Easy – Create a payment / Subscription

Please note that this is for custom integrations. If you're making use of a store/webshop, please go to: https://developers.nets.eu/nets-easy/en-EU/docs/use-a-webshop-plugin/

Index of content:

- Order Section
- Checkout Section
- Payment Methods Filtering
- Subscriptions
- Notifications / Webhooks
- Test functionality

The payload determines a lot of things for what your customer will see in the payment window as well as the notifications you receive during the different steps of a payment. I'll allow myself to add the full payload (JSON request) I've used for the example of the webhooks, since it contains most of the optional as well as the few mandatory fields/objects.

```
{
  "order": {
    "items": [
      {
        "reference": "item number",
        "name": "This is the item/product name",
        "unit": "stk",
        "unitPrice": 50000,
        "quantity": 1,
        "grossTotalAmount": 50000,
        "netTotalAmount": 50000
      }
    ],
    "reference": "This is the order number",
    "amount": 50000,
    "currency": "SEK"
  },
  "checkout": {
    "integrationType": "HostedPaymentPage",
    "returnUrl": "https://google.dk",
    "cancelUrl": "https://www.nets.eu/cancelPage",
    "termsUrl": "https://www.nets.eu/your-Terms-and-Condition-page-when-a-
customer-makes-a-purchase",
    "charge": true,
    "publicDevice": true,
    "merchantTermsUrl": "https://www.nets.eu/cookie-and-privacy-settings-
for-merchant",
    "merchantHandlesConsumerData": true,
    "appearance": {
      "displayOptions": {
        "showMerchantName": true,
        "showOrderSummary": true
      }
    },
    "consumer": {
      "reference": "consumerReference",
      "email": "test@test.dk",
      "shippingAddress": {
        "addressLine1": "Testgade 1, st.th",
        "addressLine2": null,
        "postalCode": "2300",
        "city": "København S",
        "country": "DNK"
      },
      "phoneNumber": {
        "prefix": "+45",
        "number": "12345678"
      },
      "privatePerson": {
        "firstName": "Test",
        "lastName": "Testesen"
      }
    }
  },
  "paymentMethodsConfiguration":[
    {
        "name":"Swish",
        "enabled":true
    },
    {
        "name":"MasterCard",
        "enabled":true
    },
    {
        "name":"Visa",
        "enabled":true
    },
    {
        "name":"MobilePay",
        "enabled":false
    }
  ],
  "notifications": {
    "webHooks": [
      {
        "eventName": "payment.reservation.created",
        "url": "https://webhook.site/090117b0-c7cd-4682-8458-1b75b3d7dc63",
        "authorization": "123456789",
        "header": [
          {
            "externalReference1": "thisIsATest1"
          },
          {
            "externalReference2": "this is a test 2"
          }
        ]
      },
      {
        "eventName": "payment.checkout.completed",
        "url": "https://webhook.site/090117b0-c7cd-4682-8458-1b75b3d7dc63",
        "authorization": "123456789"
      },
      {
        "eventName": "payment.reservation.created.v2",
        "url": "https://webhook.site/090117b0-c7cd-4682-8458-1b75b3d7dc63",
        "authorization": "123456789"
      },
      {
        "eventName": "payment.reservation.failed",
        "url": "https://webhook.site/090117b0-c7cd-4682-8458-1b75b3d7dc63",
        "authorization": "123456789"
      },
      {
        "eventName": "payment.charge.created",
        "url": "https://webhook.site/090117b0-c7cd-4682-8458-1b75b3d7dc63",
        "authorization": "123456789"
      },
      {
        "eventName": "payment.charge.created.v2",
        "url": "https://webhook.site/090117b0-c7cd-4682-8458-1b75b3d7dc63",
        "authorization": "123456789"
      },
      {
        "eventName": "payment.charge.failed",
        "url": "https://webhook.site/090117b0-c7cd-4682-8458-1b75b3d7dc63",
        "authorization": "123456789"
      },
      {
        "eventName": "payment.refund.initiated",
        "url": "https://webhook.site/090117b0-c7cd-4682-8458-1b75b3d7dc63",
        "authorization": "123456789"
      },
      {
        "eventName": "payment.refund.completed",
        "url": "https://webhook.site/090117b0-c7cd-4682-8458-1b75b3d7dc63",
        "authorization": "123456789"
      },
      {
        "eventName": "payment.refund.failed",
        "url": "https://webhook.site/090117b0-c7cd-4682-8458-1b75b3d7dc63",
        "authorization": "123456789"
      }
    ]
  }
}
```

Order section (back to top)

```
{
  "order": {
    "items": [
      {
        "reference": "item number",
        "name": "This is the item/product name",
        "unit": "stk",
        "unitPrice": 40000,
        "quantity": 1,
        "taxRate": 2500,
        "taxAmount": 10000,
        "grossTotalAmount": 50000,
        "netTotalAmount": 40000
      }
    ],
    "reference": "This is the order number",
    "amount": 50000,
    "currency": "DKK"
  },
```

The order.items array is what contains the items where the outer reference at the buttom where the amount and currency is stated is the Order Number, while the inner references are stated for references item numbers/id's – could for example be used to call a specific item from the db to populate name (product name/description), unitPrice (price per item), unit (stk, liter, kg etc). The amount stated is always in ears/cent/pennies so for example the reference above is stated as 50000 which is 500,00DKK for this example.

Everything within the example is mandatory except for the taxRate and taxAmount. These variables can be excluded if you do not wish to have it included. When making the POST, the amount will just be stated with 0 in VAT. If added, you can for example do the calculation like so:

taxRate is stated as 2500 which equals to 25,00%

netTotalAmount: quantity * unitPrice

grossTotalAmount: netTotalAmount * 1,25 taxAmount:

grossTotalAmount – netTotalAmount

Checkout Section (back to top)

Integration types:

```
"checkout": {
  "integrationType": "HostedPaymentPage",
  "returnUrl": "https://google.dk",
  "cancelUrl": "https://www.nets.eu/cancelPage",
  "termsUrl": "https://www.nets.eu/your-Terms-and-Condition-page-when-a-
customer-makes-a-purchase",
  "charge": true,
  "publicDevice": true,
  "merchantTermsUrl": "https://www.nets.eu/cookie-and-privacy-settings-for-
merchant",
  "merchantHandlesConsumerData": true,
  "appearance": {
    "displayOptions": {
      "showMerchantName": true,
      "showOrderSummary": true
    }
  },
  "consumer": {
    "reference": "consumerReference",
    "email": "test@test.dk",
    "shippingAddress": {
      "addressLine1": "Testgade 1, st.th",
      "addressLine2": null,
      "postalCode": "2300",
      "city": "København S",
      "country": "DNK"
    },
    "phoneNumber": {
      "prefix": "+45",
      "number": "12345678"
    },
    "privatePerson": {
      "firstName": "Test",
      "lastName": "Testesen"
    }
  }
},
```

As you can see, I've made use of the integrationType HostedPaymentPage. You can make use of 2 different types: *HostedPamentPage* and *EmbeddedCheckout*. HostedPaymentPage needs to have returnUrl which is specifying where the customer will be directed to when a payment has been created successfully. If the payment is declined, the customer will be redirected back to the checkout page/HostedPaymentPage. For the HostedPaymentPage you can add a cancelUrl. This is to specify where the customer will be redirected in case they're cancelling the payment (Could for example be if the customer had forgotten to add an item to their order and needs to go back)

If you're making use of the EmbeddedCheckout, you need to have the url stated instead. The url stated needs to be an exact match to the page where the iframe/overlay is at.

returnUrl/url (Hosted/Embedded – Mandatory)
cancelUrl (Hosted – optional)
termsUrl (mandatory) **charge** (optional – default: false)
publicDevice (optional – default: false)
merchantTermsUrl (optional)
merchantHandlesConsumerData (optional – default: false
merchantTermsUrl (optional)
appearance (optional – default merchant name and order is set to true)
consumer (optional – if merchantHandlesConsumerData is set to true, the information provided will be included)

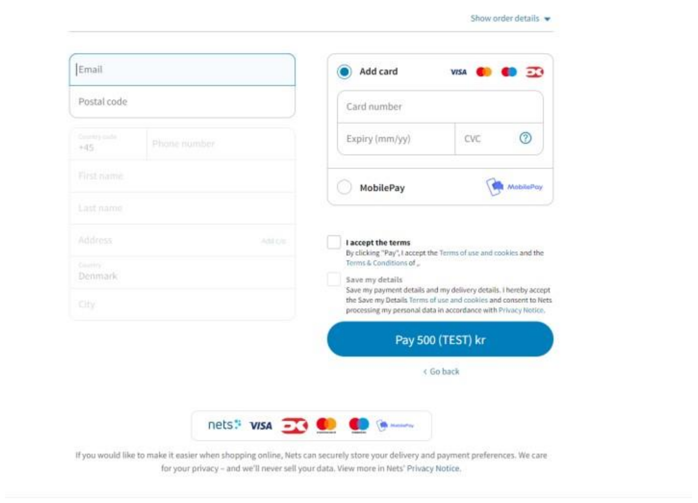
termsUrl: the page/path where your customer can read about your terms and conditions policy when placing an order.

Charge: whether a payment should be charged right after a successful reservation has been made. You can also leave this out completely, since if not stated, the default setting is "charge":false. If set to *true*, the payment will be charged immediately after.

publicDevice and merchantHandlesConsumerData is connected, since publicDevice specifies whether customer data should be stored or not + whether the customer should be allowed to save their information + card details.
If publicDevice is set to true, the customer will not have the opportunity to save their information, but it's also connected to the information either provided in the payload sent in by you (merchantHandlesConsumerData=true) or if the customer has to fill in the information themselves in the payment window. To save and store the consumer/card information, we're matching the merchantId and the postalCode + email of the consumer, however all the information is important for this process.

Examples for merchantHandlesConsumerData:

Set to *false:*



Show order details

Email
Postal code
Country code +45 | Phone number
First name
Last name
Address | Add c/o
Country Denmark
City

Add card
Card number
Expiry (mm/yy) | CVC
MobilePay

I accept the terms
By clicking "Pay", I accept the Terms of use and cookies and the Terms & Conditions of ,.

Save my details
Save my payment details and my delivery details. I hereby accept the Save my Details Terms of use and cookies and consent to Nets processing my personal data in accordance with Privacy Notice.

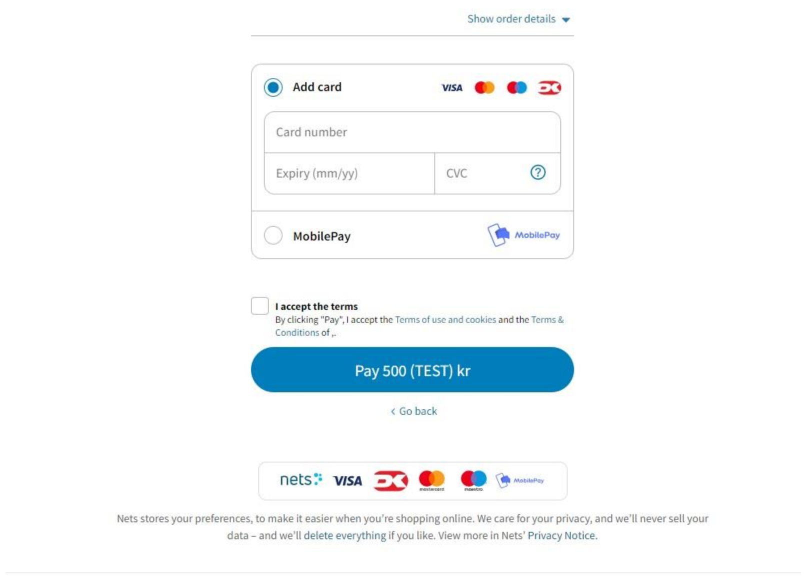
Pay 500 (TEST) kr

‹ Go back

If you would like to make it easier when shopping online, Nets can securely store your delivery and payment preferences. We care for your privacy – and we'll never sell your data. View more in Nets' Privacy Notice.

The customer is requested to fill in their information.

Set to *true:*



Show order details

Add card
Card number
Expiry (mm/yy) | CVC
MobilePay

I accept the terms
By clicking "Pay", I accept the Terms of use and cookies and the Terms & Conditions of ,.

Pay 500 (TEST) kr

‹ Go back

Nets stores your preferences, to make it easier when you're shopping online. We care for your privacy, and we'll never sell your data – and we'll delete everything if you like. View more in Nets' Privacy Notice.

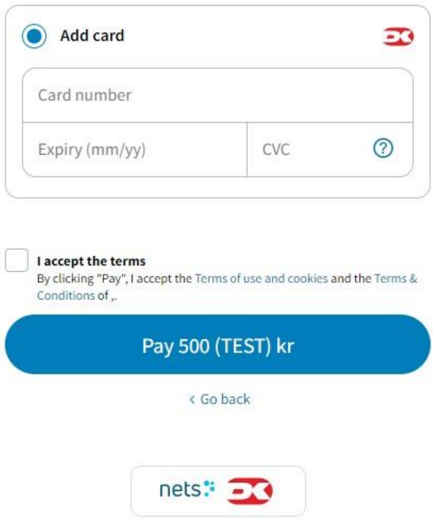
Please note that even if you set merchantHandlesConsumerData to true, you do not need the consumer object for it to work. No consumer data will be attached to the payment though.

Payment methods filtering (optional) [(back to top)]

```
"paymentMethodsConfiguration":[
  {
      "name":"Dankort",
      "enabled":true
  },
  {
      "name":"MasterCard",
      "enabled":true
  },
  {
      "name":"Visa",
      "enabled":true
  },
  {
      "name":"MobilePay",
      "enabled":false
  }
],
```

This functionality only works for payment methods which are already available to the specific account. For each method which you wish to have shown in the payment window, you can add the method here under *name* and whether the payment method should be visible. You can read about the different filter options here: https://developers.nets.eu/nets-easy/en-EU/api/payment-v1/#v1-payments-post-bodypaymentmethodsconfiguration

As an example, I've filtered everything out except for Dankort to illustrate:


Add card

Card number

Expiry (mm/yy) CVC

I accept the terms
By clicking "Pay", I accept the Terms of use and cookies and the Terms & Conditions of ,.

Pay 500 (TEST) kr

‹ Go back

nets

```
},
"paymentMethodsConfiguration":[
    {
        "name":"Dankort",
        "enabled":true
    },
    {
        "name":"MasterCard",
        "enabled":false
    },
    {
        "name":"Visa",
        "enabled":false
    },
    {
        "name":"MobilePay",
        "enabled":false
    }
],
```

If the section isn't added, all payment options will be shown in the payment window.

Subscriptions (back to top)

In case you wish need to have subscriptions, you only need to take 2 things into consideration:

1. Is the subscription required to be flexible regarding amount and/or charge dates?

2. How many subscriptions do you have/expect to have?

For the first question, it's regarding whether we recommend for you to make use of the recurring type *regular recurring* or *UCOF (Unscheduled Card On File)*. For regular recurring, the issuer expects for you to either only charge the payment for example once a month or that the amount is the same each time.

If you expect for the subscription charge to vary and/or the amount to vary, we recommend you make use of the UCOF subscription type, since it's dynamic in nature.

The greatest difference between the 2 types is that regular subscriptions are registered with an end date/expiry date for when the subscription will expire, where UCOF subscriptions expire when the subscription has been inactive for 365 days. When making use of either type, please note that we strongly recommend for you to make use of the webhook event *payment.reservation.created.v2* since you will receive the subscriptionId returned after a subscription has been created.

Regular subscription (optional):

To create a regular subscription, the only difference from a regular payment is that you need to include the following to the JSON payload:

```
"subscription":{


"interval":0,

  "endDate":"2055-12-
31T23:59:59"
},
```

Interval is how often you expect for the subscription to be charged. You can of course do as in the example and set the interval to 0. No issues will come of it. The endDate is when the subscription is set to expire. Of course, this date can be changed when the customer needs to update their card at a later date if needed by adding the following to the object: "subscriptionId":"the customers subscriptionId here" – this will show the checkout, but no amount will appear, since it's only for updating the subscription.

Unscheduled subscription (optional):

Just like with the regular subscription creation, you only need to add another object to the JSON payload:

```
"unscheduledSubscription":{

  "create":true
},
```

Unlike the regular subscription, you do not need to define neither the interval nor the endDate, since the UCOF subscription is created for the purpose to be chargeable multiple times if needed as well as to expire when the subscription has been inactive for 365 days.

Just like with the regular subscription, if a customer needs to update their card information, you can add “unscheduledSubscriptionId”:”the customers UCOF subscriptionId here”.

Please note that when you create the subscriptionId with either type, you can either include an amount if needed, or set the amount to zero. If the customer only need to create a subscription, it’s important to set the amount to 0 to make sure you do not experience issues further down the line, since 1,00DKK/EUR/SEK/NOK reservations are frowned upon.

Notifications/webhooks (back to top)

```
  "notifications": {
    "webHooks": [
      {
        "eventName": "payment.reservation.created",
        "url": "https://webhook.site/090117b0-c7cd-4682-8458-1b75b3d7dc63",
        "authorization": "123456789",
        "header": [
          {
            "externalReference1": "thisIsATest1"
          },
          {
            "externalReference2": "this is a test 2"
          }
        ]
      },
      {
        "eventName": "payment.checkout.completed",
        "url": "https://webhook.site/090117b0-c7cd-4682-8458-1b75b3d7dc63",
        "authorization": "123456789"
      },
      {
        "eventName": "payment.reservation.created.v2",
        "url": "https://webhook.site/090117b0-c7cd-4682-8458-1b75b3d7dc63",
        "authorization": "123456789"
      },
      {
        "eventName": "payment.reservation.failed",
        "url": "https://webhook.site/090117b0-c7cd-4682-8458-1b75b3d7dc63",
        "authorization": "123456789"
      },
      {
        "eventName": "payment.charge.created",
        "url": "https://webhook.site/090117b0-c7cd-4682-8458-1b75b3d7dc63",
        "authorization": "123456789"
      },
      {
        "eventName": "payment.charge.created.v2",
        "url": "https://webhook.site/090117b0-c7cd-4682-8458-1b75b3d7dc63",
        "authorization": "123456789"
      },
      {
        "eventName": "payment.charge.failed",
        "url": "https://webhook.site/090117b0-c7cd-4682-8458-1b75b3d7dc63",
        "authorization": "123456789"
      },
      {
        "eventName": "payment.refund.initiated",
        "url": "https://webhook.site/090117b0-c7cd-4682-8458-1b75b3d7dc63",
        "authorization": "123456789"
      },
      {
        "eventName": "payment.refund.completed",
        "url": "https://webhook.site/090117b0-c7cd-4682-8458-1b75b3d7dc63",
        "authorization": "123456789"
      },
      {
        "eventName": "payment.refund.failed",
        "url": "https://webhook.site/090117b0-c7cd-4682-8458-1b75b3d7dc63",
        "authorization": "123456789"
      }
    ]
  }
}
```

The webhook events varies greatly for the reservation steps, however for other events as illustrated in the document illustrating the webhook responses, does not vary when it's concerning regular payments/single payments. It's only in the reservation state, where the webhooks differs greatly, since payment.reservation.created contains the full information, however can potentially take a little bit longer to arrive compared to the other 2.
eventName – states which webhook you wish to "subscribe" to and is defined in the creation of the payment as shown above. url – which server/url should the webhooks be sent to authorization – if there is a password to your server for being able to deliver the webhook, this field should be populated with the key. If there is no password/key, then the field should be populated with dummy data (a minimum of 8 characters is required)

You can test the functionality in Postman via this link:
https://www.postman.com/nicoaioe/workspace/nets-easy-api-testing/collection/28765559-3ad210dc-5f6f-48a2-9525-7d35296c863f

Charge single payments: https://developers.nets.eu/nets-easy/en-EU/api/payment-v1/#v1payments-paymentid-charges-post

Card verification for regular subscriptions: https://developers.nets.eu/nets-easy/enEU/api/payment-v1/#v1-subscriptions-verifications-post

Charge regular subscriptions: https://developers.nets.eu/nets-easy/en-EU/api/payment-v1/#v1subscriptions-charges-post

Card verification for UCOF subscriptions: https://developers.nets.eu/nets-easy/en-EU/api/paymentv1/#v1-unscheduledsubscriptions-verifications-bulkid-get

Charge UCOF subscriptions: https://developers.nets.eu/nets-easy/en-EU/api/payment-v1/#v1unscheduledsubscriptions-charges-post

You can of course read about everything noted here on our tech site:
https://developers.nets.eu/nets-easy/en-EU/api/