	**Optional**. Unique identifier for the product.
`products.*.rate`	`numeric`	**Required**. Rate per unit of the product.
`products.*.tax`	`numeric`	**Required**. Tax rate (e.g., 0, 12, 15, 25). Unless you have other configuration.
`products.*.amount`	`numeric`	**Required**. Quantity of product.
`orderSummary.subTotal`	`numeric`	**Required**. Subtotal of products.
`orderSummary.totalTax`	`numeric`	**Required**. Total tax amount.
`orderSummary.grandTotal`	`numeric`	**Required**. Final payable amount.
`customerNotes`	`string`	**Optional**. Customer-provided notes.
`tnc`	`string`	**Optional**. Terms & Conditions.
`chargeValidity`	`string`	**Optional**. Must be numeric string.
`settings`	`array`	**Optional**. Additional settings.
`settings.secureDetails`	`boolean`	**Optional**.If `secureDetails` is set to `True`, the reservation details will not be accessible until the user’s PNumber has been verified through BankID.
`settings.isChargePartiallyRefundable`	`boolean`	**Optional**. Whether partial refunds are allowed.
`callback`	`array`	**Optional**. Callback configuration.
`callback.callbackUrl`	`url`	**Optional**. If the value is true charge will be party refundable, otherwise it will be fully refundable.

Example Success Response


```
{
  "status_code": 201,
  "status_message": "OK",
  "message": "Reservation Submitted Successfully",
  "is_data": true,
  "data": {
    "customerUuid": "CSRT1358046874",
    "reservationUuid": "RES1943140879",
```

```
    "checkoutUrl": "https://stg.frontpayment.no/reservations/details/RES1943140879"
  }
}
```


Error Response

401 Unauthorized: Missing or invalid Bearer token. Other validation errors will return appropriate HTTP error codes (e.g., 400 Bad Request) along with error messages specifying the invalid or missing fields.


```
{
    "status_code": 500,
    "status_message": "Internal Dependency Error",
    "message": "internalErrorOccurredPleaseTryAgainLater",
    "is_error": true,
    "errors": {
        "happenedAt": "String",
        "internalErrorDetails": "Array"
    }
}
```

```
{
  "status_code": 510,
  "status_message": "Internal Dependency Error",
  "message": "Payment Gateway Error(Submit Payment): Failed to Create Payment Link",
  "is_error": true,
  "errors": {
    "Payment Gateway Error(Submit Payment): Failed to Create Payment Link"
  }
}
```


Redirect to Checkout

After receiving a successful response, the customer should be redirected to the `checkoutUrl` provided. At checkout, the customer can choose from supported payment methods:

- **Visa**
- **Mastercard**

Once the payment is completed successfully, the user will be redirected to a success page.

Callback Notification

The `callbackUrl` is an endpoint on your server that our system will call via an `HTTP GET` request whenever the status of the specified order changes from its initial state. For `reservation` the `callbackUrl` will be triggered for three status changed: `Reserved`, `Captured`, `Charged`.

See the link below to understand how to work with the callback URL on your side and how to verify the request sent from our side.

Go To `Notication Via Callback Url` Page

Create Session for Reservation

The **Create Session for Reservation** endpoint enables you to export a pre-created reservation into **FrontGo** and obtain a checkout URL in a single call.

This combines reservation setup with an immediate payment session, giving the customer a seamless experience for completing the payment.

This approach is particularly useful when you already have reservation data (or have just sent a `submit reservation` request) and want to let the user pay right away without multiple round trips. The returned session can preselect a payment method (e.g., Visa, Mastercard) but still allows users to switch if needed.

Typical Flow

1. Call **Create Session for Reservation** with reservation and payment parameters.
2. Receive a `paymentUrl` and session identifiers.
3. Redirect the user to the `paymentUrl` to complete payment.
4. Receive status updates via callback as the reservation transitions through states (`Reserved`, `Captured`, `Charged`).

Prerequisites

Before integrating **Create Session for Reservation**, make sure you have the following in place:

- **Existing Reservation Data**: The endpoint assumes you have a reservation context (or that you are bundling creation & session setup).
- **API Access & Credentials**: A valid merchant account with Front Payment and access to an API environment (demo or production).
- **Bearer Token Authentication**: Include a valid Bearer token in the `Authorization` header for every request.
- **Callback Endpoint(s)**: A publicly accessible HTTPS endpoint to receive notifications (via `GET`) when status changes for the session/reservation.
- **HTTPS & JSON Support**: Your server or application must be able to send `HTTP POST` requests with JSON payloads and parse JSON responses.

- **Timestamp & Validity Control**: Ability to compute or provide valid timestamps (e.g., `dueDateForPaymentLink`) to manage how long the session remains active.
- **Testing Environment** : Access to the demo API from the following URL to validate your integration end to end before going live.

  ```
  https://demo-api.frontpayment.no
  ```

Step 1: Submit Reservation

Endpoint:

```
POST https://demo-api.frontpayment.no/api/v1/connect/reservations/create
```

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 Example

```
{
 "customerDetails": {
  "type": "private",
  "countryCode": "+47",
  "msisdn": "46567468",
  "email": "nafees@yopmail.com",
  "name": "Nafees",
  "preferredLanguage": "en",
  "personalNumber": null,
  "organizationId": null,
  "address": {
   "street": "Dhaka",
   "zip": "3500",
   "city": "Cumilla",
   "country": "NO"
  }
 },
```

```
  "orderDate": "1724294524",
  "dueDateForPaymentLink": "1724294524",
  "referenceNo": null,
  "customerReference": null,
  "sendOrderBy": {
    "sms": false,
    "email": false
  },
  "products": {
    "0": {
      "name": "Test",
      "productId": null,
      "rate": 1000,
      "tax": "0",
      "amount": 1000
    }
  },
  "orderSummary": {
    "subTotal": 1000.00,
    "totalTax": 0,
    "grandTotal": 1000.00
  },
  "chargeValidity": "55",
  "customerNotes": null,
  "tnc": null,
  "submitPayment": {
    "via": "visa"
  },
  "callback": {
    "callbackUrl": "https://wp.frontpayment.no/?order_identifier=rRbl1FWZG59o&order_status=success",
    "success": "https://wp.frontpayment.no/?order_identifier=rRbl1FWZG59o&order_status=success",
    "failure": "https://frontpayment.no/?order_identifier=rRbl1FWZG59o&order_status=failed"
  },
  "settings": {
    "secureDetails": false,
    "isChargePartiallyRefundable": true
  }
}
```

Validation Rules

Field	Type	Description
customerDetails.type	string	**Required**. Customer type (private or corporate).
customerDetails.countryCode	string	**Required**. Country code for the customer's phone number (e.g., "+47").
customerDetails.msisdn	string	**Required**. Mobile phone number without country code.
customerDetails.email	email	**Required**. Valid customer email address.
customerDetails.name	string	**Required**. Full name of the customer.
customerDetails.preferredLanguage	string	**Required**. Customer preferred language. Available languages are en, no, sv, da, de. If nothing is given it will set default to no.
customerDetails.personalNumber	string	**Optional**. Customer's personal identification number, must be exactly 11 characters containing only numbers and cannot contain spaces. When Customer type is private then you can used this for add personal number.
customerDetails.organizationId	string	**Required** Organization identification number, must contain only numbers and cannot contain spaces. When Customer type is corporate then this field is required. Otherwise you can add this as null or remove from payload.
customerDetails.address	array	**Required**. Customer address details.
customerDetails.address.street	string	**Required**. Street name.
customerDetails.address.zip	string	**Required**. Postal code.
customerDetails.address.city	string	**Required**. City name.
customerDetails.address.country	string	**Required**. ISO Alpha-2 country code (e.g., NO). Custom validation IsoAlpha2Country applies..
orderDate	string	**Required**.Unix timestamp for the Date of the order.
dueDateForPaymentLink	string	**Required**.Provide Current / Future Date as Unix timestamp for the Due Date of the order.
referenceNo	string	**Optional**. Reference number.
customerReference	string	**Optional**. Customer reference ID.
sendOrderBy	array	**Required**. Notification preferences.
sendOrderBy.sms	boolean	**Required**. Whether to send order by SMS.
sendOrderBy.email	boolean	**Required**. Whether to send order by email.
products	array	**Required**. List of product items.
products.*.name	string	**Required**. Name of the product..

Field	Type	Description
products.*.productId	string	**Optional**. Unique identifier for the product.
products.*.rate	numeric	**Required**. Rate per unit of the product.
products.*.tax	numeric	**Required**. Tax rate must be (e.g., 0, 12, 15, 25), Unless you have other configuration.
products.*.amount	numeric	**Required**. Total product amount.
orderSummary.subTotal	numeric	**Required**. Subtotal of all products before tax and discount.
orderSummary.totalTax	numeric	**Required**. Total tax for the order.
orderSummary.grandTotal	numeric	**Required**. Grand total of the order.
customerNotes	string	**Optional**. Notes from customer.
tnc	string	**Optional**. Terms and conditions.
chargeValidity	string	**Optional**. Validity in minutes (digits only).
submitPayment	array	**Required**. Payment submission details.
submitPayment.via	string	**Required**. Payment method (visa , mastercard).
callback	array	**Optional**. Callback URLs.
callback.callbackUrl	url	**Optional**. General callback URL.
callback.success	url	**Optional**. Success redirect URL.
callback.failure	url	**Optional**. Failure redirect URL.
settings	array	**Optional**. Additional settings.
settings.secureDetails	boolean	**Optional** If secureDetails is True , order details wouldn't be visible without verifying BankID.
settings.isChargePartiallyRefundable	boolean	**Optional**. If the value is true charge will be party refundable, otherwise it will be fully refundable.

Example Success Response


```
{
  "status_code": 201,
  "status_message": "OK",
  "message": "Reservation Submitted Successfully",
  "is_data": true,
  "data": {
    "customerUuid": "CSRT3463048878",
    "reservationUuid": "RES4161996022",
    "paymentUrl": "https://v1.checkout.bambora.com/aa7ec3f47b0d45b286bcc595ab0d9613"
```

```
  }
}
```

Error Response

401 Unauthorized: Missing or invalid Bearer token. Other validation errors will return appropriate HTTP error codes (e.g., 400 Bad Request) along with error messages specifying the invalid or missing fields.

```
{
   "status_code": 500,
   "status_message": "Internal Dependency Error",
   "message": "internalErrorOccurredPleaseTryAgainLater",
   "is_error": true,
   "errors": {
      "happenedAt": "String",
      "internalErrorDetails": "Array"
   }
}
```

```
{
   "status_code": 510,
   "status_message": "Internal Dependency Error",
   "message": "Payment Gateway Error: Failed to Create Checkout Session",
   "is_error": true,
   "errors": [
      "Payment Gateway Error(Submit Payment): Failed to Create Checkout Session At: PAR156"
   ]
}
```

Step 2: Payment Process

- From the success response in **Step 1**, the user is redirected to the `paymentUrl`.
- The preselected payment method (`visa` or `mastercard`) will be shown, but the user can change it.
- After successful payment:
  - The third-party system is notified via the provided `callbackUrl`.
  - The user is redirected to the `success` or `failure` URL provided in the request payload.

Notifications via Callback URL

The `callbackUrl` is an endpoint on your server that our system will call via an `HTTP GET` request whenever the status of the specified order changes from its initial state. For `reservation` the `callbackUrl` will be triggered for three status changed: `Reserved`, `Captured`, `Charged`.

See the link below to understand how to work with the callback URL on your side and how to verify the request sent from our side.

Go To `Notication Via Callback Url` Page

Get Reservation Details By UUID

The **Get Reservation Details By UUID** endpoint allows your application to fetch comprehensive information about a specific reservation by supplying its unique identifier (UUID). This API is part of the *Reservation Management* module in the FrontGO and is intended for retrieving detailed data such as customer details, reserved items, payment history, and more.

You'll use this endpoint when you need to:

- Verify the status of a reservation (pending, captured, canceled, etc.)
- Access the list of reserved products, with rates, discounts, tax, quantities, etc.
- Inspect payment details including amounts reserved, captured, refunded, and transaction history
- Review customer and organization metadata tied to the reservation

You will find the summary of how the endpoint works, its authentication scheme, and typical responses below.

Endpoint

```
GET https://demo-api.frontpayment.no/api/v1/connect/reservations/details/{{RESERVATION_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
```

Response

A successful request will return a `200 OK` status with the following JSON payload:

```
{
  "status_code": 200,
  "status_message": "OK",
  "message": "reservationRetrievedSuccessfully",
  "is_data": true,
  "data": {
    "reservationUuid": "String",
    "status": "String",
    "isPaid": "Boolean",
    "productList": [
      {
        "id": "Integer",
        "name": "String",
        "productId": "String",
        "quantity": "Float",
        "rate": "Float",
        "discount": "Float",
        "tax": "Integer",
        "amount": "Float",
        "reserved": "Float",
        "captured": "Float"
      }
    ],
    "grandTotal": "Float",
    "reservationDate": "String",
    "paymentLinkDueDate": "String",
    "sendOrderBy": {
      "sms": "Boolean",
      "email": "Boolean"
    },
    "customerDetails": {
      "countryCode": "String",
      "msisdn": "String",
      "email": "String",
      "name": "String",
      "address": {
        "street": "String",
        "zip": "String",
        "city": "String",
        "country": "String"
```

```
    }
},
"referenceNumber": "Nullable|String",
"chargeValidity": "Nullable|String",
"customerReference": "Nullable|String",
"customerNotes": "Nullable|String",
"termsAndCondition": "Nullable|String",
"paymentDetails": {
    "reservedAt": "Timestamp",
    "reservedAmount": "Float",
    "capturedAmount": "Float",
    "chargedAmount": "Float",
    "amountRefunded": {
        "fromCaptured": "Float",
        "fromCharge": "Float"
    }
},
"organizationDetails": {
    "name": "String",
    "billingAddress": {
        "countryCode": "String",
        "msisdn": "String",
        "email": "String",
        "street": "String",
        "zip": "String",
        "city": "String",
        "country": "String"
    }
},
"translationKey": "String",
"paymentHistory": {
    "reserved": [
        {
            "at": "String|Timestamp",
            "amount": "Float"
        }
    ],
    "captured": [
        {
            "at": "String|Timestamp",
```

```
            "amount": "Float",
            "reference": "String",
            "isRefunded": "Boolean",
            "refunded": "Float",
            "additionalText": "String"
        }
    ],
    "charged": [
        {
            "at": "String|Timestamp",
            "amount": "Float",
            "reference": "String",
            "isRefunded": "Boolean",
            "refunded": "Float",
            "additionalText": "String"
        }
    ],
    "refunded": [
        {
            "at": "String|Timestamp",
            "amount": "Float",
            "reference": "String",
            "isRefunded": "Boolean",
            "isPartial": "Boolean"
        }
    ]
  }
 }
}
```


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

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


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

Capture Reservation

The **Capture Reservation** endpoint is used to convert a previously reserved (authorized) amount into an actual charge — either in full or partially — by referencing the reservation's unique identifier (UUID). Capturing is typically performed when goods or services are delivered, ensuring that the funds are secured from the customer's account.

Key behaviors

- **Full or Partial Capture**: You may capture the entire reserved amount, or you may choose to only capture part of it.
- **Automatic Release**: If only part of the reserved amount is captured, the remainder is automatically released back to the customer's account within 1–3 days.
- **Reservation Validity Window**: The original reservation remains valid for a limited period (often between 7 and 31 days, depending on your business rules). After that, the reservation may expire and no longer be capturable.

Use this endpoint when you're ready to finalize payment for what was reserved (or portions thereof), once delivery or service fulfillment is confirmed.

You will find details about the method, authentication, request fields, validation rules, and possible responses below.

Endpoint

```
POST https://demo-api.frontpayment.no/api/v1/connect/reservations/capture/{{RESERVATION_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:

```
{
   "products": {
      "0": {
         "id": 298,
         "amount": 5
      },
      "1": {
         "id": 299,
         "amount": 50
      }
   },
   "grandTotal": 55,
   "additionalText" : "My additional Text for capture"
}
```


Retrieve Product ID from Get Reservation API

Validation Rules

Make sure your request meets the following requirements:

Field	Type	Description
`products.*.id`	`numeric`	**Required** Reservation product id. From which product you want to captured
`products.*.amount`	`numeric`	**Required** Captured amount for the product
`grandTotal`	`numeric`	**Required** Grand total of the captured amount.
`additionalText`	`string`	**Optional** Captured note.

Response

A successful request will return a `202 OK` status with the following JSON payload:


```
{
   "status_code": 202,
   "status_message": "OK",
   "message": "reservationCapturedSuccessfully",
```

```
    "is_data": true,
    "data": {
      "uuid": "String"
    }
}
```


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


```
{
    "status_code": 404,
    "status_message": "Not Found",
    "message": "reservationNotFound",
    "is_data": false,
    "data": 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": "Something Went Wrong",
    "is_error": true,
    "errors": "Array"
}
```


Other Rejection Errors

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

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

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

Charge Reservation

The **Charge Reservation** endpoint enables you to initiate a merchant-initiated payment transaction **outside** of the originally reserved amount, using the customer's card tokenization data. In contrast to a *capture*, which merely converts a reserved authorization into a charge, a *charge* can be invoked independently — even after the reservation window — subject to certain limits and conditions.

Use this endpoint when:

- You wish to charge a customer after services have been delivered, or at a later time, beyond the original reservation window.
- You have a valid card token associated with the customer (resulting from tokenization during the reservation process) and want to debit their card directly.
- You want flexibility in timing: the charge may be performed up to **90 days** after reservation confirmation (or within a period specified by the merchant when creating the reservation). After this timeframe, the availability of the reserved funds can no longer be guaranteed.

> **Distinction from "Capture"**
>
> - A *capture* operation draws from the previously reserved authorization.
> - A *charge*, however, is independent and can exceed (or be separate from) the reserved amount, as long as valid payment credentials exist.

You will find details about endpoint usage, authentication, request schema, validation rules, and standard responses below.

Endpoint

```
POST https://demo-api.frontpayment.no/api/v1/connect/reservations/charge/{{RESERVATION_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:

```
{
   "products": {
      "0": {
         "name": "Charge QA",
         "productId": null,
         "rate": 150,
         "tax": "0",
         "amount": 150
      }
   },
   "grandTotal": 150,
   "additionalText" : "My additional Text for capture"
}
```

Validation Rules

Make sure your request meets the following requirements:

Field	Type	Description
`products.*.name`	`string`	**Required** Name of the product.
`products.*.productId`	`string`	**Optional** Unique identifier for the product.
`products.*rate`	`numeric`	**Required** Rate per unit of the product.
`products.*.tax`	`numeric`	**Required** Tax rate must be (e.g., `0`, `12`, `15`, `25`), Unless you have other configuration.
`products.*.amount`	`numeric`	**Required** Total amount for the product line item.
`grandTotal`	`numeric`	**Required** Grand total of the captured amount.
`additionalText`	`string`	**Optional** Captured note.

Response

A successful request will return a `202 OK` status with the following JSON payload:


```
{
   "status_code": 202,
   "status_message": "OK",
   "message": "reservationChargedSuccessfully",
   "is_data": true,
   "data": {
     "uuid": "String"
   }
}
```


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


```
{
   "status_code": 404,
   "status_message": "Not Found",
   "message": "reservationNotFound",
   "is_data": false,
   "data": 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": "Something Went Wrong",
    "is_error": true,
    "errors": "Array"
}
```


Other Rejection Errors


```
{
    "status_code": 404,
    "status_message": "Not Found",
    "message": "paymentCardNotFound",
    "is_data": false,
    "data": null
}
```

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

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

Cancel Reservation

The **Cancel Reservation** endpoint allows your application to void a reservation that has been placed but not yet captured or charged. In other words, you may cancel a reservation **only while the amount is still reserved** — once a portion or the entirety of the amount has been **captured** or **charged**, cancellation is no longer allowed.

Use this endpoint when:

- You need to abort a reservation because the order is changed, declined, or otherwise not to be fulfilled.
- The funds have not yet been transferred — the state must still be a pure “reserved” (authorized) status.
- You want to supply a human-readable reason or note for cancellation, to maintain auditability and traceability in your system.

You will find the endpoint path, expected inputs, validation rules, and example responses below.

Endpoint

```
GET https://demo-api.frontpayment.no/api/v1/connect/reservations/cancel/{{RESERVATION_UUID}}
```

Authorization

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:

```
{
   "note": "Your cancellation Note here"
}
```

Validation Rules

Make sure your request meets the following requirements:

Field	Type	Description
`note`	`string`	**Required** Reservation cancellation note.

Response

A successful request will return a `202` status with the following JSON payload:


```
{
   "status_code": 202,
   "status_message": "OK",
   "message": "cancelledOrderSuccessfully",
   "is_data": false,
   "data": null
}
```


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


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


API returns a `417` error, it means requested payload is not valid.


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

API return a `400` error, it means your requested reservation is already `COMPLETED` or `CANCELLED`.


```
{
   "status_code": 400,
   "status_message": "Conflict of Business Logic",
   "message": "reservationStatusAlreadyCompleted",
   "is_error": false,
   "errors": null
}
```


Reservation status is already `EXPIRED` and **NOT IN** `SENT` or `RESERVED NOT PAID`


```
{
   "status_code": 400,
   "status_message": "Conflict of Business Logic",
   "message": "prerequisiteFailedToCancelReservation",
   "is_error": false,
   "errors": null
}
```


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


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

Complete Reservation

The **Complete Reservation** endpoint finalizes a reservation after payment activity has occurred. You may invoke this endpoint only **after** any portion of the reserved amount has been **captured** or **charged** — you cannot complete a reservation that's strictly in a reserved (authorized-only) state.

Use this endpoint when:

- You've already captured or charged an amount (fully or partially) from the reservation.
- You want to mark the reservation as fully processed/completed in your system.
- You wish to attach a note or metadata for audit or tracking purposes.

The request requires the reservation's unique identifier (UUID), and returns a success status once the operation is accepted.

You will find method details, request/response formats, error conditions, and validation rules below.

Endpoint

```
POST https://demo-api.frontpayment.no/api/v1/connect/reservations/complete/{{RESERVATION_UUID}}
```

Authorization

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:

```
{
   "note": "Test Complete Note"
}
```

Validation Rules

Make sure your request meets the following requirements:

Field	Type	Description
`note`	`string`	**Required** Reservation completing note.

Response

A successful request will return a `202` status with the following JSON payload:


```
{
   "status_code": 202,
   "status_message": "OK",
   "message": "reservationCompletedSuccessfully",
   "is_data": false,
   "data": null
}
```


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


```
{
   "status_code": 404,
   "status_message": "Not Found",
   "message": "reservationNotFound",
   "is_data": false,
   "data": null
}
```


API returns a `417` error, it means requested payload is not valid.


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

API return a `400` error, it means your requested reservation status is in `SENT`, `RESERVED NOT PAID` or `EXPIRED`.


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


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


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

Resend Reservation

The **Resend Reservation** endpoint allows you to resend the payment link associated with an existing reservation to the customer—either via SMS or email. This is useful if the customer did not receive the original link, or it expired, or you merely wish to prompt payment again.

- If you choose to resend the link via **SMS**, both `countryCode` and `msisdn` (phone number) are required.
- If you choose to resend via **email**, you must supply a valid `email` address (while `countryCode` and `msisdn` can be null).
- You must supply one of these delivery methods (SMS or email) — at least one of those fields must be non-null in the request.
- Importantly, the SMS or email does **not** need to match the contact details already stored with the customer profile. The payment link can be sent to a **new phone number** or a **different email address**, making it flexible for scenarios where the customer wants to use an alternative contact method.

This ensures that the customer can always receive the payment link, even if their original phone number or email is unavailable.

Use this endpoint when:

- The customer has lost/misplaced the payment link.
- You want to remind or prompt the customer to complete the payment.
- You want to support multiple delivery channels (SMS or email) for better customer reach.

You will find endpoint details, authorization, request schema, validation rules, and sample responses below.

Endpoint

```
POST https://demo-api.frontpayment.no/api/v1/connect/reservations/resend/{{RESERVATION_UUID}}
```

Authorization

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:

```
{
   "countryCode": +47,
   "msisdn": "xxxxxxxx",
   "email": "example-email@email.com
}
```

Validation Rules

Make sure your request meets the following requirements:

Field	Type	Description
`countryCode`	`string`	**Conditional Required** Country code for the customer's phone number (e.g., "+47").
`msisdn`	`string`	**Conditional Required** Mobile Subscriber MSISDN Number (phone number). If you want to resend order paytment link via customer phone number.
`email`	`email`	**Conditional Required** Customer's email address. If you want to resend order paytment link via email.

Response

A successful request will return a `202` status with the following JSON payload:

```
{
   "status_code": 202,
   "status_message": "OK",
   "message": "resentOrderSuccessfully",
   "is_data": true,
   "data": null
```

```
}
```


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


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


API returns a `417` error, it means requested payload is not valid.


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


API return a `400` error, it means your requested order is already `COMPELTED` or `CANCELLED`.


```
{
   "status_code": 400,
   "status_message": "Conflict of Business Logic",
   "message": "orderStatusAlreadyCompleted",
   "is_error": false,
   "errors": null
}
```



```
{
   "status_code": 400,
   "status_message": "Conflict of Business Logic",
   "message": "orderStatusAlreadyCancelled",
   "is_error": false,
   "errors": null
}
```

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


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

Refund Reservation

The **Refund Reservation** endpoint enables merchants to initiate either full or partial refunds for a reservation using its `Reservation UUID`. Depending on your business workflow, you can refund the entire order or only specific items. Upon successful submission, the API responds with a `202 Accepted`, indicating that your refund request has been accepted and is pending processing.

This endpoint is ideal for scenarios such as:

- **Returns & Exchanges**: Revert payment for returned or exchanged items.
- **Order Modifications**: Adjust invoices or correct billing mistakes.
- **Partial Cancellations**: Process refunds for specific products rather than full orders.

Endpoint

```
POST https://demo-api.frontpayment.no/api/v1/connect/reservations/refund/{{RESERVATION_UUID}}
```

Authorization

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": 15,
   "products": [
      {
         "id": 510,
         "amount": 15
      }
```

```
    ],
    "source": "captured",
    "reference": "CAP123234"
}
```


Retrieve Product ID from Get Order Details API.

Validation Rules

Make sure your request meets the following requirements:

Field	Type	Description
`type`	`string`	`Required` Using type. Available types is `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.
`source`	`string`	**Required** Avaiable values are `captured` and `charged`.
`reference`	`string`	**Required** Using `captured` or `charged` uuid.

Response

A successful request will return a `202 OK` status with the following JSON payload:


```
{
    "status_code": 202,
    "status_message": "OK",
    "message": "orderRefundedSuccessfully",
    "is_data": true,
    "data": null
}
```


Error Response

API returns a `404` error, it means requested order with `RESERVATION_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"
}
```

Other Refund Rejection 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": "refundRejectionForRefundRequestGreaterThanOrderAmount",
   "is_error": false,
   "errors": null
}
```

```
{
   "status_code": 400,
   "status_message": "Conflict of Business Logic",
   "message": "refundRejectionForProductAmountExceed",
   "is_error": true,
   "errors": "Array"
}
```

```
{
   "status_code": 400,
   "status_message": "Conflict of Business Logic",
   "message": "refundRejectionForWeeklyThresholdExceed",
   "is_error": true,
   "errors": null
}
```

```
{
   "status_code": 400,
   "status_message": "Conflict of Business Logic",
   "message": "refundRejectionForRequestAmountThresholdExceed",
   "is_error": true,
   "errors": null
}
```

Get Reservation History

The **Get Reservation History By Time Frame** API enables you to retrieve all events associated with reservations within a specified time range. If no time frame is provided, the default is the last 24 hours. Start and end timestamp format should be in Unix Format (ex: 1706674723).

This endpoint is useful for:

- Monitoring reservation activities over a specific period.
- Auditing and troubleshooting reservation events.
- Generating reports on reservation actions and statuses.

Endpoint

```
GET https://demo-api.frontpayment.no/api/v1/connect/reservations/history/{{START_TIMESTAMP}}/{{END_TIMESTAMP}}
```

Authorization

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

Example:

```
Authorization: Bearer YOUR_FRONTPAYMENT_BEARER_TOKEN
```

Response

A successful request will return a `200 OK` status with the following JSON payload:


```
{
  "status_code": 200,
  "status_message": "OK",
  "message": "reservationRetrievedSuccessfully",
  "is_data": true,
  "data": [
    {
      "uuid": "RES3410395156",
```

```
      "title": "refund-sent-from-captured",
      "datetime": "21.01.2024 04:00",
      "sentTo": "",
      "actionBy": null,
      "note": null,
      "paymentMethod": null,
      "isRefundable": false,
      "amount": "100"
    },
    {
      "uuid": "RES3410395156",
      "title": "refund-sent-from-charged",
      "datetime": "21.01.2024 04:03",
      "sentTo": "",
      "actionBy": null,
      "note": null,
      "paymentMethod": null,
      "isRefundable": false,
      "amount": "200"
    }
  ]
}
```