



unifiedpost
PAYMENTS

OCS (Online Collect Services)

Merchant API Description for Payment Items

1. Preface	7
1.1. Introduction	7
1.2. Lifecycle of a Payment Item	7
1.3. About this document (and how it evolves)	9
2. Getting Started	10
2.1. Web Service Endpoints	10
2.2. Web Service Authentication	10
2.2.1. API Keys	10
2.2.2. Simulation vs Live	10
2.2.3. API Keys for access to sensitive resources	11
2.2.4. Take care	11
2.2.5. Key expiration	11
2.3. Web Service Versioning	11
3. Payment Item Initiation	12
3.1. Quick Start	12
3.2. Request Body	14
3.2.1. Batch Support	14
3.2.2. Structure	15
3.2.3. Payment Item Attributes	16
3.2.3.1. PaymentAttributes	16
3.2.3.2. PayerAttributes	19
3.3. Response	19
3.4. Special Item Initiation Topics	21
3.4.1. Additional Notification URLs	21
3.4.2. Change Notification Callback	21
3.4.3. Amounts	22
3.4.3.1. Examples	22
3.4.3.1.1. Fixed Amount (=standard case, cfr supra)	22
3.4.3.1.2. List of Values	22
3.4.3.1.3. Free fillable amount	23
3.4.3.1.4. List of Values OR Free amount	23
3.4.3.2. adjustableAmount attributes	23
3.4.3.3. Amount Validations	24
3.4.4. Custom Checkout Profiles	25
3.4.4.1. What	25
3.4.4.2. Use	25
3.4.5. Custom MetaData	26
3.4.5.1. What	26
3.4.5.2. Definition	26

3.4.5.3. Use of customMetadata	26
3.4.6. Redirect after a Transaction is completed	27
3.4.6.1. Webflow redirect	27
3.4.6.2. Redirect on a mobile device	27
3.4.7. Skip checkout page phases	28
3.4.7.1. Product selection	28
3.4.7.2. Product specific refinement	28
3.4.7.2.1. Open Banking bank selection	28
3.4.8. Low level access to Checkout resources.	29
3.4.9. Payable documents	31
3.4.10. Supported currencies	33
3.4.10.1. Request	33
3.4.10.2. Response	34
4. Payment Item Status Information	35
4.1. Basic Payment Item Status	35
4.1.1. Request	35
4.1.2. Response	35
4.2. Extended Payment Item Status	38
4.2.1. Request	38
4.2.2. Response	38
4.3. Payment item transactions	41
4.3.1. Request	41
4.3.2. Response	41
4.4. Payment transaction details	42
4.4.1. Request	42
4.4.2. Response	42
4.4.2.1. iDEAL	42
4.4.2.2. SEPA Direct Debit	43
4.4.2.3. PXG	43
4.5. Batch Payment Item Status	44
4.5.1. Service Endpoint	44
4.5.2. Request	44
4.5.2.1. Request attributes	44
4.5.3. Response	44
5. Payment Item Update	47
5.1. Introduction	47
5.1.1. Item Status Update	47
5.1.2. Other Item Update use cases	47
5.2. Item Update Service	48
5.2.1. Service Endpoint	48

5.2.2. Request Body	48
5.2.3. Protocol	52
5.3. Usage Guidelines	53
6. Short Checkout URL	53
6.1. Introduction	53
6.2. Request	54
6.3. Response	54
7. Templates & Campaigns	55
7.1. Introduction	55
7.2. How to register a Template?	55
7.2.1. Request	55
7.2.2. Response	56
7.3. Campaign based Checkouts.	56
8. Payment QR Codes	57
8.1. Introduction	57
8.2. Service Endpoint	57
8.3. Request	57
8.4. Response	58
8.5. Different types of QR Code	59
8.6. Add extra Info to the QR Image	59
8.7. Request QR Code at Item Initiation	60
9. Bancontact Mobile URL Intent	63
9.1. Introduction	63
9.2. Service Endpoint	63
9.3. Request	63
9.4. Response	64
F10. Payout Report Requests	65
10.1. Introduction	65
10.2. Authentication Header	65
10.3. List of Payout Reports Service	65
10.3.1. Request	65
10.3.2. Response	65
10.4. Payout Report Details Service	66
10.4.1. Request	66
10.4.2. Response	66
10.5. Payout Report Format	67
11. Payment Item Identifiers	69
12. Refunds	69
12.1. Introduction	69

12.1. Endpoint	70
12.2. Request	70
12.3. Error handling	70
13. Appendix A. Hands-on examples using the API	71
12.1. CURL	71
12.1.1. Initiating a Payment Item	71
12.1.2. Fetching the Payout reports	72
12.1.3. Updating a Payment Item	73
12.2. Postman	75

Version History

Version	Publish Date	Description
2018-06-27.01	30/08/2018	Initial Version
2018-10-01.01	01/10/2018	Update of Payout Reports API Request - More generic report request to allow extension to other report types - §10
2018-10-08.01	08/10/2018	Update of Payment Item Status Response - Extension for items with Adjustable Amounts - §4
2018-11-19.01	19/11/2018	Added option to ask for QR Codes at Item Initiation - §8.7
2018-12-17.01	17/12/2018	Added support for iDEAL QR Codes - §8 Payout Reports available in CODA format - §10 Some API's support both item public identifier and MerchantItemIdentifier as input parameter - see §11 Item Public Identifier added as HTTP Header of ChangeNotificationCallback - §3.4.1
2019-02-12.01	12/02/2019	Extra QR Code generation options - §8 Support for additional notificationURLs - § 3.4.1 Payout Reports available in MT940 format - §10
2019-04-10.01	10/04/2019	Option to make documents payable - § 3.4.8
2019-12-02.01	02/12/2019	Option to use MerchantItemIdentifiers to retrieve Batch Payment Item Status - § 4.5
2020-04-29	29/04/2020	Update branding
2020-08-07	07/08/2020	Added dueDate to responses of API calls used to retrieved payment item status § 4

2020-09-15	15/09/2020	Option to request status image with text
2020-10-08	08/10/2020	Updated the maximum length for a reconciliation reference
2021-03-13	13/03/2021	Added payment app redirect URLs
2021-04-30	30/04/2021	Introduced endpoint to query OCS supported currencies
2021-08-18	18/08/2021	Clarified payment app redirect URLs vs webflow redirect
2021-11-29	29/11/2021	Introduced possibility to skip checkout page phases § 3.4.7
2022-05-10	10/05/2022	Added SEPA Direct Debit in transaction details response
2022-07-12	12/07/2022	Added Refund API
2022-09-23	2022-09-23	Updated the list with supported languages
2022-11-15	2022-11-15	Clarified the default payment item Expiry Date
2022-12-29		New paymentDetailedState attribute in PI and Trx status responses. Extended error information for PXG transactions
2023-01-06		Clarified notification is sent (temporary solution) when payment item state transitions to SUBMITTED

1. Preface

1.1. Introduction

Essentially, the OCS platform facilitates online payments: both web based and mobile. An obvious application for OCS would be to power an eCommerce webshop: checkout and pay online.

In practice, the platform goes well beyond this familiar use-case. Actually, OCS is a toolkit that provides all means to make sure that an '**Item**' gets paid: now, or at a future point in time. The implemented scope goes beyond the fundamental *pay-now* scenario.

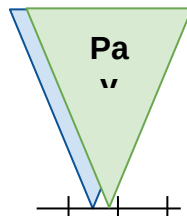
It all starts with an **Item**. The **Item** is the subject of an intended transaction. Depending on the **Item**-type, various attributes come into play. I.e. a **Payment Item**, obviously has an **amount** trait, but can also have a **dueDate** attribute. Consequently, a **dueDate**, allows the platform to actively track payment status. **Payment Items** depend upon remittance information. After all, eventually these items will have to be reconciled. Etc.

Thus, **Payment Items** are not just retail goods. They are typically also bills, invoices, fines or premiums.

Apart from **Payment Items**, there can also be SDD **eMandate Items** and PSD2 **Consent Items**.

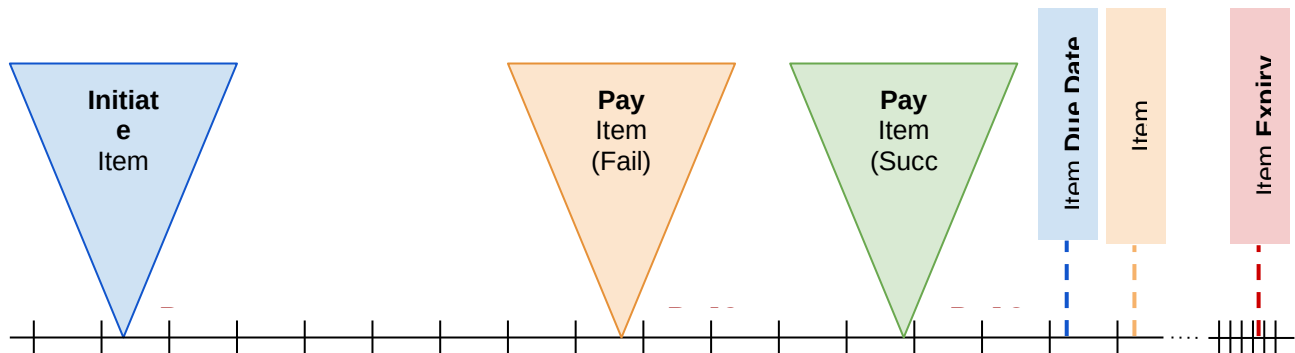
1.2. Lifecycle of a Payment Item

In the context of traditional eCommerce, the lifecycle of a Payment Item is pretty compact: An Item is created, and immediately after that, it gets paid



OCS supports a lifecycle that is more long living than the case described above. As illustrated on following diagram, a **Payment Item** can be created at point in time **D**. At that point, a Checkout Page is prepared and made available through a Checkout URL. The Checkout URL can be used immediately (i.e. embedded in a web page), to pay that Item, or at a later stage (i.e. as part of an email).

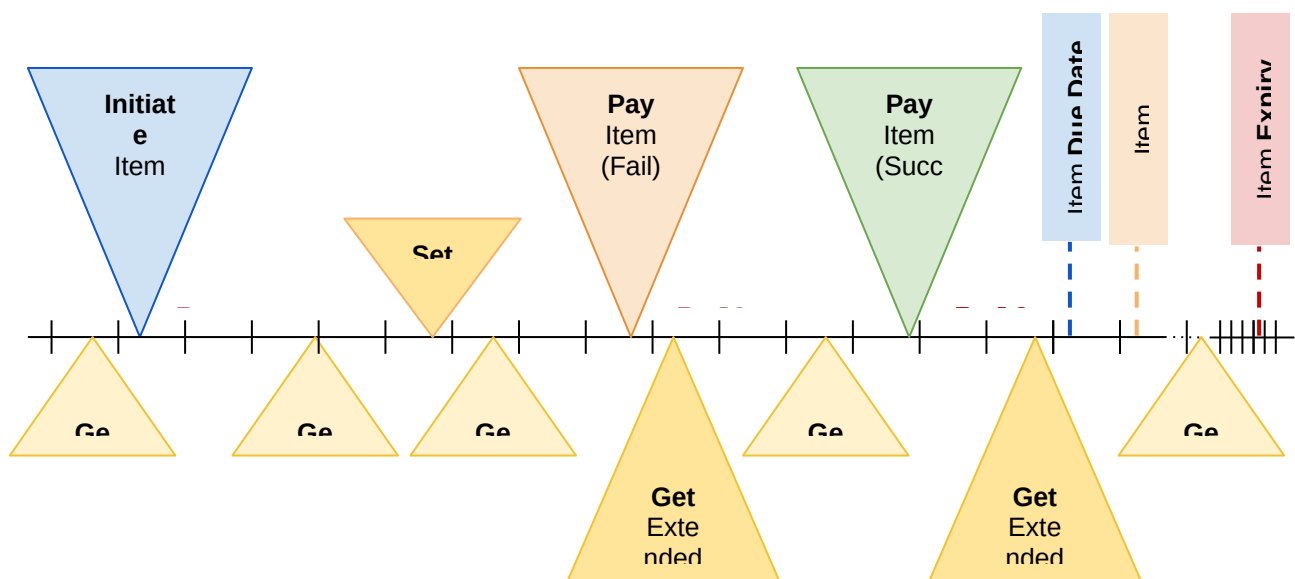
In practice, the payment can take place before, or after the **Item Due Date**. The Due Date is simply the point in time a Merchant/Biller considers the Item should be settled. Beyond that date, the Merchant/Biller may consider to charge overdue fines.



In any case, the Item will be paid before the **Item Expiry Date**, because beyond the Expiry Date, the Item is no longer tracked by the platform. It is simply no longer there, so it is no longer 'payable'.

A Merchant/Biller may choose to *suspend* an Item at any point in time (but obviously before it expires). Thus, the **Item Suspension Date** is the date after which the Item can no longer be paid. It is still part of the system, so Checkout URL's in older emails will still behave correctly. The Item is naturally no longer payable, because the Merchant/Biller decided so.

During its lifespan, the Item runs through multiple State changes. At any time, it is possible to query, or even update, the actual Status of the Item.



Dynamic Status information can be integrated into online resources like websites, emails and mobile apps. These resources get the most up to date Status information directly from the payments platform. On top of that, the Merchant itself has exclusive access to a more elaborate set of 'Extended Status' Information, including the timestamp when a payment has been authorised, and which payment product has been used.

Furthermore, the platform also exposes an Item Update interface. Through the Item Update API, a Merchant/Biller can communicate whether an Item has been paid through a different channel. Even if an Item has been settled through i.e. a regular Credit Transfer, it is valuable to maintain,

and feedback, the correct payment status to the payer. Additional possibilities offered by the Item Update API are discussed later in this document.

1.3. About this document (and how it evolves)

This document is not final. You may want to obtain an up-to-date revision at regular intervals.

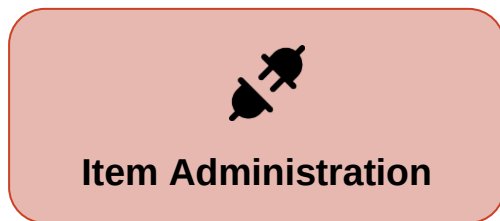
2. Getting Started

2.1. Web Service Endpoints

There are two levels of access to the platform:

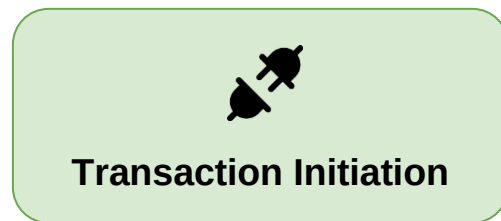
1. **Administration of Items** that need to be paid, signed, approved, etc
2. **Transaction Initiation** by the actual payer, approver, etc

Both API's are serviced by a different host:



All services require an API

{hostname} = **dolphin.nav-**



All endpoints are public

{hostname} = **tx.nav-**

This document primarily describes the **Item Administration** API. So generally, replace {hostname} by dolphin.pay-nxt.com.

2.2. Web Service Authentication

2.2.1. API Keys

Each OCS account comes with a set of API Keys. An API Key authenticates the caller to the platform. An authenticated caller is authorised to administer all Items belonging to its account: create new Items, update attributes and query for status.

All requests carry the API key as an [HTTP Authentication header](#).

Example:

Authorization: OCS-API 59956956-ac3e-46b3-a418-8d6e2aa19999

2.2.2. Simulation vs Live

There are two types of API Keys:

1. **Simulation** API Keys
2. **Live** API Keys

Use a Simulation API Key while integrating your platform with OCS. In Simulation Mode, no fund flows will take place. Actually, not even all payment products are available in Simulation Mode: only the ones that support some kind of end-to-end simulation behaviour. This makes it possible to test against the API, without actually having to perform Penny Tests ¹: Statuses will be updated, feedback will be triggered, reminders will be issued,... but no actual money is involved.

¹ Penny Tests are live production tests, using small amounts like 0.05€

This also means that eventually Penny Tests *are* required, to assess the reconciliation flow in the Merchants backoffice.

2.2.3. API Keys for access to sensitive resources

Regular API Keys can be upgraded to an elevated security level. Such API Keys are required for access to resources that contain sensitive information or to initiate actions that require a higher security level. An example of this is the information in the [Payout Report](#).

Upgrading regular API keys can be done via the Merchant Dashboard (see [OCS - Unifiedpost Payments™ Dashboard - User Manual](#)).

2.2.4. Take care

API Keys are first class credentials.

Please take into account following measures:

- Make sure to manage API Keys as secret passwords.
- During the integration process, only use Simulation API Keys.
- When in doubt about the integrity of a Key, issue a new Key and set an end-date on the compromised Key.²
- Create dedicated API Keys for access to sensitive information. Handle these keys with even more attention.

2.2.5. Key expiration

API keys expire if they have not been used for 3 months. If this happens, a Merchant can renew the key via the Merchant Dashboard (see [OCS - Unifiedpost Payments™ Dashboard - User Manual](#)).

2.3. Web Service Versioning

This document describes **Version 1** of the Transaction Item API.

Therefore, replace **{version}** by **v1**.

Overall ...

`https://{hostname}/api/{version}/{resource}`

... becomes ...

`https://dolphin.pay-nxt.com/api/v1/{resource}`

² API Keys are currently managed through the OCS service desk.

3. Payment Item Initiation

3.1. Quick Start

Following snippet is the most concise possible JSON body to initiate one Payment Item:

```
{
  "items":{
    "1":{
      "paymentAttributes":{
        "amount":143.16,
        "description":"Utility Invoice 766243 - November 2018",
        "reconciliationReference":"REF-900000766243"
      }
    }
  }
}
```

When it is **POST**'ed to:

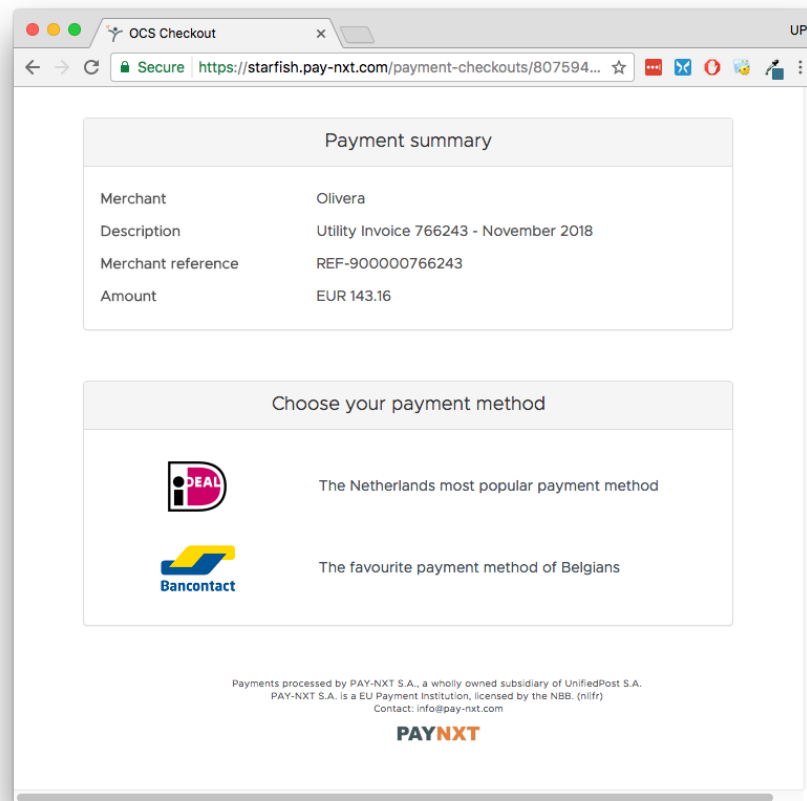
```
https://{hostname}/api/{version}/items
```

*(don't forget to add the API Key to the **Authorization** header)*

... then the platform returns a set of URL's:

```
{
  "1":{
    "id":"807594ae-d9c0-4ae5-ad81-166eca6e81a7",
    "paymentLinks":{
      "checkout":"https://tx.pay-nxt.com/payment-checkouts/807594ae-...-166eca6e81a7",
      "statusText":"https://tx.pay-nxt.com/payment-items/807594ae-...-166eca6e81a7/status.txt",
      "statusImage":"https://tx.pay-nxt.com/payment-items/807594ae-...-166eca6e81a7/status.png"
    }
  }
}
```

The Checkout URL (**checkout**) is probably the first one you are looking for, as it points to the Checkout page of our Item:



The Checkout page reflects the **amount**, **description** and **reconciliationReference** provided in the JSON body from the request. The presented payment products, are pre-configured for the given Merchant.

It is up to the developer to actually put the response URL's to work. They can be integrated in webpages, embedded in emails, etc. They can also be generated on the fly (i.e. as part of a dynamic website), or requested upfront (i.e. in context of a batch system).

See Section "[Appendix A. Hands-on examples using the API](#)", for a more detailed description to reproduce this example.

3.2. Request Body

3.2.1. Batch Support

The Request Body is a sequence of 1 to 100 Items (Grouped by the **items** element). Grouping multiple Items in a single request may be beneficial during batch processing.

```
{
  "items": {
    "1": {
      "paymentAttributes": {
        "amount": 143.16,
        "description": "Utility Invoice 766243 - November 2018",
        "reconciliationReference": "REF-900000766243"
      }
    },
    "2": {
      "paymentAttributes": {
        "amount": 345.16,
        "description": "Utility Invoice 766768 - November 2018",
        "reconciliationReference": "REF-900000766736"
      }
    }
  }
}
```

Each Item is prefixed by a name. Here, that's "1" and "2". These names reappear in the Response to be able to correlate the result back to the original request.

```
{
  "1": {
    "id": "c97d82f7-f82b-4bc5-b6b5-5d96cbf50342",
    "paymentLinks": {
      "checkout": "https://tx.pay-nxt.com/payment-checkouts/c97d82f7-...-5d96cbf50342",
      "statusText": "https://tx.pay-nxt.com/payment-items/c97d82f7-...-5d96cbf50342/status.txt",
      "statusImage": "https://tx.pay-nxt.com/payment-items/c97d82f7-...-5d96cbf50342/status.png"
    }
  },
  "2": {
    "id": "04023ebd-588e-43ed-a4aa-7e97fb536231",
    "paymentLinks": {
      "checkout": "https://tx.pay-nxt.com/payment-checkouts/04023ebd-...-7e97fb536231",
      "statusText": "https://tx.pay-nxt.com/payment-items/04023ebd-...-7e97fb536231/status.txt",
      "statusImage": "https://tx.pay-nxt.com/payment-items/04023ebd-...-7e97fb536231/status.png"
    }
  }
}
```

In this example, the names are chosen to be "1" and "2", but it is also possible to choose any alpha-numerical name like e.g. "62765a4e3ca" and "b2e6583e3a5".

3.2.2. Structure

Each Item element will result in one single Checkout page.

```
{
  "expiryDate": "2018-08-17",
  "paymentAttributes": {
    ...
  }
  "payerAttributes": {
    ...
  }
}
```

Attribute Name	Description	Example Value
expiryDate	The date after which the Item will be removed from the platform. Beyond this date, the platform will no longer be able to serve Status feedback. Note: This is a processing instruction for the payment item, not a business attribute and should only be used in very specific circumstances. The expiry date should NOT be confused with the due date, the date by when the Item should be settled. The due date can be set as payment attribute. Type: Date Format: YYYY-MM-DD Valid Values: A date between TODAY and [TODAY + 24 months] Required: No Default if not provided: - If dueDate is not provided: [TODAY + 9 months] - If dueDate is provided: [dueDate + 9 months]	2018-08-10
paymentAttributes	Element specifying the properties of the payable item itself	
payerAttributes	Element grouping the attributes of the person to whom the payment item will be addressed (i.e. the payer)	

If the same Checkout page is also used to initiate ancillary Items, like SDD eMandates, there exist additional sections for these Item types as well. In this section, only **paymentAttributes** and **payerAttributes** are described.

3.2.3. Payment Item Attributes

The following is an example of the specification of a standard payment item, ie with a fixed amount (see special topic Amounts for alternatives to specify amounts):

```
***
  "paymentAttributes":{
    "amount":239.83,
    "description":"Water Consumption November 2017",
    "reconciliationReference":"133767115325",
    "reconciliationReferenceType":"OGM_VCS",
    "reconciliationReferenceDisplay":"Customer:1337 / Invoice:67115325",
    "dueDate":"2018-02-17",
    "almostDueDate":"2018-02-12",
    "changeNotificationUrl":"https://www.merchant.com/backoffice/itm/4424",
    "paymentAppRedirectUrls":{
      "redirectURL":"https://en.wikipedia.org/wiki/Home_page",
      "successURL":"https://en.wikipedia.org/wiki/OK",
      "failureURL":"https://en.wikipedia.org/wiki/Failure"
    }
  },
  "payerAttributes":{
    "language":"nl-BE",
    "mobileNr":"+32474586969",
    "email":"jsmith@gmail.com"
  }
}
***
```

3.2.3.1. PaymentAttributes

The **paymentAttributes** groups the business attributes of the payable item itself.

Attribute Name	Description	Example Value
amount	The amount that will be settled through the online transaction Type: Decimal Format: 999990.99 Required: Yes	239.83
currency	The currency of a payment item should be passed in the payment initiation request. If no currency is passed, it will be set to EUR. Remark: a payment item can only be paid with a payment product that supports that currency for the given merchant. Type: Text Format: Alphabetical 3 positions (ISO 4217) Valid values: The currency must be one of the currencies supported by OCS. To find out which currencies are supported by OCS, a dedicated endpoint can be queried. See 3.4.9. Supported currencies . Required: No Default if not provided: EUR	EUR
description	A short textual description about the Item to be	Water

	paid, aimed at the human payer, so he/she knows what is being paid. Type: Text Format: AN max 50 positions Required: Yes (*) Some additional constraints may apply depending on the payment methods offered See "Merchant Implementation Guide"	Consumption November 2017
reconciliationReference	The information that will be passed on as Remittance information. The Merchant/Biller will observe this value in the OCS Account Statements and/or Bank Account Statements to allow him to reconcile the incoming funds with the original Payment Items. Type: Text Format: AN 140 positions Required: Yes (*) Some additional constraints may apply depending on the payment methods offered. See "Merchant Implementation Guide"	133767115325
reconciliationReferenceType	In some business contexts, it is common to format the information in reconciliationReference in line with a well known industry standard. If the Merchant/Biller prefers to adhere to such a standard, that can be specified here. Type: Text Valid Values: See "Merchant Implementation Guide" Required: No Default if not provided: unstructured	OGM_VCS
reconciliationReferenceDisplay	The reconciliationReference value will always be shown in the header of the Checkout page. If the Merchant/Biller prefers not to expose the reconciliationReference as is, then he has the option to provide an alternative value here. Type: Text Format: AN 70 positions Required: No Default if not provided: None	Customer:1337 / Invoice:67115325
dueDate	The date by when the Item should be settled. This date drives logic like triggering reminder events or changing the face of the graphical status icon. Type: Date Format: YYYY-MM-DD Valid Values: today() <=dueDate <=expiryDate Required: No Default if not provided: None	2018-02-17
almostDueDate	The date as from when the system considers an item to be 'nearly due'. This attribute influences the moment when the status icon will turn to "Nearly Due".	2018-02-10

	Type: Date Format: YYYY-MM-DD Valid Values: Between dueDate and dueDate-20 Required: No Default if not provided: dueDate-5	
merchantItemIdentifier	A natural key, chosen by the Merchant/Biller to reference the Item at a later stage in another API call. For more details, see "Linking data feeds together". Type: Text Format: AN 50 positions Valid Values: Has to be unique across all Items posted by the Merchant. In other words, it is not possible to reuse a given merchantItemIdentifier for multiple items. Required: No Default if not provided: None	abe01d2f8-23e-17cfe
changeNotificationUrl	A URL that will be called by the platform whenever change-events happen on the Item. For more details, see "Change Notification Callback". Type: Text Format: AN 400 positions Required: Only if Item will have to be updated, later on. Valid Values: An HTTP(S) URL Default if not provided: None	https://www.merchant.com/backoffice/item/4424
paymentAppRedirectUrls	URLs to redirect on a mobile device after a transaction is completed, in case the payment is done with a payment app on that device.	
redirectURL	redirectURL will be used in any outcome (success, failure or canceled by user) unless a more specific url is provided (successURL or failureURL) In case redirectURL is not provided the starfish status page will be used as default. Valid values: A URI to be resolved by the mobile device. This can be an HTML URL, resulting in a mobile webpage being shown. Or it can be a mobile deep link, pointing to a specific location in a particular mobile app. Required: At least one of redirectURL, successURL or failureURL must be present. The system will choose the most applicable one.	https://tx.pay-nxt.com/payment-items/2645505f-d0a4-41dc-b444-5ff51134ed75/status or myApp://item/656257/successful-payment
successURL		
failureURL		
profile	Please refer to 3.4.4. Custom Checkout Profiles	

3.2.3.2. PayerAttributes

The **payerAttributes** groups the business attributes of the payer for whom the payment item is created and to whom it will be sent:

Attribute Name	Description	Example Value
language	Which language to use in communication with the payer (reminders, notifications, checkout-page,...) Type: Text Required: No Default: NULL Format: Case Insensitive Validation³: Should be one of the supported subset of language codes: "en", "ca-ES", "cs-CZ", "da-DK", "de", "de-LU", "el-GR", "es-ES", "et-EE", "fi-FI", "fr-BE", "fr-FR", "fr-LU", "it-IT", "lt-LT", "lv-LV", "nb-NO", "nl-BE", "nl-NL", "pl-PL", "pt-PT", "ro", "ru-RU", "sk-SK", "sr-Latn", "sv-SE"	nl-BE
mobileNbr	The mobile number to use for an SMS communication with the payer Type: Text Format: AN Required: No Validation: Starts with '+' Followed by numbers, no dots, no spaces, no separators	+3247985697
email	The email address to use for an email communication with the payer Type: Text Format: AN Required: No Validation: Following structure should be followed: 1) At least one character : A-Z0-9._%+- 2) @ 3) At least one character : A-Z0-9.- 4) A dot 5) At least 2 characters A-Z All characters are case insensitive.	jsmith@gmail.com Or john.doe@thebooker.co.uk

3.3. Response

Every initiated Payment Item, results in a set of URL's, grouped in the **paymentLinks** element:

³ OCS has a flexible cascading fallback mechanism in place and will try to identify the desired language even if the provided language code is not an exact match (e.g. if the language is not recognized at all, OCS will not reject the call but fall back into English).

```

"id": "807594ae-d9c0-4ae5-ad81-166eca6e81a7",
"paymentLinks": {
  "checkout": "https://tx.pay-nxt.com/payment-checkouts/807594ae-...-166eca6e81a7",
  "statusText": "https://tx.pay-nxt.com/payment-items/807594ae-...-166eca6e81a7/status.txt",
  "statusImage": "https://tx.pay-nxt.com/payment-items/807594ae-...-166eca6e81a7/status.png"
}

```

- **checkout** : represents the URL of the Checkout page.
- **statusImage** : contains the URL of the image that reflects the Item's current payment status. The server returns an image that changes dynamically, depending on the actual state. The **Content-Type** is image/png. If the payment item id doesn't exist, "unknown" is returned.



Open



Almost due



Past due



Paid



Expired



Suspended



Unknown



Submitted

- **statusText** : provides an endpoint from which the current payment status of an item can be fetched. Basically identical to **statusImage**, except that the **Content-Type** is text/plain. This endpoint provides CORS headers so that it can be called from a mail / external web page. It is meant to be used in the backend for automation and not to be displayed as text on a user screen

Remark: parameters can be added to the URL of the status image to display the status text below the icon. Example:

```

https://tx.pay-nxt.com/payment-items/807594ae-...-166eca6e81a7/status.png?label=true&lang=fr-FR

```

Parameter Name	Description	Example Value
----------------	-------------	---------------

label	Allows the user to have the status text displayed below the label. Valid Values: true, false	true
lang	Allows the user to determine in which language the status text should be shown. Validation⁴: Should be one of the supported subset of language codes: "en", "ca-ES", "cs-CZ", "da-DK", "de", "de-LU", "el-GR", "es-ES", "et-EE", "fi-FI", "fr-BE", "fr-FR", "fr-LU", "it-IT", "lt-LT", "lv-LV", "nb-NO", "nl-BE", "nl-NL", "pl-PL", "pt-PT", "ro", "ru-RU", "sk-SK", "sr-Latn", "sv-SE	fr-FR

3.4. Special Item Initiation Topics

3.4.1. Additional Notification URLs

Additional changeNotificationUrls can be added by **POST**ing following request:

```
{
  "stateChangeNotificationURL": "https://updelivery.com/ocs-item-update/66652"
}
```

To:

```
https://{hostname}/api/{version}/payment-items/{item-identifier}/notifications
```

where {**item-identifier**} can be either the item public identifier or the merchant item identifier (see Chapter 11 on the use of Item identifiers).

3.4.2. Change Notification Callback

The platform will issue an HTTP GET on all the **changeNotificationUrl**'s whenever the Item state changes to **submitted⁵** or **paid**.

To allow the remote system to map a specific notification to the correct payment item, the item public identifier is passed as value in the HTTP header "**X-Item-Identifier**".

```
X-Item-Identifier: 807594ae-d9c0-4ae5-ad81-166eca6e81a7
```

⁴ OCS has a flexible cascading fallback mechanism in place and will try to identify the desired language even if the provided language code is not an exact match (e.g. if the language is not recognized at all, OCS will not reject the call but fall back into English).

⁵ Temporary solution, which will be removed in the near future

If the remote system responds with a status code different from 5xx, the callback is considered successful. If the remote system responds with a 5xx status code, OCS will retry the callback 5 times within a one minute interval.

This callback mechanism deliberately does not communicate state information to the remote merchant system. It concerns a simple 'signal' to indicate something has changed. It is up to the merchant system to retrieve the actual state, by calling one of the status url's - see [Payment Item Status Information](#) for the different options to retrieve status information.

The platform does not inject status information back into the merchant system because of security considerations. Such endpoints, exposed by merchants, are often subject to abuse. Without proper authentication, any fraudster would be able to trick the merchant into believing that an Item was paid for. Therefore, the platform forces the merchant system to explicitly pick up the state from the payments platform.

3.4.3. Amounts

A standard payment item will define a fixed amount to pay eg when the payment item corresponds with an invoice to be paid.

In other scenarios an adjustable amount may be defined, eg in case of donations several amounts may be presented to the payer who can then choose the one he prefers to donate.

To support this, an optional element **adjustableAmount** is foreseen to specify the amount options for a payment item.

On Item initiation, either the **amount** or the **adjustableAmount** needs to be filled, depending on the use case.

3.4.3.1. Examples

3.4.3.1.1. Fixed Amount (=standard case, cfr supra)

```
...
  "paymentAttributes":{
    "Amount":239.83,
    ...
  }
...
```

3.4.3.1.2. List of Values

```
...
  "paymentAttributes":{
    "adjustableAmount":{
      "default": 10.00,
      "options": [
        {
          "value": 10.00,
          "description": "stickers"
        },
        {
          "value": 15.00,
          "description": "pencils"
        },
        {

```

```

        "value": 20.00,
        "description": "stickers+pencils"
      }
    ],
    ...
  }
}

```

3.4.3.1.3. Free fillable amount

```

...
"paymentAttributes":{
  "adjustableAmount":{
    "default": 100.00,
    "range": { "minimum": 5, "maximum": 200 }
  },
  ...
}
...

```

3.4.3.1.4. List of Values OR Free amount

```

...
"paymentAttributes":{
  "adjustableAmount":{
    "default": 15.00,
    "options": [
      {
        "value": 10.00
      },
      {
        "value": 15.00
      },
      {
        "value": 20.00
      }
    ],
    "range": { "minimum": 5, "maximum": 100 }
  },
  ...
}
...

```

3.4.3.2. adjustableAmount attributes

Attribute Name	Description	Example Value
----------------	-------------	---------------

default	The default amount that will be presented in the checkout page. In the 'list of values' use case, this value should therefore be one of the options defined. See 'Validations' In the 'range' use case, this value should fit inside the defined range. See 'Validations' Type: Decimal Format: 999990.99 Required: Yes	143.16
options	An array that enumerates a list of key/value objects. To be used for the 'list of values' use case. The amounts will be presented to the payer in the order they are listed. Type: JSON Array Required: No	
value	One of the amounts to be chosen from a list of amounts. Type: Decimal Format: 999990.99 Required: Yes	10.00
description	An optional description linked to the amount presented in the list of amounts. Type: Text Format: Alphanumeric (max 30) Required: No	stickers
range	A JSON object defining a range of amounts allowed for this payment item. Relevant in the 'free amount' use case. Required: No	
minimum	If present, the 'free amount' use case is applicable and this value specifies the minimum amount to be filled in in the editable field. Type: Decimal Format: 999990.99 Required: No	10.00
maximum	If present, the 'free amount' use case is applicable and this value specifies the maximum amount to be filled in in the editable field Type: Decimal Format: 999990.99 Required: No	1000.00

3.4.3.3. Amount Validations

- One of the two must be present: the simple **amount**, or the complex **adjustableAmount**.
- The maximum number of **options** to be defined = 5.
- In case the options are defined, the **default** has to be one of the options.amounts.
- In case a range is defined, the **default** has to fit inside the range.
- In case both options and a range are defined, the default has to be either:

- One of the options.amounts; OR
- Fit inside the range
- All amounts defined: **default, amount, value, minimum, maximum**
 - Should be defined with 2 decimals
 - Need to be > 0.
- When a **minimum** is defined, the **maximum** needs to be defined as well and vice versa.
- Minimum should be > [the platform wide minimum amount (currently set to 0.01€)]
- Maximum should be < 10.000
- Minimum should be < maximum

3.4.4. Custom Checkout Profiles

3.4.4.1. What

A Checkout Profile allows to specify a (sub)set of Payment Methods to be presented to the Payer on the checkout page. Several Checkout Profiles can be defined, each profile specifying a different set of available Payment Methods.

At Payment Item Initiation, it is possible to specify the profile to be used on the Checkout Page for that item. If no profile is specified, the default Profile is used.

3.4.4.2. Use

For example:

```

...
  "paymentAttributes":{
    "amount":239.83,
    "description":"Water Consumption November 2017",
    "reconciliationReference":"133767115325",
    ...
    "profile":"74ae03f8-d2bb-11e7-8941-cec278b6b50a"
  },
...

```

Attribute Name	Description	Example Value
profile	The unique identifier of the Checkout Profile to be used for the Payment Item to be generated. It is optional, if not specified, the default profile will be used. Type: UUID Required: No	74ae03f8-d2bb-11e7-8941-cec278b6b50a

3.4.5. Custom MetaData

3.4.5.1. What

Merchants may want to add additional (customer) data to the remittance information of the credit transfers for pay-out. These data may be useful for different purposes:

- To refund the payer if necessary
- To solve reconciliation issues (the remittance info is the starting point for reconciliation)
- ...

To facilitate this, a customMetaData field is defined and can be specified during payment item initiation.

3.4.5.2. Definition

The customMetaData field is defined as a list of key/value pairs:

```
"customMetaData": {  
  {  
    "customKey1" : "value1",  
    "customKey2" : "value2",  
    ...  
  }  
}
```

It is an optional field and may contain up to 5 key/value pairs.

For example:

```
...  
  "paymentAttributes": {  
    "amount": 239.83,  
    "description": "Water Consumption November 2017",  
    "reconciliationReference": "133767115325",  
    ...  
    "customMetaData": {  
      "documentID" : "FFH_273589298736",  
      "customerID" : "77365423393",  
    }  
  }  
...
```

3.4.5.3. Use of customMetadata

If available, the customMetadata will appear in:

- The response body of the **Extended Payment Item Status** service (see [Extended Payment Item Status Information](#))
- The Payout Report (see [OCS - Merchant Implementation Guide](#))
- The MVA generic statement file (see [OCS - Merchant Implementation Guide](#))

Next, a Merchant may define a specific remittance information to be used in the PAIN.001 file using the extra data defined in the customMetaData.

See [OCS - Merchant Implementation Guide](#) for more details on how to do this.

3.4.6. Redirect after a Transaction is completed

3.4.6.1. Webflow redirect

After a transaction is completed (failed or successfully), the status confirmation page is shown to the payer. The developer may decide to redirect the browser to another page, instead of stopping the webflow at the status confirmation page.

Redirect URLs for one given Item may be different in case of different use-cases. The browser may be redirected differently if a payment transaction was initiated from an eMail, or from a specific web-page. Therefore, this information is not part of the initial Payment Item attributes, but can it be added dynamically, whenever the Checkout URL is presented.

Thus, it is possible to postfix the **checkout** URL as follows:

?redirectURL =http%3A%2F%2Fwww.google.be	The URL is called whenever the browser arrives at the confirmation page, successful or not. This URL is used in case the successURL or failureURL is not provided.
?successURL =http%3A%2F%2Fwww.google.be	The URL is called whenever the browser arrives at the confirmation page and the transaction outcome was successful.
?failureURL =http%3A%2F%2Fwww.google.be	The URL is called whenever the browser arrives at the confirmation page and the transaction outcome was <i>not</i> successful.
&redirectDelay =23	Override the default delay before redirecting. The default (and minimum) timeout before redirecting is 5 seconds.

As illustrated above, the URL's need to be [URL-encoded](#).

Example:

<code>https://tx.pay-nxt.com/payment-checkouts/807594ae-...-166eca6e81a7?successURL=http%3A%2F%2Fwww.google.be&redirectDelay=12</code>
--

Note: Webflow redirect URLs need to be whitelisted before they can be used. To do this contact your implementation consultant or the Unifiedpost Payments servicedesk.

3.4.6.2. Redirect on a mobile device

When a payment attempt is made with a payment app, it might be interesting to redirect the payment app to another app.

Unlike the webflow redirect, this information is added as part of the initial Payment Item attributes. See the description of the attribute **paymentAppRedirectUrIs** in paragraph [3.2.3. Payment Item Attributes](#).

3.4.7. Skip checkout page phases

3.4.7.1. Product selection

The checkout process can be optimised to allow for the payer to jump directly to a product selected by the merchant. For some products, the payer can even be redirected further in the payment process, e.g. to the authentication page of the selected bank (see [3.4.7.2.1 Open Banking bank selection](#)).

The direct product selection can be achieved by appending the **checkout** URL as follows:

<code>?product={product}</code>	This will redirect the payer to the respective checkout page of the selected product Example product values: • OPEN_BANKING • BANCONTACT • IDEAL • ...
---------------------------------	---

Example:

<code>https://tx.pay-nxt.com/payment-checkouts/807594ae-...-166eca6e81a7?product=OPEN_BANKING</code>
--

This will redirect the payer immediately to the Open Banking checkout page which allows for the selection of the desired bank.

3.4.7.2. Product specific refinement

3.4.7.2.1. Open Banking bank selection

For the Open Banking product, the payer can be redirected to a specific bank without going through the product- and bank selection page. This functionality can be applied, for example, to deliver invoices via email, containing direct bank links.

Each bank is identified with a Financial Platform ID. To skip the bank selection page as well, the respective financialPlatformId should be appended to the **checkout** URL as follows:

<code>?product=OPEN_BANKING&financialPlatform={financialPlatformId}</code>	This will redirect the payer to the authentication process page of the bank, corresponding to the given Financial Platform Id. The merchant needs to be aware of the Financial Platform IDs assigned to the supported banks.
--	--

Example:

<code>https://tx.pay-nxt.com/payment-checkouts/807594ae-...-166eca6e81a7?product=OPEN_BANKING&financialPlatform=c3f695f0-...-92ec73a0a762</code>
--

3.4.8. Low level access to Checkout resources.

The Checkout URL (**checkout**), obtained when a Payment Item is initiated, is typically used to return an html-based webpage. This yields a very straightforward way to integrate payment support to your existing online applications.

Notice however that it is also possible to return a JSON equivalent of that same page:

Perform a regular GET on the Checkout resource (<https://tx.pay-nxt.com/payment-checkouts/{item-public-identifier}>), but add the HTTP **Accept** header:

Accept: application/json

The system will no longer return the html page, but instead, a JSON representation of the full resource.

As such, it is possible for i.e. mobile apps to obtain the dynamic link to open up the Bancontact mobile app:

Deprecated response:

```
{
  "type": "BCMC",
  "id": "89449013-c5d8-4285-8611-71a0b954fc97",
  "title": "Bancontact",
  "description": "bc.description",
  "iconAddress": "/img/bcmc/logo_small.png",
  "vaultUrl": "https://turtle.pay-nxt.com/card-data",
  "qrCodeBase64EncodedData":
    "iVBORw0KGgoAAAANSUHEUgAAAPoAAAD6AQAAAAACgl2eQAAABp01EQVR42uZZXY7DIAyELXGAHomrcyQ0gMTaMyZNmofVvvASInaqsD34
    /wNa0bPa5rV0muZbbzmiC3TAvztm7MbvjSHY4l9IcCNGjBz4Mhi6ZgiEJ++8H345xaAm8JlvSoCDLnDQLy+x0TmANL/OGp03K0
    +bA7k4xUsT0s2rH6t1XsD7qPIfdauAoabWgDtYvqnm7opAr4PAytjLYRSDEgHMMVQwSbrm8ZAQWSLEo4DgzWSAjJTInFw0oGVb1pACmLmT
    /0gcAYBRNznX4ts55BQALoMZ60FR0y6lWAGgptBfLRstq6zPQgAE0c0sQkyMgXcpYgrASAYIMYQG
    +svOv4MCEPMsxtiCAQTKHsoi8TT2iswUP4Wy2Lg0IPsD6aDVmWCYqmczNY8XjoSZ
    +AAu06ICgJsv7xYF0ZdxVwC0aoyOC8DMyiW5Ehrq0VKA3wqYhoAb9vYrvPC5MNMWCWBjiR2jh3GqEgTYooSPuqEUT0Wg9DWSQ
    +vvd6SbAxlyVJbV9IS7T04NMP2XoLfjqkEKeP6NeoAH+G/gB2fGJ1HkMUXwAAAAAE1FTkSuQmCC",
  "qrCodeTransactionId": "1dd27fa2-10e8-448d-a18c-a5cccfddbb3d",
  "urlIntent": "BEPGenApp://DoTx?transId=1bep.turtle.pay-nxt.com/bc-bep*22a37a01c506$NZH2HU4M2QZQUA6ZAHORKSP6",
  "cardPaymentMethodVisible": true,
  "qrCodePaymentMethodVisible": true
},
```

For Bancontact, the **qrCodeBase64EncodedData** and **urlIntent** fields were moved under the **qrCodes** field. It can contain one or two QR codes, depending on the following scenarios:

In case of a standalone consent or a standalone payment, one qr code is returned:

<pre>{ "type": "BCMC", "id": "2f30c34d-9fbc-4d01-90a3-3c5bdd1c3f4b", "title": "BANCONTACT", "description": "bc.description", "iconAddress": "/img/bcmc/logo_small.png", "headerIconAddress": "/header/img/bcmc/logo_small.png", "wipEnabled": true, "vaultUrl": "http://localhost:9200/card-data", "qrCodeBase64EncodedData": "iVBORw0KGgoAAAANSUHEUgAAAPoAAAD6AQAAAAACgl2eQAAABsUIEQVR4Xu2YUYrDMAxEBT5Aj+Sr50g5QEC</pre>

```

rN3IWtyy7vzsQkYIrv5+pR5LTyN/jiM/MRzxAxwN0PECHgCMqXketxjmv6lFwAupzFXDGAJ76uvJGQIkqYGgxTgTW1xZ
uBwRMRh+WK5ACaj0xniGQsly8soAh4/3oyX8OYLJXJbeHVO+5ACvkNJQGGLf4ADitki2wWhlmO+f+O1gAasLEwGbs
Dj1mwF04JW1waihV7RgBtN+xNGp9Ro9IK6AWEnWCYTnNI3w7LAOgZYYGCrVfGhs2A4Ly534ye3HRDcyA0pUa65V
cAu+G4AVwM2GyT22VA5nyVkB2jcQ9E4F117ICql6WwVpjynh7+TsAydHogEpg9GQxBAAWi8rslqtNK4DCGRqLPdNL
bPKGrj0bgMMKkt9NoCQHGSgtybzfaaWXTSuADnzdr7TAuqXk1qsdl6DVqYD6qqBcQJYh+4kfeO6KJ9ja2leQCqpk1I
rRiBKvQDdDys/uwlQOPr/zRCgFYPP1DgvxPoBobgY60Er2CxnAWQXy5omuG68v0ZZAMhT42qzbbVjBPwWD9DxAB
0P0PE38AXzecKk3c6WAgAAAABJRU5ErkJggg==",
  "urlIntent": "BEPGenApp://DoTx?TransId=1bep.localhost:/bc-
bep*cc6526bdeebb$OBFQLIAQN2M3LOFWY556SQEN",
  "cardPaymentMethodVisible": true,
  "qrCodePaymentMethodVisible": true,
  "qrCodes": [
    {
      "qrCodeBase64EncodedData":
        "iVBORw0KGgoAAAANSUHEUgAAAPoAAAD6AQAAACgl2eQAAABsUIEQVR4Xu2YUYrDMAxEBT5Aj+Sr50g5QEC
        rN3IWtyy7vzsQkYIrv5+pR5LTyN/jiM/MRzxAxwN0PECHgCMqXketxjmv6lFwAupzFXDGAJ76uvJGQIkqYGgxTgTW1xZ
        uBwRMRh+WK5ACaj0xniGQsly8soAh4/3oyX8OYLJXJbeHVO+5ACvkNJQGGLf4ADitki2wWhlmO+f+O1gAasLEwGbs
        Dj1mwF04JW1waihV7RgBtN+xNGp9Ro9IK6AWEnWCYTnNI3w7LAOgZYYGCrVfGhs2A4Ly534ye3HRDcyA0pUa65V
        cAu+G4AVwM2GyT22VA5nyVkB2jcQ9E4F117ICql6WwVpjynh7+TsAydHogEpg9GQxBAAWi8rslqtNK4DCGRqLPdNL
        bPKGrj0bgMMKkt9NoCQHGSgtybzfaaWXTSuADnzdr7TAuqXk1qsdl6DVqYD6qqBcQJYh+4kfeO6KJ9ja2leQCqpk1I
        rRiBKvQDdDys/uwlQOPr/zRCgFYPP1DgvxPoBobgY60Er2CxnAWQXy5omuG68v0ZZAMhT42qzbbVjBPwWD9DxAB
        0P0PE38AXzecKk3c6WAgAAAABJRU5ErkJggg==",
      "urlIntent": "BEPGenApp://DoTx?TransId=1bep.localhost:/bc-
bep*cc6526bdeebb$OBFQLIAQN2M3LOFWY556SQEN"
    }
  ]
}

```

In case of a payment initiated with a consent request, a qrCode will contain the following fields: **qrCodeBase64EncodedData**, **urlIntent** and **payerConsentAgreed**.

The qr code that corresponds to payerConsentAgreed=true will capture the consent.

The qr code that corresponds to payerConsentAgreed=false will not capture the consent.

If captureType=REQUIRED => one QR code is returned, with payerConsentAgreed=true

If captureType=PRECHECKED/OPTIONAL => two QR codes are returned, one of them with payerConsentAgreed=true and the other one with payerConsentAgreed=false.

```

...
{
  "type": "BCMC",
  "id": "2f30c34d-9fbc-4d01-90a3-3c5bdd1c3f4b",
  "title": "BANCONTACT",
  "description": "bc.description",
  "iconAddress": "/img/bcmc/logo_small.png",

```

```

    "headerIconAddress": "/header/img/bcmc/logo_small.png",
    "wipEnabled": true,
    "vaultUrl": "http://localhost:9200/card-data",
    "cardPaymentMethodVisible": true,
    "qrCodePaymentMethodVisible": true,
    "qrCodes": [
      {
        "qrCodeBase64EncodedData":
"iVBORw0KGgoAAAANSUHEUgAAAPoAAAD6AQAAACgl2eQAAABuEIEQVR4Xu2XUWrEMAxEBT5Aj+Sr+0g5QM
CdN3LZpZT2tWMRaelobz+mGslJ7d9j1ffMt3iAjgfoelAOA6sUH0urcc19zbt0EFmA/m4BV429q6ZvTz4lkCgBQ4uPJb3l2
xYeByhzO0mxUoFdVirLyXiJANLlcB4Yb/3oyX8O0Oqu1Osg1c9SgBO3G18aC40ncgCcidiqF5SiWjPf+f4gASF6sulKaxs
PtEwVwxWOq0YX3cBukH8QAqx8Nun5ZY52SJQFbGURROqzyP1WrARgWdTZUBBbheX4aRCg8VU9ezXHGrIZBl
mALLc2+anzoHfagVnARqGZQeGmex8yC1B1uPgDpJnhd60oglxuVSNbjrPrNbOAM8HsOtXNoyAP4MpnLDU6pB2YB
dhydM3tlhEprH2YA6gudZ9pPPtWR6+DAJeG3t9fmyOWexUrAljMXn+MD7+xs6cUSqOADqnbw9KsesEkAaxVHQE4z
WWCeRtiEcC2tFbX9XKSfBBA7x+/ze1XILLxAgEv2OJRhrE5gE6o257YdVZwD66tK2fUTzY4okgAHndNaynz9U/zAF+i
wfoelCOB+j4G/gE8o3rkBjS3sYAAAAASUVORK5CYII=",
        "urlIntent": "BEPGenApp://DoTx?TransId=1bep.localhost:/bc-
bep*92c76c42d673$ES6EXYC4I6JOM3ZWJ5RITUNP",
        "payerConsentAgreed": true
      },
      {
        "qrCodeBase64EncodedData":
"iVBORw0KGgoAAAANSUHEUgAAAPoAAAD6AQAAACgl2eQAAABskIEQVR4Xu2W0W3DMAxECWiAjKTPZiHE
MDyHZXECND0tweYSFObfPk4UDWq8nsc8Zn5iBvoulGOG+gQcETFo77nOGeeC2QuUI2zAepvPTLPGMBzvypvBJSo
ypTGefDS91HCHYFVAs+oT5Wcga5W5jeZ/xpQ5tIs8Z03Ahh1Nej9IdU1F6Dj6NnfB++VtQGOcQbNEIBWLKXkvQBey
TM7Gh+U1lqxAhIWlql8qWYzXmR6ABVDx2zhXUim8jYxD0DzjoMntSwx5znAjIB6pUeal5Kmlu1+GQGJd6k15KWuJXs
BiQ/vltULJrAlOwEh72r7omt7J5oBPftDU9PLRcy0AjCupdlhrTD4ZC/NsgAYnNQe6R6pa3G1YguA/9x1y42P0NRop3TB
BUDOS9f2YQ7eS6YJsldlHzYm6PlsBOzuSFpgwjTuKtMCqMHnvJUD0ClherYCOtiM47kfQYQXwHNw12Uzdqhr+pkPKE
hjXppRxg9Y0kWB9nJEaf/QD+ipqRLXLe69hkC2j8kHJpWryzkAqZGvNi3duJZ8GL1WACdMq2Sbgl5fhw/wLW6g4wY6b
qDjb+AHwhD3Kozt/LYAAAAASUVORK5CYII=",
        "urlIntent": "BEPGenApp://DoTx?TransId=1bep.localhost:/bc-
bep*1fe8111ce3d8$KZBD7S6ZHX2AOGQ3OUVEHVWW",
        "payerConsentAgreed": false
      }
    ]
  }
  ...

```

The old fields(**qrCodeBase64EncodedData** and **urlIntent**) will still be returned in the response(only for standalone consents and payments) until clients will migrate to the new structure.

3.4.9. Payable documents

A merchant can have his own document delivery infrastructure but might want to add payment services offered by Unifiedpost Payments. To support this, when OCS Payment Items are created,

it is possible to add extra document and delivery related attributes to link to the corresponding Documents.

Additionally the items will be delivered to *itspaid app*, a mobile app to process payment requests.

To support this at Item Initiation time, it's possible to add the document attributes as **paymentOption** in the Request body.

Example Request:

```
{
  "items": {
    "1": {
      "paymentAttributes": {
        "amount": 50.12,
        "description": "Invoice March 2019",
        "reconciliationReference": "+++234/2334/37873+++",
        ...
      }
      "payerAttributes": {
        ...
      }
      "paymentOptions": {
        "document": {
          "documentIdentifier" : "201900056215",
          "documentDisplayName" : "Invoice March 2019",
          "imageURL" : "http://www.speos.be/documents/doc1258965-48512",
          "imageAuth" : "SPEOS_APIKEY",
          "receiverIdentifier" : "125698745",
          "receiverAddresses" : {
            "emailAddresses": ["jos@gmail.com", "jos@unifiedpost.com"],
            "mobileNumbers": ["+32555985696"],
            "ibanNumbers": ["BE2611112223333"]
          }
        }
      }
    }
  }
}
```

Attribute Name	Description	Example Value
document	Contains details on the document for which this Payment Item is created. Required: No	
documentIdentifier	Unique identifier for this Document as assigned by the Merchant/Sender of the Document (aka 'invoice number') <i>Can be the same as the MerchantItemIdentifier</i>	201900056215

	Type: 1-255 chars Required: Yes	
documentDisplayName	Name that will be shown in the DeliveryChannel Type: 1-255 chars Required: No. Default: <i>description</i> of the Payment Item	Invoice March 2019
imageURL	Specifies the URL to retrieve the related document Type: URL Required: Yes Constraint: ? (URL specific definition) 400 chars	http://www.speos.be/documents/doc1258965-48512
imageAuth	Should specify the way authorization is set up to access the document URL. Possible values to be defined - start without	SPEOS_APIKEY
receiverIdentifier	Unique identifier of the receiver of the Document as defined and known by the Merchant (aka 'customer number') Type: 1-255 chars Required: Yes	125698745
receiverAddresses	An array of addresses to whom the document needs to be delivered. Defined as key/value pairs Where the key specifies the receiver type. Following keys/receiver types are possible: - MOBILE - IBAN - EMAIL More addresses of the same type may be specified.. (see example with 2 email addresses) Type: Array of Key/Value Pairs Required: Yes Constraint: cfr above	["jos@gmail.com", "jos@unifiedpost.com"]

3.4.10. Supported currencies

OCS only supports a limited set of currencies. To find out which currencies are supported at a given moment in time, a dedicated endpoint can be queried.

3.4.10.1. Request

```
GET {hostname}/api/{version}/supported-currencies

eg
GET https://dolphin.pay-nxt.com/api/v1/supported-currencies
```

3.4.10.2. Response

The list of ISO 4217 currency codes of the supported currencies is returned.

```
[  
  "HRK",  
  "RON",  
  "EUR",  
  "PLN",  
  "GBP",  
  "CZK",  
  "DKK",  
  "SEK",  
  "HUF",  
  "NOK"  
]
```

4. Payment Item Status Information

The Merchant/Biller can retrieve Status Information of one or more specific payment items. Several services exist to retrieve different sets of (extra) information on a payment item.

The URLs are not exposed in the public domain, but go back to the Item Administration API. To **GET** this (extended) status information, the API Key needs to be provided in an **Authorization** header.

4.1. Basic Payment Item Status

This service retrieves basic status information, without details on related transactions or additional properties. For performance reasons, it is *highly recommended to use this basic service* to load item details - it is the fastest one.

4.1.1. Request

```
GET {hostname}/api/{version}/payment-items/{PID}

eg
GET https://dolphin.pay-nxt.com/api/v1/payment-items/e63a8fa6-c7e0-41c3-9785-e159823464e1
```

4.1.2. Response

For Payment Items with a fixed amount:

```
{
  "id": "e63a8fa6-c7e0-41c3-9785-e159823464e1",
  "status": {
    "state": "SUSPENDED",
    "payable": false,
    "suspendReason": "PAID",
    "paymentDetailedState": "REGISTERED"
  },
  "amount": {
    "currency": "EUR",
    "fixed": 5.20,
    "collected": 0
  },
  "description": "Invoice 2016 - eni",
  "created": "2017-12-12T09:11:39.441Z",
  "lastUpdate": "2017-12-12T09:11:39.441Z",
  "expiryDate": "2019-12-12",
  "dueDate" : "2019-11-11"
}
```

For Payment Items with an adjustable amount:

```
{
  "id": "e63a8fa6-c7e0-41c3-9785-e159823464e1",
  "status": {
    "state": "OPEN",
    "payable": true
  },
  "amount": {
```

```

    "currency": "EUR",
    "adjustable": {
      "default": 5,
      "options": [
        {
          "value": 1
        },
        {
          "value": 5
        },
        {
          "value": 10
        }
      ],
      "range": {
        "minimum": 1,
        "maximum": 100
      }
    },
    "collected": 0
  },
  "description": "Invoice 2016 - eni",
  "created": "2017-12-12T09:11:39.441Z",
  "lastUpdate": "2017-12-12T09:11:39.441Z",
  "expiryDate": "2019-12-12",
  "dueDate" : "2019-11-11"
}

```

Attribute Name	Description
id	The unique identifier of the payment item as returned at item initiation Type: UUID
status	The current status of the payment item Type: JSON
state	Specifies the state in the lifecycle of a payment item (cfr supra). Possible values: • OPEN - The item is still to be paid • PAID - The item is paid • ALMOST_DUE - The item is nearing the due date, and should be paid soon • PAST_DUE - The item is overdue but still payable. (Merchant can decide to update the amount and charge overdue penalty). • SUSPENDED - It is no longer possible to pay the item. It has been cancelled. • EXPIRED - The item has expired (past its expiry date or is unable to proceed) and is no longer payable. This state is final. • SUBMITTED⁶ - A payment for the item has been submitted for further processing. It is still possible to pay (again) for the item. • UNKNOWN - No item with the requested ID could be found; either the request input was wrong or the item has been deleted (during a cleanup x months after having reached an end-state). Used in the statusText and statusImage.

⁶ Currently only used for SEPA Direct Debit

	Type: Text
payable	Indicates whether the payment item is still to be paid Can be used to decide whether to show a payment button or not - without having to interpret the value of state. Type: Boolean
suspendReason	The state given by the merchant. Possible values: SUSPENDED, PAID, null. Type: Text
paymentDetailedState	The summary state of the payment by OCS (does <u>not</u> take into account the suspendReason). Possible values (in order of ascending precedence): • NONE (no transactions) • FAILED • OPEN • REGISTERED (specialization of OPEN): created at UP • SUBMITTED (specialization of OPEN): forwarded to PSP • SUCCEEDED The value taken will be the one with the highest precedence amongst all transactions. For example: SUBMITTED is more relevant than previous or later FAILEDs. Type: Text
amount	The amount of the payment item, both the values specified at item initiation (fixed or adjustable) and the amount actually collected. Ico fixed amounts, the fixed Ico adjustable amounts, the collected value holds the actual amount paid.
currency	The currency of the payment item
fixed	Only returned ico a Fixed Amount Payment Item. Contains the amount specified at item initiation
adjustable	Only returned ico an Adjustable Amount Payment Item. Contains all options specified at item initiation.
collected	The amount actually collected. For items that are still <i>payable</i>, the value is zero. For <i>paid</i> items, the value specifies the actual collected amount which: - Ico Fixed amount Items, corresponds with the fixed amount - Ico Adjustable Amount Items, corresponds with the amount actually selected and paid by the Payer
description	The description of the payment item Type: Text
created	The date/time the payment item was initiated Type: Time Format: yyyy-mm-ddThh:mm:ss.mmmZ (ISO)
lastUpdate	The date/time the payment item was updated (eg paid) the last time Type: Time Format: yyyy-mm-ddThh:mm:ss.mmmZ (ISO)

expiryDate	The date the payment item expires Type: Date Format: yyyy-mm-dd
dueDate	The date the payment item is due (optional) Type: Date Format: yyyy-mm-dd

4.2. Extended Payment Item Status

This endpoint extends the basic payment item status: it provides extra details of transactions and adds some item properties.

If no transactions are linked to the payment item, the transactions array will be empty.

4.2.1. Request

```
GET {hostname}/api/{version}/payment-items/{PID}/details

eg
GET https://dolphin.pay-nxt.com/api/v1/payment-items/e63a8fa6-c7e0-41c3-9785-e159823464e1/details
```

4.2.2. Response

```
{
  "id": "9d9112bb-875f-4e6a-b9bf-3058504d3667",
  "status": {
    "state": "PAID",
    "payable": false,
    "suspendReason": "SUSPENDED",
    "paymentDetailedState": "SUCCEEDED"
  },
  "amount": {
    "currency": "EUR",
    "fixed": 1.03,
    "collected": 2.06
  },
  "description": "Test Invoice",
  "created": "2022-05-05T09:11:39.441Z",
  "lastUpdate": "2022-05-05T10:30:05.514Z",
  "expiryDate": "2023-02-05",
  "dueDate": "2023-01-05",
  "reconciliationReference": "12379677",
  "reconciliationReferenceType": "unstructured",
  "transactions": [
    {
      "id": "5c9eaf04-2d7b-4cfd-bf07-a44b1d8a3e5a",
      "state": "SUCCESS",
      "paymentDetailedState": "SUCCEEDED",
      "product": "CARDS",
      "created": "2022-05-05T09:21:27.545Z",
      "lastUpdate": "2022-05-05T09:23:27.545Z",
      "refund": false,
      "amount": {
        "currency": "EUR",
        "value": 1.03
      }
    }
  ]
}
```

```

{
  "id": "3df5d1b8-0263-48a3-9128-d93a8944a09c",
  "state": "SUCCESS",
  "paymentDetailedState": "SUCCEEDED",
  "product": "CARDS",
  "created": "2022-05-05T10:36:48.208Z",
  "lastUpdate": "2022-05-05T10:38:48.208Z",
  "refund": true,
  "amount": {
    "currency": "EUR",
    "value": 0.03
  }
},
{
  "id": "511f14fc-c713-4f8c-aa35-49d2291b0570",
  "state": "FAILURE",
  "paymentDetailedState": "FAILED",
  "failureReason": {
    "type": "pxg.error.code.bank.intiation.failed",
    "title": "Starting the SEPA bank transfer has failed.",
    "rejections": [
      {
        "code": "INSUFFICIENT_FUNDS",
        "title": "Amount of funds available to cover specified message amount
is insufficient.",
        "detail": "The payment could not be completed because of insufficient
funds"
      }
    ],
    "rejectionCategory": "BANK_SPECIFIC_ERROR"
  },
  "product": "PXG_PAY",
  "created": "2022-11-24T12:24:04.687Z",
  "lastUpdate": "2022-11-24T12:30:05.069Z",
  "refund": false,
  "amount": {
    "currency": "EUR",
    "value": 200000000.00
  }
}
]
"customMetaData": {
  "documentID" : "FFH_273589298736",
  "customerID" : "77365423393",
}
}

```

Attribute Name	Description
reconciliationReference	The reconciliationReference assigned by the Merchant to this payment item at the time of item initiation Type: Text
reconciliationReferenceType	The type of reconciliationReference assigned - see chapter on Reconciliation References in Merchant Implementation Manual for all details Type: Text
customMetaData	The custom metadata assigned by the Merchant to this payment

	item at the time of item initiation - contains a list of Key/Value pairs Type: JSON Object
transactions	The transactions that have been executed on this payment item together with their status and main properties - no details
id	The unique identification of the transaction Type: UUID
state	The state of the transaction Possible values: • OPEN - The transaction has not been completed yet or we don't have information on the final state. • SUCCESS - The transaction has been successfully completed, i.e. the payment for the transaction was successful. This state is final. • FAILURE - The transaction has failed. This state is final. • SUBMITTED⁷ - The transaction has been submitted for further processing. This state is final. Type: Text
paymentDetailedState	The state of the transaction by OCS which is a specialization of the state field. Possible values (in order of ascending precedence): • FAILED • OPEN • REGISTERED (specialization of OPEN): created at UP • SUBMITTED (specialization of OPEN): forwarded to PSP • SUCCEEDED Type: Text
product	The (payment) product used during the transaction. Type: Text
created	The date-time the transaction was executed Type: Time Format: yyyy-mm-ddThh:mm:ss.mmmZ (ISO)
refund	Indicates whether the transaction was a refund. Type: Boolean
amount	The amount of the transaction Type: JSON Object
currency	The currency of the transaction Type: Text
value	The value of the transaction Type: Decimal

⁷ Currently only used for SEPA Direct Debit

4.3. Payment item transactions

This service returns the array of transactions linked to the payment item, including limited information such as state. An optional query parameter (state) can be used to limit the results based on transaction state.

The response is a subset of the data returned by the extended payment item status service (cfr above).

4.3.1. Request

```
GET {hostname}/api/{version}/payment-items/{PID}/transactions?state={success|failure}
```

eg

```
GET https://dolphin.pay-nxt.com/api/v1/payment-items/e63fa6-c7e0-41c3-9785-e15984e1/transactions
```

4.3.2. Response

```
[
  {
    "id": "3df5d1b8-0263-48a3-9128-d93a8944a09c",
    "state": "SUCCESS",
    "paymentDetailedState": "SUCCEEDED",
    "product": "CARDS",
    "created": "2022-05-05T10:36:48.208Z",
    "lastUpdate": "2022-05-05T10:38:48.208Z",
    "refund": true,
    "amount": {
      "currency": "EUR",
      "value": 0.03
    }
  },
  {
    "id": "be557605-5a60-4d42-80ec-3a46d25c5427",
    "state": "SUCCESS",
    "paymentDetailedState": "SUCCEEDED",
    "product": "CARDS",
    "created": "2022-05-05T10:29:51.168Z",
    "lastUpdate": "2022-05-05T10:38:48.208Z",
    "refund": false,
    "amount": {
      "currency": "EUR",
      "value": 1.03
    }
  },
  {
    "id": "511f14fc-c713-4f8c-aa35-49d2291b0570",
    "state": "FAILURE",
    "paymentDetailedState": "SUCCEEDED",
    "failureReason": {
      "type": "pxg.error.code.bank.intiation.failed",
      "title": "Starting the SEPA bank transfer has failed.",
      "rejections": [
        {
          "code": "INSUFFICIENT_FUNDS",
          "title": "Amount of funds available to cover specified message amount is insufficient."
        }
      ]
    }
  }
]
```

```

        "detail": "The payment could not be completed because of insufficient
funds"
      },
      "rejectionCategory": "BANK_SPECIFIC_ERROR"
    },
    "product": "PXG_PAY",
    "created": "2022-11-24T12:24:04.687Z",
    "lastUpdate": "2022-11-24T12:30:05.069Z",
    "refund": false,
    "amount": {
      "currency": "EUR",
      "value": 200000000.00
    }
  }
]

```

4.4. Payment transaction details

More transaction details can be requested through the Payment Transaction Details service. It returns the available transaction data including state extended with available additional properties.

The properties that are available for a specific transaction mainly depend on the payment product used in the transaction.

Non exhaustive examples for different payment products can be found below.

4.4.1. Request

```

GET {hostname}/api/{version}/payment-transactions/{transactionID}

eg
GET https://dolphin.pay-nxt.com/api/v1/payment-transactions/e63fa6-c7e0-41c3-9785-e15984e2

```

4.4.2. Response

4.4.2.1. iDEAL

An iDEAL transaction contains a transactionExternalReference property.

```

{
  "id": "5c9eaf04-2d7b-4cfd-bf07-a44b1d8a3e5a",
  "state": "SUCCESS",
  "paymentDetailedState": "SUCCEEDED",
  "product": "IDEAL_COLLECT",
  "created": "2017-12-12T09:21:27.545Z",
  "lastUpdate": "2017-12-12T09:23:27.545Z",
  "refund": false,
  "amount": {
    "currency": "EUR",
    "value": 1.03
  }
  "properties": {
    "transactionExternalReference": "0020000094246211"
  }
}

```

```
}
```

4.4.2.2. SEPA Direct Debit

SEPA Direct Debit transaction contains a batchId property.

```
{
  "id": "5c9eaf04-2d7b-4cfd-bf07-a44b1d8a3e5a",
  "state": "SUBMITTED",
  "paymentDetailedState": "SUBMITTED",
  "product": "SEPA_DIRECT_DEBIT",
  "created": "2017-12-12T09:21:27.545Z",
  "lastUpdate": "2017-12-12T09:23:27.545Z",
  "refund": false,
  "amount": {
    "currency": "EUR",
    "value": 1.03
  }
  "properties": {
    "batchId": "20220210-sddc-1234567890abcdef"
  }
}
```

4.4.2.3. PXG

Failed transaction containing the failureReason details.

```
{
  "id": "511f14fc-c713-4f8c-aa35-49d2291b0570",
  "state": "FAILURE",
  "paymentDetailedState": "FAILED",
  "failureReason": {
    "type": "pxg.error.code.bank.intiation.failed",
    "title": "Starting the SEPA bank transfer has failed.",
    "detail": "Starting the SEPA bank transfer has failed.",
    "rejections": [
      {
        "code": "INSUFFICIENT_FUNDS",
        "title": "Amount of funds available to cover specified message amount is insufficient.",
        "detail": "The payment could not be completed because of insufficient funds"
      }
    ],
    "rejectionCategory": "BANK_SPECIFIC_ERROR"
  },
  "product": "PXG_PAY",
  "created": "2022-11-24T12:24:04.687Z",
  "lastUpdate": "2022-11-24T12:30:05.069Z",
  "refund": false,
  "amount": {
    "currency": "EUR",
    "value": 200000000.00
  },
  "properties": {
    "transactionExternalReference": "d22b7aec-cfb2-42e3-ad6f-c8f4aabf0479"
  }
}
```

4.5. Batch Payment Item Status

This service is an extension of the *Basic Payment Item Status* service and allows retrieving status information for a set of payment items.

4.5.1. Service Endpoint

The request should be **POST**'ed to:

```
{hostname}/api/{version}/payment-items/batch
```

4.5.2. Request

The Request Body is a sequence of 1 to 20 Items (grouped by the **paymentItems** element), and an indication which type of identifier is used: the item public identifier or the merchant item identifier (see Chapter 11 on the use of Item identifiers)

```
{
  "itemIdentifierType" : "INTERNAL",
  "paymentItems": [ "e63a8fa6-c7e0-41c3-9785-e159823464e1",
                    "65b25ad4-c7e0-41c3-9785-a256db8695fa",
                    "8562fd45-c7e0-41c3-9785-efda5216bc65",
                    "1d22c43a-1395-4d16-99be-6d8d3dc0b6c9",
                    "55ca02c2-812c-4d58-9c3a-f902fa72f799" ]
}
```

4.5.2.1. Request attributes

Attribute Name	Description
itemIdentifierType	The type of identifier that will be used to identify the items. When identifierType = INTERNAL (or not present), OCS will treat the content the paymentItems list as a list of item public identifiers otherwise as a list of merchantItemIdentifiers. Type: Text Valid Values: - INTERNAL - EXTERNAL Required: No Default if not provided: INTERNAL
paymentItems	An array of items - only an identifier needs to be provided for each item. The identifier can be either the item public identifier or the merchant item identifier.

4.5.3. Response

The response contains an array of items with their status information and the [item-public-identifier](#).

(response body = the one from the existing Basic Item Status Request, repeated for all Payment Items listed in the Request)

```
[
  {
    "id": "e63a8fa6-c7e0-41c3-9785-e159823464e1",
    "status": {
      "state": "OPEN",
      "payable": true,
      "suspendReason": "SUSPENDED",
      "paymentDetailedState": "REGISTERED"
    },
    "amount": {
      "currency": "EUR",
      "fixed": 522.22,
      "collected": 0
    },
    "description": "Invoice 2016 - eni",
    "created": "2017-12-12T09:11:39.441Z",
    "lastUpdate": "2017-12-12T09:11:39.441Z",
    "expiryDate": "2019-12-12",
    "dueDate" : "2019-11-11"
  },
  {
    "id": "65b25ad4-c7e0-41c3-9785-a256db8695fa",
    "status": {
      "state": "OPEN",
      "payable": true,
      "suspendReason": null,
      "paymentDetailedState": "SUBMITTED"
    },
    "amount": {
      "currency": "EUR",
      "fixed": 25.00,
      "collected": 0
    },
    "description": "Order 123",
    "created": "2018-03-31T09:11:39.441Z",
    "lastUpdate": "2018-04-10T09:11:39.441Z",
    "expiryDate": "2019-03-31"
  }
,
  {
    "id": "8562fd45-c7e0-41c3-9785-efda5216bc65",
    "status": {
      "state": "PAID",
      "payable": false,
      "suspendReason": null,
      "paymentDetailedState": "SUCCEEDED"
    },
    "amount": {
      "currency": "EUR",
      "fixed": 125.00,
      "collected": 125.00
    },
    "description": "Invoice 201700052",
    "created": "2018-02-28T09:11:39.441Z",
    "lastUpdate": "2018-03-05T09:11:39.441Z",
    "expiryDate": "2019-02-28"
  }
,
  {
    "id": "1d22c43a-1395-4d16-99be-6d8d3dc0b6c9",
```

```

    "status": {
      "state": "OPEN",
      "payable": true,
      "suspendReason": null,
      "paymentDetailedState": "SUBMITTED"
    },
    "amount": {
      "currency": "EUR",
      "adjustable": {
        "default": 5,
        "options": [
          {
            "value": 1
          },
          {
            "value": 5
          },
          {
            "value": 10
          }
        ],
        "range": {
          "minimum": 1,
          "maximum": 100
        }
      },
      "collected": 0
    },
    "description": "Invoice 201700052",
    "created": "2018-02-28T09:11:39.441Z",
    "lastUpdate": "2018-03-05T09:11:39.441Z",
    "expiryDate": "2019-02-28"
  },
  {
    "id": "55ca02c2-812c-4d58-9c3a-f902fa72f799",
    "status": {
      "state": "PAID",
      "payable": true,
      "suspendReason": null,
      "paymentDetailedState": "SUCCEEDED"
    },
    "amount": {
      "currency": "EUR",
      "adjustable": {
        "default": 5,
        "options": [
          {
            "value": 1
          },
          {
            "value": 5
          },
          {
            "value": 10
          }
        ],
        "range": {
          "minimum": 1,
          "maximum": 100
        }
      },
      "collected": 2
    },
    "description": "Invoice 201700052",
    "created": "2018-02-28T09:11:39.441Z",
    "lastUpdate": "2018-03-05T09:11:39.441Z",
    "expiryDate": "2019-02-28"
  }

```

```
}  
]
```

5. Payment Item Update

5.1. Introduction

5.1.1. Item Status Update

Broadly speaking, in billing applications, two main flows are involved:

1. A **Billing system** generates invoices (aka Payment Items'). Based on these invoices, online payments are initiated.
2. An **A/R system** has the most complete view on which invoices (Payment Items) are fully paid. The A/R system is able to generate feeders that allow OCS to enrich/maintain the payment statuses.

In practice, the above systems often operate independently. They typically don't know about each other, so we have to make sure that a specific Item is identified in both flows by the same natural key.

In the **Item Update service** (described below), the Item is referenced through the '**itemIdentifier**' attribute.

In the **Item Initiation service**, the same value must be added to the request body as the '**merchantItemIdentifier**' attribute:

```
{  
  "items": {  
    "1": {  
      "expiryDate": "2017-06-17",  
      "paymentAttributes": {  
        "amount": 50.12,  
        "description": "Invoice March 2016",  
        "reconciliationReference": "+++234/2334/37873++",  
        "dueDate": "2017-08-28",  
        "merchantItemIdentifier": "OI0000201700002877"  
      }  
    }  
  }  
}
```

5.1.2. Other Item Update use cases

In some cases, Merchants may want to update the amount of a payable item.

Some examples:

- When the due date has passed, merchants may want to add a late payment penalty.
- The Merchant may want to change the options offered to his customers (ico adjustable amounts)

Therefore, the Item Update Service also allows to update the amount and the description.

5.2. *Item Update Service*

5.2.1. Service Endpoint

Item updates can be **POST**'ed to:

```
https://{hostname}/api/{version}/item-updates
```

5.2.2. Request Body

The Request:

```

{
  "paymentItems": [
    {
      "itemIdentifier": "OI0000201700002877",
      "properties": {
        "status": "paid"
      }
    },
    {
      "itemIdentifier": "OI0000201700002898",
      "properties": {
        "status": "paid"
      }
    },
    {
      "itemIdentifier": "OI0000201700002914",
      "properties": {
        "status": "suspended"
      }
    },
    {
      "itemIdentifier": "OI0000201700002992",
      "properties": {
        "status": "open",
        "description": "Services provided in May 2017",
      }
    },
    {
      "itemIdentifier": "OI0000201700002877",
      "properties": {
        "amount": {
          "currency": "EUR",
          "fixed": 52.5 },
        "description": "50EUR + 2.5EUR late payment penalty"
      }
    },
    {
      "itemIdentifier": "OI0000201700003252",
      "properties": {
        "description": "Donation options",
        "amount": {
          "currency": "EUR",
          "adjustable": {
            "default": 10,
            "options": [
              {"value": 5.00,
                "description": "Pencils"},
              {"value": 10.00,
                "description": "Chocolates"},
              {"value": 25.00,
                "description": "Wine"}
            ]
          }
        }
      }
    }
  ]
}

```

The array is named **paymentItems**⁸ and enumerates a list of key/value objects.

itemIdentifier	The value for this element should match the value of merchantItemIdentifier in the original Item Initiation request. Format mask: AN {1,50}
status	This element defines the new status of the Item. In case of Payment Items, the allowed values are: • open The Item is 'not yet paid' from the merchant's perspective. It is only useful to reset the status from 'suspended' or 'paid' (in case of a status correction for example). Normally this status does not need to be used. • paid The Item is considered 'paid' using a channel other than Unifiedpost Payments. If it was paid through Unifiedpost Payments, then it is not necessary to set this since the item is already Paid. Browsing the Item will result in a message indicating that this particular item has been paid already.⁹ • suspended The system should no longer accept payments for this Item. The Checkout page will indicate that it is no longer possible to pay this item online. If Unifiedpost Payments consider the item 'paid' then an error will be returned.
amount	This element defines the new amount of the Item. This can be either a fixed amount or an adjustable amount (see above for how to defined adjustable amounts)
description	This element defines the new description of the Item.

When the **itemIdentifier** is not (yet) known to the system, this means that the Item has not (yet) been initiated. Such updates will be ignored and the platform will report the Item as **open** ('not yet paid'). A failure will be reported for this item.

When **status** is not recognised as one of the allowed values, the entry will be ignored.

Items Status updates are fairly flexible. Item Status can be set to any of the allowed states (**open** or **paid** or **suspended**) in basically any order, but only if following conditions are met:

1. Item Status can be updated only, as long as *the Item is not paid online through the platform*.
In other words: If an Item arrives at the **paid** state, because it was paid online, the **paid** state is final and cannot be updated anymore.
2. Item Status can be updated only, as long as *the lifetime of the Item is not expired*.
Remember that at Item Initiation time, an *expiry date* is fixed for the Item. The expiry date is

⁸ The example treats Status Updates for *Payment* Items. *Mandate* Items on the other hand, are held by a "mandateStatus" array. Mandate Items are not in scope of this document.

⁹ When a specific Item is paid online, *through the platform*, the Confirmation Page will show a number of relevant details like a transaction timestamp and a unique item id.

When it is paid *externally*, this information is not available. As a result, the Confirmation page will be more compact.

provided through the **expiryDate** attribute. Or, when **expiryDate** has not been provided, the system fixes the default expiry date to 9 months after creation date (or 9 months after the Due Date if the latter was provided).

3.

The table below summarizes the Update API behaviour for the different states:

Item Status in the Platform	Item Status update requested through the API by the Merchant	Is Status updated? Is Error reported? (¹⁰)	How the status is presented to the Payer - in status icon and Confirmation Page	Is the item still Payable?
OPEN ALMOST_DUE PAST_DUE	open	Updated	Payable (Almost Due, Past Due as appropriate)	Yes
	suspended	Updated	Not payable	No
	paid	Updated	Paid (no details)	No
EXPIRED	open	Ignored	Not payable	No
	suspended	Ignored	Not payable	No
	paid	Ignored	(hidden)	No
PAID	open	Ignored if paid through trx (1) Else (2) Updated	(1) Paid (2) Open	(1) No (2) Yes
	suspended	Error Reported	Paid	No
	paid	Ignored	Paid	No
SUSPENDED	open	Updated	Payable	Yes
	paid	Updated	Paid	No
	suspended	Ignored	Not payable	No

¹⁰ The Update API only deliberately reports errors in case a PAID item is set to SUSPENDED. Other update operations may be ignored but are not reported as errors.

SUBMITTED	open	Ignored	Submitted	Yes
	paid	Updated	Paid	No
	suspended	Updated	Not payable	No

5.2.3. Protocol

When posting Status updates to the platform, the platform will synchronously respond with an [HTTP 202 Accepted](#) message. This response includes a [Location header](#), indicating where to find processing feedback.

Querying this feedback URI (through HTTP **GET**), provides more information about the Status Update job. Jobs are scheduled so querying immediately or very frequently is not useful and may cause the Firewall to block requests from this IP address.

When the job is not yet completed, the platform will respond:

```
{
  "processingStatus": "pending"
}
```

A job can remain in pending state for several minutes.¹¹

After a job is completed, the platform will provide more detailed feedback on the update:

```
{
  "processingStatus": "completed",
  "processingDetails": {
    "paymentItems": {
      "successfullyUpdated": 231,
      "failures": [
        {
          "itemIdentifier": "OI0000201700002897",
          "reason": "unknown status"
        },
        {
          "itemIdentifier": "OI0000201700002898",
          "reason": "item not updatable"
        },
        {
          "itemIdentifier": "OI0000201700002899",
          "reason": "unknown identifier"
        }
      ]
    },
    "mandateItems": {
      "successfullyUpdated": 0,
      "failures": []
    }
  }
}
```

¹¹ At the time of writing, Item Update jobs are executed every 30 minutes.

Processing feedback remains available for a limited amount of time only. After 15 days, the platform will respond with an [HTTP 404 Not Found](#) message.

There is a limit of 2500 failures. If this is exceeded the last failure message will be “**Failure limit exceeded. All updates rejected!**”, in this case no payment items will be updated.

5.3. Usage Guidelines

Before calling the Item Update API, please take into account following guidelines:

- Make sure to submit Status Updates only for Items that have been initiated with a valid **merchantItemIdentifier** attribute. Only then, the platform can apply the requested update. In case the **itemIdentifier** is not recognised, an **unknown_identifier** will be reported.
- If the platform does not know the Item Status (i.e. because it has been initiated in stateless mode), the Status is considered **open**. Please do not send **open** statuses for Items that have not yet been paid (and are not suspended). These entries will be ignored, and the API may ignore the entire request.
- Please do not repeat the **paid** status for paid items. If in doubt, it is ok to (re-)send a **paid** status, but consecutive updates to **paid**, will result in the platform ignoring the entire request.
- Status Update requests will be processed by the platform as soon as possible, and always within 24 hours.
- Please take into account following usage limits:
 - a. Maximum 100 Status Update requests can be posted per 2 hours:
 - by a Merchant/Biller using his/her Merchant API key
 - by a Platform using its Partner API key on behalf of any Merchant
 - b. If a request generates more than 2500 failures the entire request will be rejected and no status updates will be made.

6. Short Checkout URL

6.1. Introduction

In some use cases, a Shorter Checkout URL than the standard one might be useful. Eg to send a reminder to the payer by sms including the checkout URL.

A standard checkout URL looks like:

<https://tx.pay-nxt.com/payment-checkouts/0252d7e8-26c0-4191-9c83-3571ae31e23c>
With 0252d7e8-26c0-4191-9c83-3571ae31e23c = the UUID or item public identifier (36 pos)

The Short Checkout URL service will return a short checkout URL of **35** positions and will look like:

<https://tx.pay-nxt.com/c/126c0419>

With **126c0419** being a short UID of 8 positions - generated upon this request

6.2. Request

```
GET https://{hostname}/api/{version}/payment-items/{public_identifier}/short-url
```

eg

```
GET https://dolphin.pay-nxt.com/api/v1/payment-items/807594ae-d9c0-4ae5-ad81-166eca6e81a7/short-url
```

6.3. Response

```
{  
  "response": "https://tx.pay-nxt.com/c/807594ae"  
}
```

7. Templates & Campaigns

7.1. Introduction

By design, an Item can be paid *only once*.

In some use cases however, we may want to publish a single URL (or QR code) that can be paid *multiple times*. For this purpose, Templates have been introduced.

A **Template** is similar to a **Payment Item**. The difference is that a *single Template* will 'hatch' *one or more Payment Items*.

7.2. How to register a Template?

The idea is to register a **Template**, containing the same set of attributes as for a regular **Payment Item**. Once registered, it can be used to initiate checkout pages 'on the fly' - not in context of one single Payment Item, but in the context of a **Campaign**.

Instead of posting an Item, register a Template:

```
POST https://{hostname}/api/{version}/templates
eg
POST https://dolphin.pay-nxt.com/api/v1/templates
```

7.2.1. Request

The Request Body consist of a single Item. The same validation rules as in Item Initiation apply. Notice though, it concerns just one *single* Item: no need for an index. The reason why it can be exactly the same JSON structure, is because, in the next step, it will be taken as-is to create Payment Items on the fly:

```
{
  "expiryDate": "2017-06-17",
  "paymentAttributes": {
    "amount": 50.12,
    "description": "Invoice March 2016",
    "reconciliationReference": "+++234/2334/37873+++",
    "dueDate": "2017-08-28",
    "merchantItemIdentifier": "010000201700002877"
  }
}
```

... two attributes are **not** allowed though:

- **merchantItemIdentifier**. This would cause conflicts across different Items in a Campaign, as the **merchantItemIdentifier** has to be unique across Items.
- **dueDate**. This has no meaning in the context of a template.

As with other Merchant API endpoints, the Merchant API Key is required as an Authorisation Header.

7.2.2. Response

The Response Body contains the Template Public ID, as well as the Campaign link:

```
{
  "templateId": "56434888-ee13-4628-8829-6503225",
  "campaignLink": "https://tx.pay-nxt.com/campaign-checkouts/56434888-ee13-4628-8829-6503225"
}
```

7.3. Campaign based Checkouts.

The equivalent of a Payment Checkout URL, is the Campaign URL:

```
https://tx.pay-nxt.com/campaign-checkouts/{item-template-public-identifier}
```

Similar to a Payment Item link, a Campaign link also returns a Checkout resource, but operates in two steps:

1. An Item is initiated dynamically, based on the template referenced by `{item-template-public-identifier}`
2. The checkout page for this new Item is generated.

8. Payment QR Codes

8.1. Introduction

There is a wide variety in QR codes. Consider following categorisation:

Target:

- A QR-Code, containing an HTTPS URL to the **OCS** Checkout Page
- A QR-Code, resulting in a **Bancontact** transaction, to be picked up by a Bancontact Payment app
- A QR-Code, ready to be consumed by the generic **iDEAL** Payment scanner

Cardinality:

- One QR code can be paid only once - typically in context of a given Item
- One QR code can be paid many times (actually initiating multiple Items on the fly)

8.2. Service Endpoint

OCS QR Codes are generated by **POST**'ing instructions to:

```
https://{hostname}/api/{version}/qr
```

i.e.

```
https://dolphin.pay-nxt.com/api/v1/qr
```

8.3. Request

The Request Body contains two mandatory attributes: **type** and **resourceIdentifier**.

Attribute Name	Description	Example Value
type	The type of QR code to be created Valid values: • PAYMENT_ITEM • PAYMENT_ITEM_TEMPLATE • PAYMENT_ITEM_BC_NATIVE • PAYMENT_ITEM_IDEAL_NATIVE Required: Yes	PAYMENT_ITEM
resourceIdentifier	The identifier of a Payment Item or a Template. Ico PAYMENT_ITEM_TEMPLATE - The identifier is the template uuid For the other QR types The identifier can be EITHER: - The payment item uuid; OR - The merchant item identifier	3f438cc5-46fb-4b16-aeb1-ad23eb93d146

	As specified in the identifierType Required: Yes (if not requested at item initiation - see §8.7) Having the item created before making this post request is a prerequisite, otherwise the urls and images will not work as expected.	
size	The size specifies the QR Image size in pixels and should be a value between • Min 100px • max 2000 px • Default: 250 px	250
	The type of resourceIdentifier used, can be either: - SYSTEM (default), OR - EXTERNAL	
properties	To be defined	

An example request body:

```
{
  "type" : "PAYMENT_ITEM",
  "resourceIdentifier" : "3f438cc5-46fb-4b16-aeb1-ad23eb93d146"
}
```

As with other Merchant API endpoints, the Merchant API Key is required as an Authorisation Header.

8.4. Response

In case of successful requests, the response status will be 201 - Created.

The response body has the same structure regardless of the specified type, but it will behave differently.

An example response:

```
{
  "qrContent": "https://tx.pay-nxt.com/qr-checkouts/88112542-5d49-435a-a96b-c48dd7127878",
  "qrImageUri": "https://tx.pay-nxt.com/qr-checkouts/88112542-5d49-435a-a96b-c48dd7127878/image",
  "qrImageBase64":
  "iVBORw0KGgoAAAANSUHEUgAAAPoAAAD6AQAACg12eQAAABsU1EQVR42u2ZUYrDMAxEBT6Aj+Sr+0g5gMHVjJQ0Lt39WPbDAw1po077EZ
  JmpNTm71e3B3iAB/gb0M2v0svEPY/m98CRaQH+GbWXw4a1IEfNcyHAg/ITs+bPJGu3qgj4eT0E2QdvTQABsvyak4pAlJwz/hP06vea3Bxg+
  2e08v6iD5sDb93NfNWftHpnAGEORuqNb4zRs8bECQEsmNSgfET++iWLAXgbi1WKcomUgLKTPkKZzcqMxVACJjsdeSoEKaaIXdaAEessEKv
  tIjUnyYFdLY8rISH/Wy1pgWECUKHYesTtk4F0AKyWewatJC7e5gaAK8SmQoFqPFVCIB3s0TCRCLMudiAJBdw8bPTSRqTwqAJ0KKXZPZNQj
  WM7WMKPsDPb+yxkra41qTEgCdsVy7LFP1McPsD8QCdQ1XmK0tWr0/QNVC42SmOLKU+7YoAOTC109G6Cnh+FJAarrSsNIvyGzm0KAExWeGF1d
  n1j0vJKQDj3Diw0lqkbK1JISAU00b2cs4tcsB7dKez30t0Agh/n+dbBePbhvFZk7sDV/uz50Dy3EGaFPD8G/UAD/DfwAsPPOCYpCGhXQAAA
  ABJRUS5ErkJggg==",
  "expiryDate": "2020-11-14T13:23:05.712Z"
}
```

Attribute Name	Description
----------------	-------------

qrContent	The qrContent URL will redirect to the payment checkout page where the transaction can be completed by scanning the qr code, or paid using 3DS protocol. Note Since the QR Content is not always available for an iDEAL QR code, this output is not returned for QR type PAYMENT_ITEM_IDEAL_NATIVE.
qrImageURL	The qrImageURL will provide the image of the QR code
qrImageBase64	The qrImageBase64 will also provide the image of the QR code, but encoded in Base64.
expiryDate	The expiry date of the QR Image. This will always be the same date as the expiry date of the payment item or item-template.

8.5. Different types of QR Code

type	Underlying Resource	Target	Cardinality
PAYMENT_ITEM	Payment Item	OCS Checkout Web Page	Only once
PAYMENT_ITEM_TEMPLATE	Template		Multiple
PAYMENT_ITEM_BC_NATIVE	Payment Item	Bancontact App and all Belgian Banking Apps	Only once
PAYMENT_ITEM_IDEAL_NATIVE	Payment Item	Dedicated iDEAL app and selected Dutch Banking Apps (ABN Amro, Rabobank and ING)	Only once

8.6. Add extra Info to the QR Image

In some cases - eg when QR codes are printed on paper - extra information added to the QR image may help identify which item this QR code refers to.

Therefore it is possible to add following query parameters to the **qrImageURL**:

?label=	The label specifies which text is added to the QR image. It corresponds with data defined for the linked payment item (template). Following label values are supported: • reconciliation-reference • description • qr-type The label parameter can be added more than once (max nr = nr of supported values)
----------------	--

Example:

```
https://tx.pay-nxt.com/qr-checkouts/20c9626f-79a0-4101-a86c-3a89f47f0aba/image?label=reconciliation-reference&label=description&label=qr-type
```

... will produce:



QR with labels
Test QR
PAYMENT_ITEM

8.7. Request QR Code at Item Initiation

As described above, retrieving QR codes for a Payment Item is typically done through an additional **qr** request, *after* Payment Initiation, returning *both* QR *content* and *images*. For high-volume work flows, such a second request may cause too much overhead. Not least, because the *image* rendering may be time consuming. Specialised print houses have dedicated QR image generation software, so there may be a preference to avoid QR Code generation by the platform.

To support these optimisations, OCS offers two features:

1. Get access to the QR details, immediately through the initial request that creates the Item. (which also allows for batch processing)
2. Distinguish between generating (1) just QR code *content*, or (2) also produce the QR code *images*.

To support all this at Item Initiation time, it's possible to add the **qrCode** as **paymentOption** in the Request body.

Example Request:

```
{
  "items": {
    "1": {
      "paymentAttributes": {
        "amount": 0.01,
        "description": "Test Batch QR 1",
        "reconciliationReference": "Test Batch QR",
        "merchantItemIdentifier": "TestQR456"
      },
      "paymentOptions": {
        "qrCode": {
          "type": "PAYMENT_ITEM",
          "size": 100
        }
      }
    }
  }
}
```

```

    }
  },
  "2": {
    "paymentAttributes": {
      "amount": 0.02,
      "description": "Test Batch QR 2",
      "reconciliationReference": "Test Batch QR"
    },
    "paymentOptions": {
      "qrCode": {
        "type": "PAYMENT_ITEM_BC_NATIVE",
        "size": 200
      }
    }
  }
}

```

Attribute Name	Description	Example Value
paymentOptions	Specifies additional input/output options for a Payment Item. Type: Array Required: No	
qrCode	Specifies what extra QR output to be generated at Payment Item Initiation. Possible attributes and values are the same as when requesting a QR Code stand alone (as specified above). Except for the resourceIdentifier which is not known at the time of initiation. Required: No	

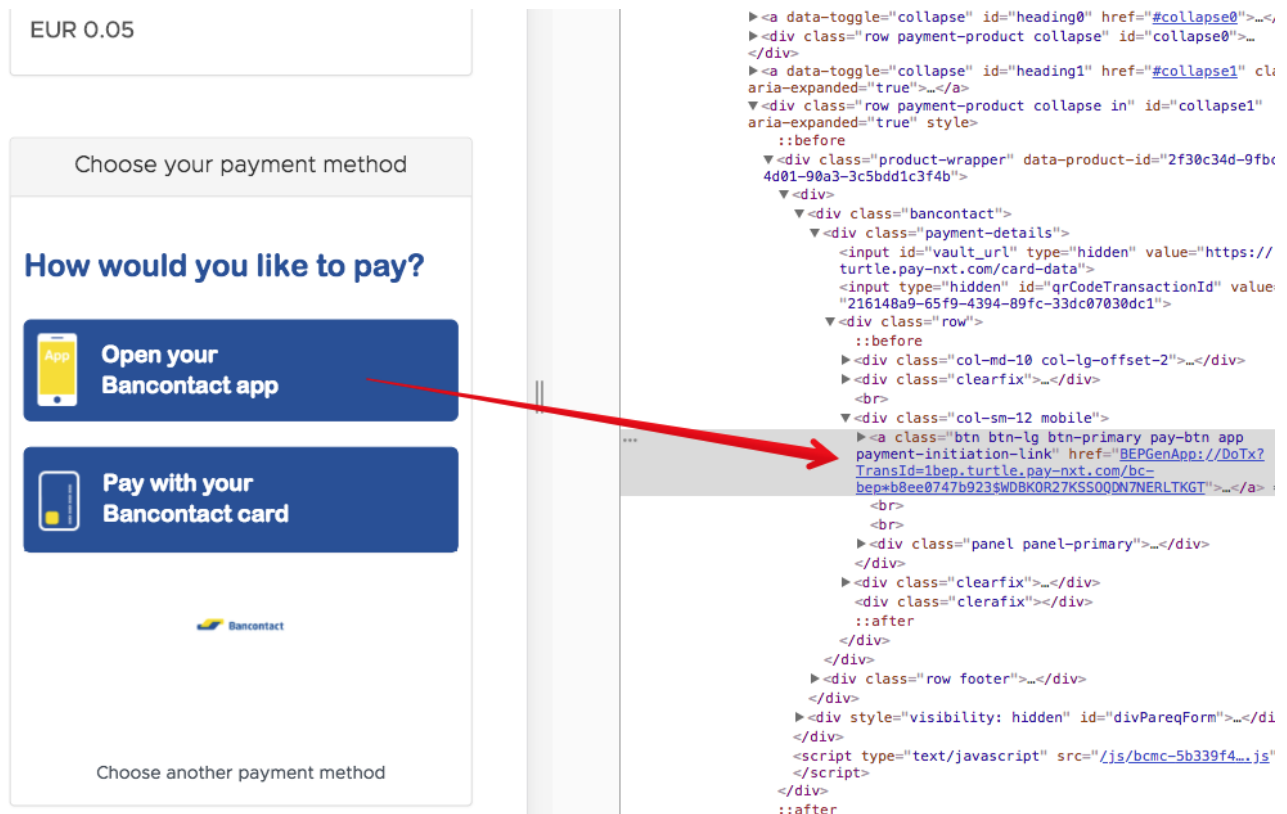
Example Response:

```
...
{
  "1": {
    "id": "8748e302-4a66-4fb5-8163-810cf1806781",
    "paymentLinks": {
      "checkout": "https://tx.test.pay-nxt.com/payment-checkouts/8748e302-xxx-810cf1806781",
      "statusText": "https://tx.test.pay-nxt.com/payment-items/8748e302-xxx-810cf1806781/status.txt",
      "statusImage": "https://tx.test.pay-nxt.com/payment-items/8748e302-xxx-810cf1806781/status.png"
    },
    "paymentOptions": {
      "qrCode": {
        "qrContent": "https://tx.test.pay-nxt.com/qr-checkouts/591c6a38-5677-4808-abcd-b811d79bc547",
        "qrImageUrl": "https://tx.test.pay-nxt.com/qr-checkouts/591c6a38-5677-4808-abcd-b811d79bc547/image",
        "qrImageBase64":
          "iVBORw0KGgoAAAANSUHEugAAAPoAAAD6AQAAAAACgl2eQAAABqE1EQVR42u2ZwY3EIAxFLVFASqJ1SqIAJK/9bbLZJNrDaA58iYiJAryLhf1tM6L/P002sIENfAY0sedQexWb9apdhi8JF2C/cTRjhlT7GPZx5DoRYEbZitlogM2ddJ4RaNhqBSdFC2j4mFkqODI6IFzO3axXt1TffXJxA0HfxnW86MPiQD62gsAxvr5q9dpAw5bp8LTXvnu1QQYg6hUr7nUWOD6ECNdTcVl16xAvxlzMpAFgoFuKk5KZGZmAMBUXp+axU3xwAQ05sYXLuQK4IMNqKiASit1lNs6EmDwRELFfkOUj/N33fl20BAiv8+JkSvHNTAYgypLwMXtH+JcuXMC1AFf+K1bKwycXByItFrSBm0qsGLkAPTsp1I165noiWKKXyO16gYxJSXG5avTww83t2hVCzevNJAiC10Jqpz0xycTk0AEX7tXSPKRUwG6a4IU65GJpABZwtreYWHC+uFYgAJBSd9VW05Gk1ETCj3s3Me548Mj4gM3vHdS5sZwSwW6NoH4eOx2EtDsxt+w74oJ9/JFiAiDCX0+Xe5buBMD+N2oDG/g28AP2LNEYh7E/4gAAAABJRUSErkJggg==",
        "expiryDate": "2019-05-23T07:02:46.276Z"
      }
    }
  },
  "2": {
    "id": "d8acc813-015f-41d0-95d5-7ef92928a7eb",
    "paymentLinks": {
      "checkout": "https://tx.test.pay-nxt.com/payment-checkouts/d8acc813-xxx-7ef92928a7eb",
      "statusText": "https://tx.test.pay-nxt.com/payment-items/d8acc813-xxx-7ef92928a7eb/status.txt",
      "statusImage": "https://tx.test.pay-nxt.com/payment-items/d8acc813-xxx-7ef92928a7eb/status.png"
    },
    "paymentOptions": {
      "qrCode": {
        "qrContent": "BEP://1bep.turtle.test.pay-nxt.com/bc-bep*4d331d5a37f9$B4QWWCGOX7PZYJU4U0V2WETJ",
        "qrImageUrl": "https://tx.test.pay-nxt.com/qr-checkouts/a0661e0e-25b9-44c8-8915-fb0cb691ebad/image",
        "qrImageBase64":
          "iVBORw0KGgoAAAANSUHEugAAAPoAAAD6AQAAAAACgl2eQAAACOE1EQVR42u2ZMw6FQAxEjSi25AjcZLkYEKj/YnCTPQL1FghnPesi4H+LSIOlQIEieEVMxjP2Rvt3a5IHeIAH+NeA4GpFm1F0a14aZl3t0eAKCKprG7ZGU+6WwQVa+1Z1cQZM0re505dmkTpJBwCUQ0A3GRJ+qjVXKMonEAYQCD3o7BLQ8kRa6HW02/pBMDcD7KwwQwU/tw+tdzPAS0y0aH/02MR3Vx07GZAKH1gimgpfwZP0i4jJwxVg9mn2NMBD1yH1CEvNR9F6AHKEHqBSsVLgAZv0oscqPABqzmQeIIGNZqUEbwD00rbZRIGvvEIKuJ1F6wDAu1ehvvV6yU0PAH5tqDRauEeOIWb+k/gCaE+7R7EA3RvNFQAVQAo2fMQihZVFuQLR3hnpIT2vdH5cveHBwAjvYT7hlLwFFd0BtgsnDBrwqMYSQS6Y/s7AKgHe0jI0dhys4ceZHAfMf5RSvEoy00UcRatAwATCFazoteGTQXLn5fBFcCpnUsQ14y9lEMVHGcakiV84kDMWeQUSR4AWyKxZJRZziakt3B3AiIYCuHRNnrinhgStTkVQ/YAFJS0zD0xc3K5/i9sBm0buUWXJgB70bu8B0H3xpUftpcAS1BXAnTcLVMCprpzDdNff/GagnB4wjX523nPeWd2w6JaQe0SeHY/n4AyyDoteczjpnIG4Am+VKEPII5v3I626Aeohl9WEkMUBVFVD0q7Hz2vBh59Xm+2/HbjcDzz84HuABHuAvwBeXJifNTL402QAAAAABJRUSErkJggg==",
        "expiryDate": "2021-02-11T07:02:46.307Z"
      }
    }
  }
}
...
```

9. Bancontact Mobile URL Intent

9.1. Introduction

To initiate a mobile Bancontact transaction from a mobile device, implies sending a URL Intent to a locally installed, Bancontact compatible app. The most common example is the URL Intent, embedded in the OCS Checkout page, displayed in a mobile browser.



The image shows a mobile checkout interface for Bancontact. At the top, it displays 'EUR 0.05'. Below this, a section titled 'Choose your payment method' contains the question 'How would you like to pay?'. There are two main options: 'Open your Bancontact app' (with an app icon) and 'Pay with your Bancontact card' (with a card icon). At the bottom, there is a 'Choose another payment method' link. A red arrow points from the 'Open your Bancontact app' button to the corresponding URL Intent in the HTML source code on the right. The source code snippet shows a button with the class 'btn btn-lg btn-primary pay-btn app payment-initiation-link' and a href attribute containing the URL Intent: `href="BEPGenApp://DoTx?TransId=1bep.turtle.pay-nxt.com/bc-bep*b8ee0747b923$wDBKOR27KSS00DN7NERLTKGT"`.

OCS offers an API to generate these URL Intents, so it becomes possible to initiate an app-to-app payment experience.

9.2. Service Endpoint

Bancontact URL Intents are generated by **POST**ing instructions to:

```
https://{hostname}/api/{version}/payment-items/{payment-item-public-Identifier}/bep-tokens
```

i.e.
`https://dolphin.pay-nxt.com/api/v1/payment-items/3f438cc5-46fb-4b16-aeb1-ad23eb93d146/bep-tokens`

9.3. Request

The Request Body described the redirect policy.

An example request body:

```
{
  "redirectURL" : "https://en.wikipedia.org/wiki/Home_page",
  "successURL" : "https://en.wikipedia.org/wiki/OK",
  "failureURL" : "https://en.wikipedia.org/wiki/Failure"
}
```

Attribute Name	Description	Example Value
redirectURL	redirectURL is sent in the first communication with Bancontact and it will be used in any outcome (success, failure or canceled by user) unless a more specific url is provided (successURL or failureURL) In case redirectURL is not provided the starfish status page will be used as default. Valid values: A URI to be resolved by the mobile device. This can be an HTML URL, resulting in a mobile webpage being shown. Or it can be a mobile deep link, pointing to a specific location in a particular mobile app. Required: At least one of redirectURL , successURL or failureURL must be present. The system will choose the most applicable one.	https://tx.pay-nxt.com/payment-items/2645505f-d0a4-41dc-b444-5ff51134ed75/status
successURL		or
failureURL		myApp://item/656257/successful-payment

As with other Merchant API endpoints, the Merchant API Key is required as an Authorisation Header.

9.4. Response

url contains the URI to be embedded in the mobile app.

```
{
  "url": "BEPGenApp://DoTx?TransId=1bep.turtle.pay-nxt.com/bc-bep*d6b7bd36f8a0$AA3XU3UR6RXRDSQFQOETLRUV",
  "expiryDate": "2018-03-06T09:15:41.449Z"
}
```

10. Payout Report Requests

10.1. Introduction

Items that have been paid are settled with the Merchant periodically.

After each payout to a Merchant, a Payout Report is generated, containing both information on the global payout and details on the individual items being part of that payout. For more details on this report, see companion document "[OCS - Merchant Implementation Guide](#)".

10.2. Authentication Header

The calls to request Payout Reports are protected by API Keys that are flagged for sensitive operations. Such an API Key will have to be created before being able to make use of the services described in this chapter.

See Section "[API Keys for access to sensitive resources](#)".

10.3. List of Payout Reports Service

10.3.1. Request

```
GET https://{hostname}/api/{version}/reports/payout
```

eg

```
GET https://dolphin.pay-nxt.com/api/v1/reports/payout
```

As a curl command:

```
curl -H "Authorization: OCS-API 083a77d4-051f-46ff-9850-940193f1239a" https://dolphin.pay-nxt.com/api/v1/reports/payout
```

10.3.2. Response

The response contains a list of URLs

```
"reportLinks": [
  {
    "reportNumber": 12,
    "reportDate": "2017-09-11",
    "reportURL": "https://dolphin.pay-nxt.com/api/v1/reports/payout/9d696b33-0dd1-4120-828b-127c51c25gf7"
  },
  {
    "reportNumber": 13,
    "date": "2017-09-12",
    "reportURL": "https://dolphin.pay-nxt.com/api/v1/reports/payout/9d696b33-0dd1-4120-828b-127c51c25gf5"
  }
]
```

Attribute Name	Description
reportLinks	An array that enumerates a list of payoutReportDetails Type: JSON Array
reportNumber	The unique Payout Number of this payout.

	In case of Balance Payout, this number is also shown in the remittance information of the payout. Type: Integer
reportDate	The Payout Report Date, ie the date of the payout reported (= 'the day the Pay-Out was initiated by Unifiedpost Payments') Type: Text Format: yyyy-mm-dd
reportURL	The URL at which the Merchant can retrieve the Payout Report. This URL is not exposed in the public domain. To GET this body, the appropriate API key needs to be provided in the Authorization header. (see above)

10.4. Payout Report Details Service

10.4.1. Request

There are two paths to fetch a Payout Report:

- A. Using the **Payout Report** [public-identifier](#) as a criteria:

```
GET https://{hostname}/api/{version}/reports/payout/{public-identifier}
```

eg

```
GET https://dolphin.pay-nxt.com/api/v1/reports/payout/9d696b33-0dd1-4120-828b-127c51c25gf7
```

As a curl command:

```
curl -H "Authorization: OCS-API 083a77d4-051f-46ff-9850-940193f1239a" https://dolphin.pay-nxt.com/api/v1/reports/payout/6943da71-11e9-4e86-badc-bfd65a34aaef
```

- B. Using the **Payout** [Payout Number](#) as a criteria

```
GET https://{hostname}/api/{version}/reports/payout/report-number/{report-number}
```

eg

```
GET https://dolphin.pay-nxt.com/api/v1/reports/payout/report-number/83
```

As a curl command:

```
curl -H "Authorization: OCS-API 083a77d4-051f-46ff-9850-940193f1239a" https://dolphin.pay-nxt.com/api/v1/reports/payout/report-number/83
```

10.4.2. Response

The response contains a file in the format specified, default being csv (layout details are explained in the [OCS - Merchant Implementation Guide](#))

```
HEAD,PAYOUT_REPORT,36,INDIVIDUAL_TRANSACTION,2017-12-29,OC80000001414,BE24363053252838,BBRUBEBB,PAYNXT
S.A.,BE97310072950049,BBRUBEBB,EUR,Sabam,Rue d'Arlon 75-77,1040 Brussels,BE,2,251.08
TRANSACTION_CREDIT,ba6ad670-599d-4135-917a-297767da83e3,192.39,192.39,EUR,2017-12-28,2017-12-
28T09:37:40.103Z,Anton Reygelton
Reygel,BE63457450245108,670342XXXXXX0020,,BANCONTACT_COLLECT,OGM_VCS,017027388686,,
TRANSACTION_CREDIT,85754570-9ebe-4f24-87e9-cbb021eb85a9,58.69,58.69,EUR,2017-12-28,2017-12-
28T07:18:15.054Z,NA,BE34001797312390,670304XXXXXX1723,,BANCONTACT_COLLECT,OGM_VCS,017027558438,,
```

10.5. Payout Report Format

Next to the default csv format, a Payout Report can also be requested in a CODA like format or a MT940 like format, following the CODA or MT940 standards, but for the 'Virtual Merchant Account' instead of a physical bank account. The layout of these files is described in the [OCS - Merchant Implementation Guide](#).

An optional query parameter can be used to specify the format in which Payout Reports should be returned. The default format is MPO (csv).

?format=	The format in which the Payout Report is returned. Can be one of these values: • MPO: default csv report is returned • CODA • MT940
-----------------	---

The parameter can be added to each of the Payout Requests:

```
GET https://{hostname}/api/{version}/reports/payout?format={format}

GET https://{hostname}/api/{version}/reports/payout/{public-identifier}?format={format}

GET https://{hostname}/api/{version}/reports/payout/report-number/{report-
number}?format={format}
```

Example:

Assuming the requested format is CODA:

```
GET https://{hostname}/api/{version}/reports/payout/{public-identifier}?format=CODA
```

The Response will be:

```
0000017022000005      0000000271Unifiedpost PAYMENTS NV  PANXBEB1  0000000000 00000      2
130000C40000000000      EUR00000000000000000160220Unifiedpost PAYMENTS NV  PAY-NXT virtual account  000
2100010000      0000000000000010170220001500001101133767115325      17022000001 0
2300010001BE47979623542980      EURNA      0 0
2100020000      1000000000000010170220001020000271      17022000001 0
2300020001BE25068890397682      EURUnifiedpost PAYMENTS NV      0 0
80000C40000000000      EUR000000000000000000170220      0
9      00000800000000000001000000000000010      2
```

Example:

Assuming the requested format is MT940:

GET <https://{hostname}/api/{version}/reports/payout/{public-identifier}?format=MT940>

The Response will be:

```
{1:F01PANXBEB1XXXX0000000000}{2:0940000000000000PANXBEB1XXXX000000000000000000N}{3:}{4:
:20:302
:25:0C400000000000
:28C:302/1
:60F:C200221EUR0,00
:61:2002210221C0,03N100NONREF
:86:/TRTP/SEPA TRANSFER/IBAN/NL84RAB00304935115/NAME/J.H.J. VAN B
ULCK/REMI/TestRelease127.3/EREF/1999b7fe-027f-483c-beb8-b68bf4ea6
9bd
:61:2002220222C0,03N100NONREF
:86:/TRTP/SEPA TRANSFER/IBAN/BE55068890432644/NAME/NA/REMI/TestRe
l_28/EREF/b55b820a-97b3-43d0-a079-235ac81ead5e
:61:2002220222C0,03N100NONREF
:86:/TRTP/SEPA TRANSFER/IBAN/BE55068890432644/NAME/jan van bulck/
REMI/TestRelease127.3/EREF/5c8047cb-103c-46d0-946b-5dbb20a964c0
:61:2002220222C0,03N100NONREF
:86:/TRTP/SEPA TRANSFER/IBAN/BE55068890432644/NAME/NA/REMI/TestRe
lease127.3/EREF/5fe9fbeb-88ed-48ca-b7e5-a6ea7f8d762a
:61:2002220222D0,12N100NONREF
:86:/TRTP/SEPA TRANSFER/IBAN/BE25068890397682/NAME/Unifiedpost PA
YMENTS NV/REMI/302
:62F:C200222EUR0,00
-}{5:}
```

11. Payment Item Identifiers

Payment items are usually identified during operations using the `item-public-identifier` (a `UUID`), returned after Item Initiation.

In some cases we support identifying the payment item using the `merchantItemIdentifier` property specified during initiation (*if* one was provided).

To do this, the caller will have to specify the `merchantItemIdentifier` as item identifier and append the following query parameter to the url:

<code>?identifier-type=EXTERNAL</code>	Indicates the identifier being specified as being a <code>merchantItemIdentifier</code>
--	---

Example:

Assuming a payment item has following properties:

- Item Public Identifier = "5b1b793e-448f-4827-865b-9f75f0ba5373"
- merchantItemIdentifier = "mer-123"

The following requests will result in the same outcome

```
GET {hostname}/api/{version}/payment-items/5b1b793e-448f-4827-865b-9f75f0ba5373
GET {hostname}/api/{version}/payment-items/mer-123?identifier-type=EXTERNAL
```

The api's which support this identification method are listed below:

- [Basic Payment Item Status](#)
- [Extended Payment Item Status](#)
- [Payment Item Transactions](#)
- [Short Checkout URL](#)
- [Bancontact Mobile URL Intent](#)
- [AddNotificationURL](#)

12. Refunds

12.1. Introduction

Merchants can refund (part of) a transaction to their payers. Currently this feature is supported for iDEAL, Bancontact and Cards.

Refunds are only possible:

- for the remainder of the amount, the part not refunded yet
- within 2 years after the original payment date
- only with the product/card/account used for the original transaction

Refunds can be done through the OCS Merchant Dashboard, and by calling the endpoint described below.

12.1. Endpoint

POST https://{hostname}/api/{version}/transaction-refunds/{transactionID}

eg

POST dolphin.pay-nxt.com/api/v1/transaction-refunds/5c9eaf04-2d7b-4cfd-bf07-a44b1d8a3e5a

12.2. Request

```
{
  "amount" : 5.00,
  "reason" : "Double payment"
}
```

Attribute Name	Description	Example Value
refundAttributes	Refund attributes	
amount	The amount to be refunded. The currency will always be equal to the original transaction currency. Required: No Default: If no amount is provided, the entire transaction amount will be refunded.	5.00
reason	A short textual description about the reason to refund the money, for traceability reasons. Type: Text Format: AN max 255 positions Required: Yes	Double payment

12.3. Error handling

In case of exceptions, a response of media type "**application/problem+json**" [RFC 7807] will be returned, with details about the problem.

Example:

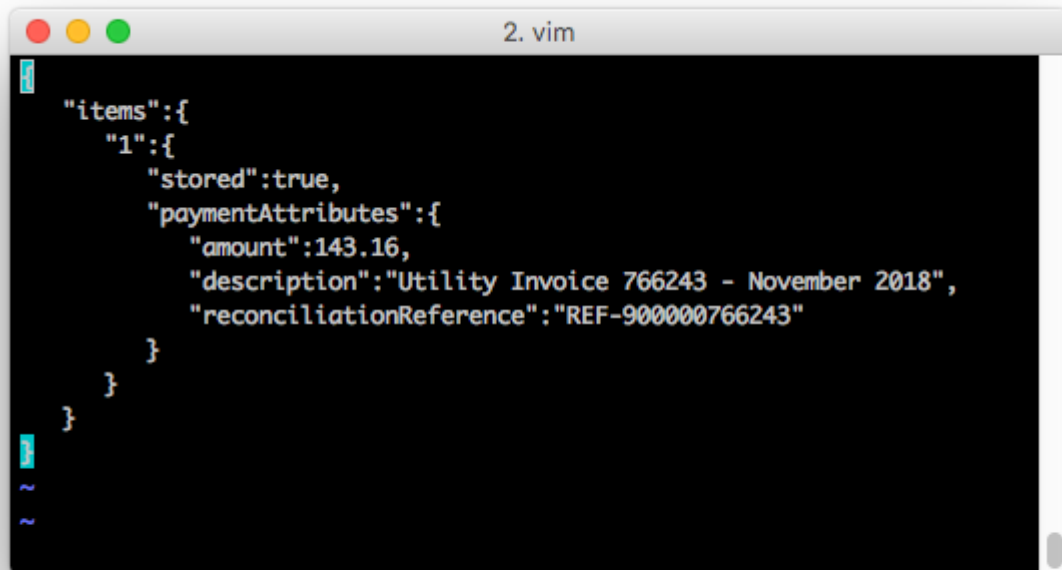
HTTP response code	Payload
400	<pre>"type": "transaction_mandatory", "title": "Transaction mandatory", "detail": "A refund always has to be linked to an existing transaction"</pre>
...	...

13. Appendix A. Hands-on examples using the API

12.1. CURL

12.1.1. Initiating a Payment Item

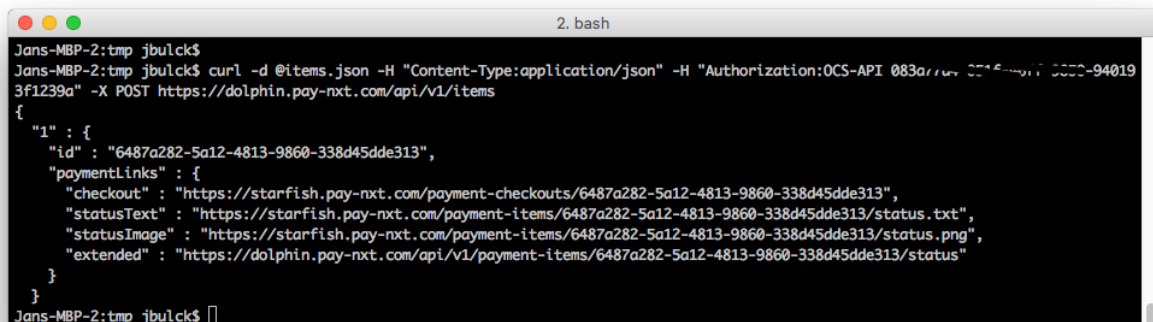
Create a small file called `items.json` and copy/paste the Payment Initiation body in there:



```
2. vim
{"items":{
  "1":{
    "stored":true,
    "paymentAttributes":{
      "amount":143.16,
      "description":"Utility Invoice 766243 - November 2018",
      "reconciliationReference":"REF-900000766243"
    }
  }
}}
```

Next, **POST** `items.json` to the OCS endpoint:

```
curl -d @items.json -H "Content-Type:application/json" -H "Authorization:OCS-API
083a77d4-xxxx-xxxx-9850-940193fxxxxx" -X POST https://dolphins.pay-
nxt.com/api/v1/items
```



```
2. bash
Jans-MBP-2:tmp jbulck$
Jans-MBP-2:tmp jbulck$ curl -d @items.json -H "Content-Type:application/json" -H "Authorization:OCS-API 083a77d4-xxxx-xxxx-9850-940193f1239a" -X POST https://dolphins.pay-nxt.com/api/v1/items
{"1": {
  "id": "6487a282-5a12-4813-9860-338d45dde313",
  "paymentLinks": {
    "checkout": "https://starfish.pay-nxt.com/payment-checkouts/6487a282-5a12-4813-9860-338d45dde313",
    "statusText": "https://starfish.pay-nxt.com/payment-items/6487a282-5a12-4813-9860-338d45dde313/status.txt",
    "statusImage": "https://starfish.pay-nxt.com/payment-items/6487a282-5a12-4813-9860-338d45dde313/status.png",
    "extended": "https://dolphins.pay-nxt.com/api/v1/payment-items/6487a282-5a12-4813-9860-338d45dde313/status"
  }
}}
```

12.1.2. Fetching the Payout reports

To get the list available Payout Reports:

```
curl -H "Authorization: OCS-API 083a77d4-051f-46ff-9850-940193f12999"
https://dolphin.pay-nxt.com/api/v1/reports/payout
```

This will return :

```
{
  "payoutReportLinks": [
    {
      "reportNumber": 99,
      "date": "2018-02-18",
      "reportURL": "https://dolphin.pay-nxt.com/api/v1/reports/payout/bfa1088c-a4a0-4950-8a3d-d242d2b36999"
    },
    {
      "reportNumber": 98,
      "date": "2018-02-15",
      "reportURL": "https://dolphin.pay-nxt.com/api/v1/reports/payout/850c645f-d009-47c9-a30e-937d6a00b999"
    },
    ...
    {
      "reportNumber": 83,
      "date": "2018-01-19",
      "reportURL": "https://dolphin.pay-nxt.com/api/v1/reports/payout/6943da71-11e9-4e86-badc-bfd65a34a999"
    },
    {
      "reportNumber": 82,
      "date": "2018-01-16",
      "reportURL": "https://dolphin.pay-nxt.com/api/v1/reports/payout/73b36380-1e57-4912-b61b-c3f1df43f999"
    }
  ]
}
```

Then, you can either query using the URL provided in the payout report list or query per payout number, which is a sequence. The two following HTTP requests return the same report:

```
curl -H "Authorization: OCS-API 083a77d4-051f-46ff-9850-940193f12999" https://dolphin.pay-nxt.com/api/v1/reports/payout/report-number/83
```

```
curl -H "Authorization: OCS-API 083a77d4-051f-46ff-9850-940193f12999" https://dolphin.pay-nxt.com/api/v1/reports/payout/6943da71-11e9-4e86-badc-bfd65a34aaef
```

This will return:

```
HEAD,PAYOUT_REPORT,83,INDIVIDUAL_TRANSACTION,2018-01-20,OC400000000000,BE24363053252838,BBRUBEBB,PAYNXT
```

```
S.A.,BE25068890397682,GKCCBEBB,EUR,UNIFIEDPOST PAYMENTS NV,Avenue Reine Astrid 92A,1310 La Hulpe,BE,2,0.12  
TRANSACTION_CREDIT,047409dc-812d-46ba-b06f-04a5384e996c,0.05,0.05,EUR,2018-01-19,2018-01-19T10:29:30.040Z,NA,BE54799529270697,670305XXXXXX5003,,,BANCONTACT_COLLECT,UNSTRUCTURED,Campaign Test,,  
TRANSACTION_CREDIT,836aa17d-515e-48dd-8903-cd9bf53d6d4b,0.07,0.07,EUR,2018-01-19,2018-01-19T10:46:40.067Z,NA,BE26001188805829,670304XXXXXX7099,,,BANCONTACT_COLLECT,UNSTRUCTURED,001/4037/32870,,
```

12.1.3. Updating a Payment Item

To submit a status update for the following 'batch-update.json' file containing:¹²

```
{ "paymentItems": [  
  { "itemIdentifier": "11111111111", "properties": { "status": "suspended" } },  
  { "itemIdentifier": "22222222222", "properties": { "status": "suspended" } },  
  { "itemIdentifier": "33333333333", "properties": { "status": "suspended" } }  
]
```

You can execute the following curl command:¹³

```
curl -D - -H "Content-Type: application/json" -H "Authorization: OCS-API 083a77d4-051f-46ff-9850-940193f12999" https://dolphin.pay-nxt.com/api/v1/item-updates --data "@batch-update.json"
```

On success, the http response of this file will be a status 202 (meaning accepted for processing) with a "Location:" header linking to the status of this asynchronous processing.

Immediately after the update status submit, you will get (using the previously mentioned link):

```
curl -H "Content-Type: application/json" -H "Authorization: OCS-API 083a77d4-051f-46ff-9850-940193f1239a" https://dolphin.pay-nxt.com/api/v1/item-updates/95a4717c-2695-44c6-9561-73fde8afd999
```

```
{"processingStatus":"pending"}
```

And after one minute or so, the same request will give:

```
{  
  "processingStatus": "completed",  
  "processingDetails": {  
    "paymentItems": {  
      "successfullyUpdated": 3,  
      "failures": []  
    }  
  }  
}
```

¹² '11111111111' being the **merchantItemIdentifier**.

¹³ "-D -" to output response headers.

12.2. Postman ¹⁴

Setup the **POST** method, target URL and HTTP Headers:

POST 		https://dolphin.pay-nxt.com/api/v1/items
Authorization Headers (2) Body ● Pre-request Script Tests		
	Key	Value
☒	Content-Type	application/json
☒	Authorization	OCS-API 6fc54c67-60c7-4880-8080-80814-102f86978c4e

Add the body:

POST 		https://dolphin.pay-nxt.com/api/v1/items
Authorization Headers (2) Body ● Pre-request Script Tests		
☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary JSON (application/json) 		
1 -	{	
2 -	"items":{	
3 -	"1":{	
4 -	"stored":true,	
5 -	"paymentAttributes":{	
6 -	"amount":143.16,	
7 -	"description":"Utility Invoice 766243 - November 2018",	
8 -	"reconciliationReference":"REF-900000766243"	
9 -	}	
10 -	}	
11 -	}	
12 -	}	
13 -		

Click  to obtain the response:

Body		Cookies (1)	Headers (17)	Tests	Status: 201 Created	Time: 299 ms	Size: 1.25 KB
Pretty Raw Preview JSON   							
1 -	{						
2 -	"1": {						
3 -	"id": "f4bc5ebf-6c6b-47ea-ae00-8718158ce9e0",						
4 -	"paymentLinks": {						
5 -	"checkout": "https://starfish.pay-nxt.com/payment-checkouts/f4bc5ebf-6c6b-47ea-ae00-8718158ce9e0",						
6 -	"statusText": "https://starfish.pay-nxt.com/payment-items/f4bc5ebf-6c6b-47ea-ae00-8718158ce9e0/status.txt",						
7 -	"statusImage": "https://starfish.pay-nxt.com/payment-items/f4bc5ebf-6c6b-47ea-ae00-8718158ce9e0/status.png",						
8 -	"extended": "https://dolphin.pay-nxt.com/api/v1/payment-items/f4bc5ebf-6c6b-47ea-ae00-8718158ce9e0/status"						
9 -	}						
10 -	}						
11 -	}						

¹⁴ <https://www.getpostman.com>