Integration Manual

Order API 1.3

Version 1.0

Content

1. Introduction

The purpose of this document is to describe the technical interface of the Payment system and its minimum required data set. Based on the information in this document System Integrators and Web developers will be able to successfully integrate the Payment services into their systems.

In order to process a payment CM offers an Application Programming Interface (API) to interact with a Merchant's system. The next chapters describe the interface and its commands.

2. Interface

2.1. Communication through SOAP

The Order API communicates solely using a SOAP interface over HTTP. SOAP allows the Order API to make a strong definition of the required input and output for its different operations. This definition is described in a WSDL document, which can easily be exchanged with the Merchant, who can use this to implement the required SOAP functionality. For more information on SOAP, please refer to W3C's official documentation on their website: http://www.w3.org/.

2.2. TEST system

CM offers a separate payment system for integration and test purposes. The test system behaves like the production system but will not communicate transactions with acquiring parties. This way a system integrator or web developer can integrate the Payment services and optimize the payment process without financial impact.

SOAP endpoint:

https://test.docdatapayments.com/ps/services/paymentservice/1_3

WSDL:

https://test.docdatapayments.com/ps/services/paymentservice/1_3?wsdl

WSDL viewer:

https://test.docdatapayments.com/ps/orderapi-1_3.wsdl

XSD:

https://test.docdatapayments.com/ps/services/paymentservice/1_3?xsd=1

XSD type definition:

https://test.docdatapayments.com/ps/services/paymentservice/1_3?xsd=2

The required elements and data fields as described in the WSDL are explained in the XSD. Each element where minOccurs="1" indicates that the information is MANDATORY for that element in order to successfully process a payment order. The definition types are described in the XSD type definition source.

For consultants or process owners the WSDL formatted as a document may be a better source to read about the specification. This document can be presented using following link:

http://services.w3.org/xslt?xslfile=http://tomi.vanek.sk/xml/wsdl-viewer.xsl&xmlfile=https://test.docdatapayments.com/ps/services/paymentservice/1_3?wsdl&transform=Submit

2.3. Production system

Once the system integrator or web developer has finished the integration and test process, the web shop or web application should be linked to the CM production system in order to process live payments.

SOAP endpoint:

https://secure.docdatapayments.com/ps/services/paymentservice/1_3

WSDL:

https://secure.docdatapayments.com/ps/services/paymentservice/1_3?wsdl

WSDL viewer:

https://secure.docdatapayments.com/ps/orderapi-1_3.wsdl

XSD:

https://secure.docdatapayments.com/ps/services/paymentservice/1_3?xsd=1

XSD type definition:

https://secure.docdatapayments.com/ps/services/paymentservice/1_3?xsd=2

2.4. Definitions across systems

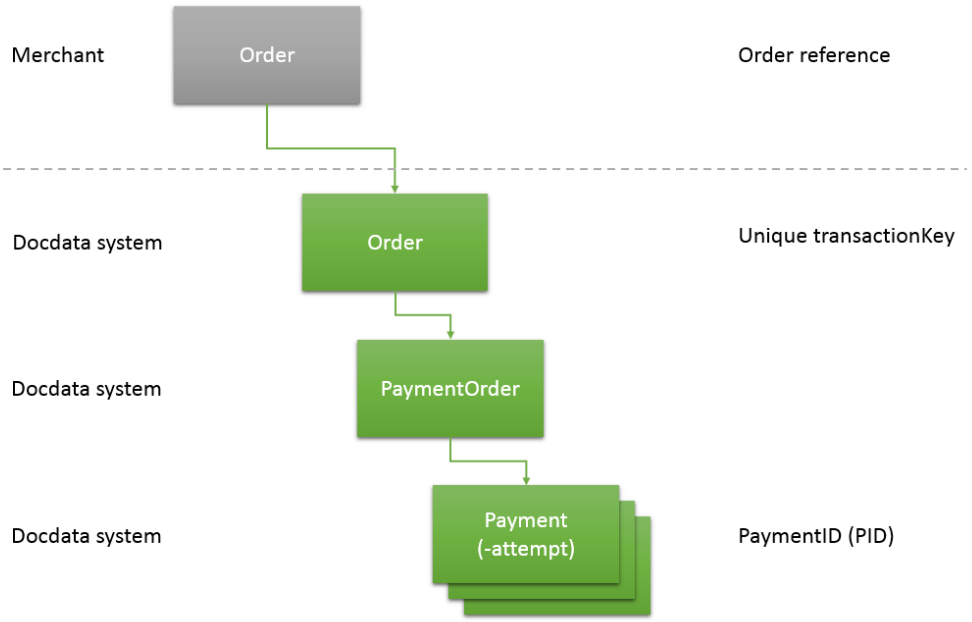

Figure 1: Relation between an Order and Payment(-attempt)s

Typically the Merchant system refers to an order using a unique order reference. This order reference is sent to the Payment system to be able to cross link the order in the web shop to its payments in the Payment system. Once the Payment system has accepted the instruction to process a payment from the web shop, an 'Order' in the Payment system is created. This 'Order' is identified by a unique transaction key which represents the cluster of services and actions.

The next step in the Payment system is to create a PaymentOrder as identifier of the payment service requested. Then each payment (-attempt) is registered to this PaymentOrder by a unique Payment ID (PID).

2.5. Payment scenario

2.5.1. One Page checkout

CM offers a web based payment menu to which a shopper can be redirected to complete a payment. This payment menu is called the One Page Checkout (OPC) or checkout page and is hosted by CM.

The typical process flow for this scenario is:

1. In the web shop a shopper collects products and moves to the checkout page in order to pay.
2. The web shop initiates the creation of an Order in the Payments system. The Payment system returns a digital key which is the unique reference for the Order and its subsequent payment(-attempt)s processed in the Payment system.
3. Using the unique key the web shop redirects the shopper to the One Page Checkout. The shopper may then choose any of the (pre-) configured payment methods to pay the total order amount.
4. Once the shopper has completed the payment the shopper is redirected back to the web shop landing page.
5. The web shop then checks the final status of the Order to confirm if the payment indeed is successfully authorized and/or captured.
6. The web shop then processes the order logistics so the Merchant can ship the products to the shopper.

Note

- In step 3 the shopper may cancel the payment after which he is redirected back to the web shop.

2.5.2. Webdirect scenario

Some payment methods do not require direct interaction with the shopper. For example a shopper may wire the money to a CM bank account by means of a bank transfer. The bank account details can be provided from the web shop itself hence it is not required to forward the shopper to the One Page Checkout to complete a payment.

The typical process flow in this scenario is:

1. In the web shop a shopper collects a product and moves to the checkout page in order to pay.
2. The shopper provides the necessary details for shipment and invoicing for the products ordered.
3. The web shop provides the instructions to the shopper to which bank account the money is to be wired. The shopper may then be redirected back to the product catalogue in the web shop.
4. The web shop initiates the creation of an Order in the Payment system.
5. Thepayment system returns a digital key which is the unique reference for the Order and its subsequent payment(-attempt)s as processed in the Payment system.
6. Using the unique key the web shop now provides the necessary details to the Payment system in order for CM to further process the payment.

Note

- The Webdirect scenario is limited to a specific set of payment methods. For more information how to use Webdirect see the chapter 4.3.

2.6. Order API operations

This chapter describes the different operations available in the Order API and their basic use.

2.6.1. Create

The first step in a payment process is to create an Order (transaction) in the Payment system. This is achieved by initiating a CREATE request. Once the transaction is started, payment actions can be started; either by the shopper via the One Page Checkout or through the Order API in case of a Webdirect payment scenario.

2.6.2. Start

In the scenario where the One Page Checkout is not used, i.e. in a Webdirect scenario, the START operation is used for starting a payment order by the Merchant. The Payment system returns a payment ID (PID) to uniquely identify the payment order.

2.6.3. Proceed

In the scenario where the One Page Checkout is not used, i.e. in a Webdirect scenario, the PROCEED operation is used to finish the authorisation after the merchant returns from the acquirer. For example this is the case when the merchant redirected the shopper to 3D secure. The response from 3D secure needs to be sent to CM to finalize the authorisation.

2.6.4. Cancel

The CANCEL operation is used for cancelling a previously created Payment Order.

2.6.5. Capture

Once a Payment(-attempt) is successfully AUTHORIZED the payment is to be CAPTURED to confirm the acquirer has accepted the payment for processing.

2.6.6. Refund

The REFUND operation is used to perform one or more refunds on payments which have been captured successfully.

2.6.7. Status

The STATUS request is used to query the status of an Order and its payments, captures or refunds in the Payment system. Refer to see chapter 4.13 to determine whether an order is considered “paid”.

2.6.8. Extended Status

The EXTENDED STATUS request is used to retrieve additional status information of an Order and its payments, captures or refunds from the Payment system.

3. Payment process

3.1. Using the One Page Checkout

The workflow as shown in Figure 2 depicts the different steps in which an Order and its payments are processed using the Order API interface and the One Page Checkout.

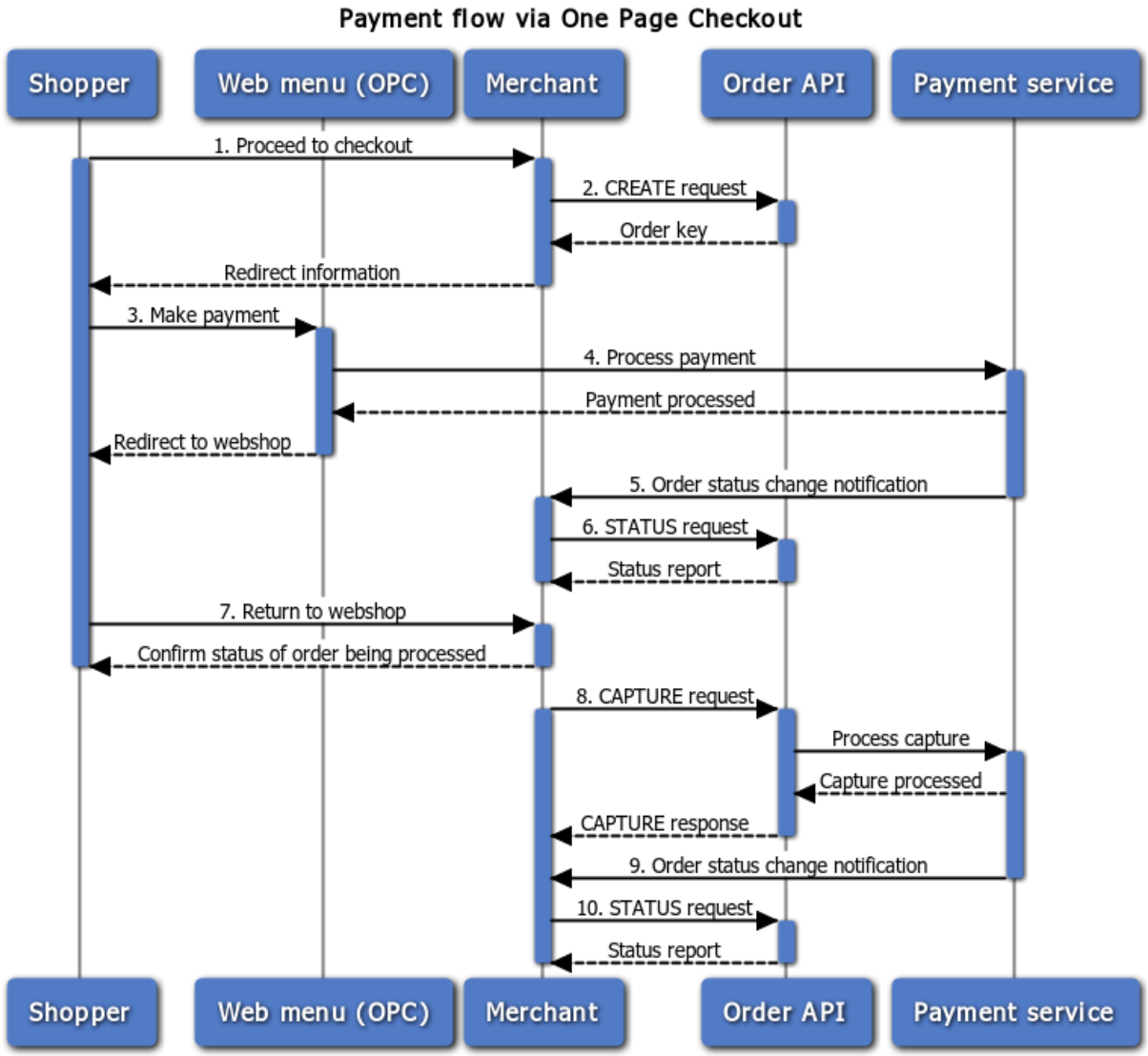

Figure 2: Process flow based on the One Page Checkout scenario

Step 1: The shopper has finished shopping and continues to check out to pay for the order.

Step 2: The merchant first creates an Order in the Payment system using the Order API CREATE request. The Order API returns a unique Order key.

Step 3: The shopper completes his/her payment in the One Page Checkout.

Step 4: The One Page checkout requests the Payment system to process the payment.

Step 5: By means of an asynchronous Status Update call the Payment system notifies the merchant when the status of an Order has been updated.

Step 6: Using the digital key as provided in the response of the CREATE request the merchant checks the status of the Order and whether the payment has been authorized successfully. This adds up to a confidence level to decide whether or not to ship the products.

Step 7: The shopper is redirected back to the Merchant's web shop via the web menu's redirect URL. The landing page of the web shop can be used to display a confirmation of the order to the shopper.

Step 8: When the Merchant is ready to ship the products, in some cases a manual capture is required by the Merchant by means of a CAPTURE request.

Step 9: When the capture has been successfully processed, the Payment system notifies the merchant of a change in the Order by submitting an asynchronous Status Update call.

Step 10: The merchant then again checks the status of the Order to confirm whether or not to ship the products to the shopper. This information is also used to synchronize the web shop administration.

Note

- As depicted in step 5 and step 9 asynchronous Status Update calls may be sent shortly after one another. The web shop should ensure correct handling of numerous asynchronous Status Update calls.
- Some direct payment methods automatically capture a payment immediately after authorization. Please refer to the payment method documentation for more information how to ensure a solid payment process.

3.2. Using Webdirect

The workflow as shown in Figure 3 depict the different steps in which an Order and its payment(-attempt)s are processed in a Webdirect scenario.

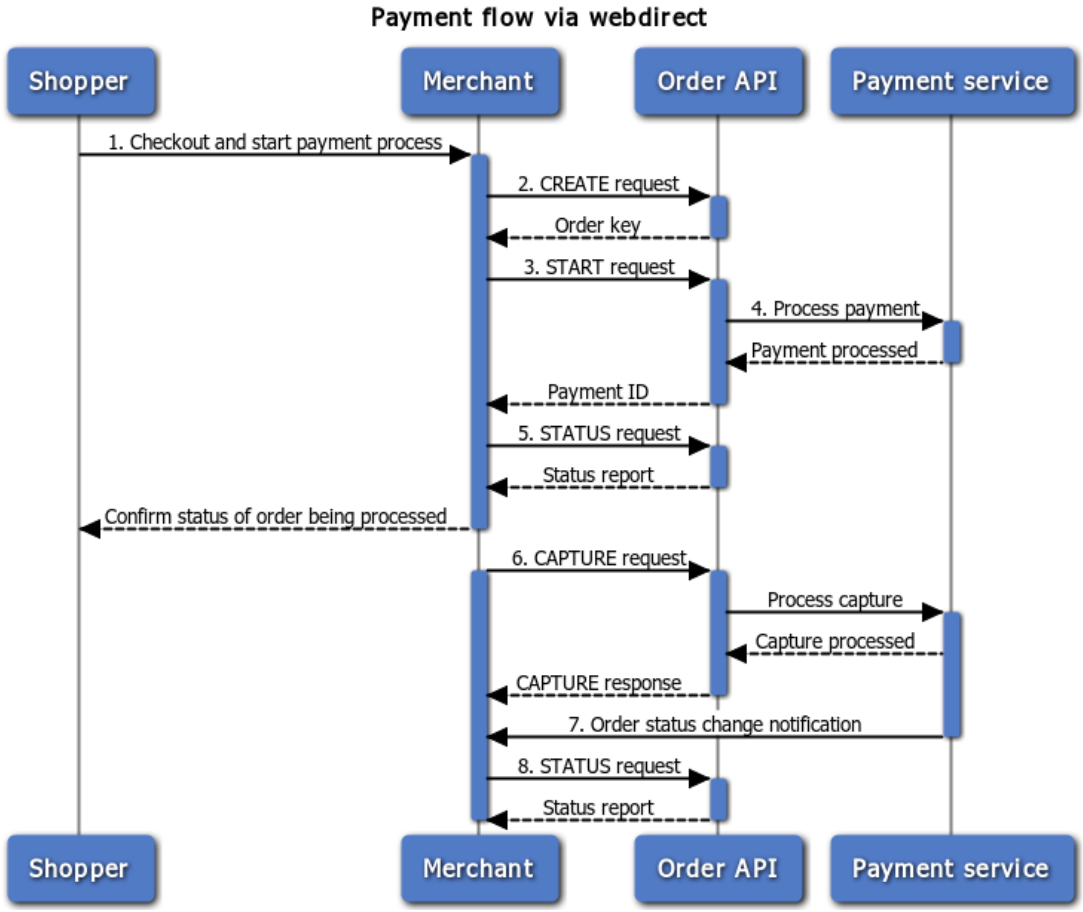

Figure 3: Process flow Webdirect scenario

Step 1: The Shopper has finished shopping and moves to the checkout page of the web shop to start the payment.

Step 2: The merchant initiates an Order in the Payment system by means of a CREATE request.

Step 3: Using the unique digital key as provided in the response of the CREATE request the merchant then initiates the creation of a Payment Order using the START request. The Payment system will return a payment ID as a unique reference for each payment (-attempt).

Step 4: The order API requests the Payment system to process the payment.

Step 5: Using the digital key as provided in the response of the CREATE request the merchant checks the status of the Order and whether the payment has been authorized successfully. This adds up to a confidence level to decide whether or not to ship the products.

Step 6: When the Merchant is ready to ship the products or services, the request to CAPTURE the payment order is initiated by the web shop.

Step 7: By means of an asynchronous Status Update call the Payment system notifies the merchant when the status of an Order has changed.

Step 8: The merchant then checks the status of the Order to confirm whether or not to ship the products to the shopper. This information is also to be used to synchronize the web shop administration.

Note:

Please refer to the payment method documentation for more specific information on how to ensure a solid payment process.

4. Order API calls

This chapter outlines the different SOAP requests and its minimum required parameters as defined in the Order API. For each API call an example request is provided including the response.

4.1. createRequest

The createRequest operation is the first step in the payment service workflow and will create an Order in the Payment system. In response of the createRequest a digital Key is returned which is the unique reference to the Order and its payment(-attempt)s.

4.1.1. Request

The CREATE request should contain all minimum required information which is mandatory to successfully process a payment. If multiple payment methods are offered, please make sure that all required information for any of the offered payment methods is already included in the CREATE request.

Request	Explanation and requirements (See XSD and XSD type definition)
<createRequest xmlns="http://www.docdatapayments.com/services/paymentservice/1 3/" version="1.3"> <merchant name="webshopXYZ" password="p5V4MZqs"/> <merchantOrderReference>1211141202</merchantOrderReference>	*For each call this section identifies the Merchant and the order reference.* *1.The element <merchant> provides the Merchant credentials as setup in the Merchant Back Office.* *2.The element <merchantOrderReference> refers to the Merchant's internal unique reference for this order and is used by CM for informational purposes in the Merchant Back Office.*
<paymentPreferences> <profile>profile-name</profile> <numberOfDaysToPay>14</numberOfDaysToPay> <exhortation> <period1 numberOfDays="7" profile="unknown"/> <period2 numberOfDays="7"/> </exhortation> <terminalId>terminal 1</terminalId> </paymentPreferences>	*This element specifies the settings to use for all payments which are going to be made on this order.* *1.In element <profile> provide the payment profile as configured in the Merchant Back Office that is used to group the payment methods that are offered to pay for this order.* *2.The element <numberOfDaysToPay> indicates the number of days within the shopper is expected to pay. In case this date is due the payment order is considered expired.* *3.In element <exhortation> provide the expected number of days after which a reminder e-mail is to be sent. This email will be sent from the payment system using the email account as provided in the element <email> .* *4. The element <terminalID> is the ID of the terminal that is going to be used when using the payment method Point2Pay.*
<menuPreferences> <css id="1"/> </menuPreferences>	*Preferences to be used for the web menu. Provide the style sheet used to format the web menu.*

	In case the css id is not provided the system will use style sheet 1 (<css id="1"/>) as configured in the account settings.
`<shopper id="client id">` `  <name>` `  <first>Billy</first>` `  <middle>van</middle>` `  <last>Borstel</last>` `  </name>` `  <email>billy.borstel@cmpayments.com</email>` `  <language code="nl"/>` `  <gender>M</gender>` `</shopper>`	*This element contains the information about the shopper.* *1.Providing a valid email address is a requirement in order to successfully process payments.* *2. Provide the preferred language to use for the web menu according ISO 639-1:2002 Part 1: Alpha-2 Language Codes (lowercase).* *3. Provide the gender of the shopper according the information in XSD2:* *M = Male* *F = Female* *U = Undefined*
`<totalGrossAmount currency="EUR">3330</totalGrossAmount>`	*This element sets the total gross amount for the shopper to pay.* *1.Provide the currency of the amount to pay as an integer in the smallest available unit.*
`<billTo>` `  <name>` `  <first>John</first>` `  <last>Doe</last>` `  </name>` `  <address>` `  <street>Street</street>` `  <houseNumber>124</houseNumber>` `  <postalCode>3970AC</postalCode>` `  <city>Utrecht</city>` `  <country code="NL"/>` `  </address>` `</billTo>`	*The element <billTo> provides information about the billing destination.* *Element <postalCode> can only contain a combination of letters and numbers.*
`<description>default transaction</description>` `<receiptText>Thanks for your purchase</receiptText>`	*Description / Additional description – A description for this order.* *The message which is provided in the element <receiptText> will also be submitted to the acquirer to use on the bank statements.*
`<invoice>` `  <shipTo>` `  <name>` `  <first>Billy</first>` `  <last>Borstel</last>` `  </name>` `  <address>` `  <street>Street</street>` `  <houseNumber>124</houseNumber>` `  <postalCode>3970AC</postalCode>` `  <city>Utrecht</city>` `  <country code="NL"/>` `  </address>` `  </shipTo>`	*The element <shipTo> provides the name and address to use for shipping. As part of the <invoice> element this information is mandatory for payment methods which require the invoice information as part of a risk check process. i.e. AfterPay and Klarna Invoice.*
`<additionalDescription>Add. Description</additionalDescription>` `</invoice>`	*In case the description in the <description> element is insufficient, for the invoice an additional description may be added in the element <additionalDescription> for informational purposes.*

<integrationInfo> <webshopPlugin>Webshop plugin</webshopPlugin> <webshopPluginVersion>Webshop plugin version</webshopPluginVersion> <integratorName>Integrator name</integratorName> <programmingLanguage>Programming language</programmingLanguage> <operatingSystem>OS: Linux</operatingSystem> <operatingSystemVersion>OS version: 3.13.0-46-generic</operatingSystemVersion> <ddpXsdVersion>1.3.1</ddpXsdVersion> </integrationInfo> </createRequest>	*This element contains the information about the integration:* *1.When a plugin is used the plugin and the version can be stated.* *2. When an external integrator is used the name of the integrator can be stated.* *3. The programming language* *4. The operating system and the OS version.* *5. The XSD version of CM that is used.*

Figure 4: Example CREATE Request

4.2. CreateRequest Afterpay & Klarna

4.2.1. Request

To support invoicing to shoppers by third parties like Afterpay and Klarna additional information about the shopper and the content of the invoice is required as outlined below:

1. Date of birth of the shopper
2. Valid phone number
3. All products as a separate item

The below example of the CREATE request describes the various elements more specifically.

Request	**Explanation and requirements (See XSD and XSD type definition)**
<shopper id="client_id"> <name> <first>Johan</first> <middle>de</middle> <last>Vries</last> </name> <email>johan.devries@cmpayments.com</email> <language code="nl"/> <gender>M</gender> <dateOfBirth>1983-03-21</dateOfBirth> <phoneNumber> <mobilePhoneNumber>0612345678</mobilePhoneNumber> </shopper>	*This element contains the information about the shopper.* *1.Providing a valid email address is a requirement in order to successfully process payments.* *2. Provide the preferred language to use for the web menu according ISO 639-1:2002 Part 1: Alpha-2 Language Codes (lowercase)..* *3. Provide the gender of the shopper according the information in XSD2:* *M = Male* *F = Female* *U = Undefined* *4. Provide the date of birth as yyyy-mm-dd.* *5. Provide either a home or a mobile number.*
<totalGrossAmount currency="EUR">3330</totalGrossAmount>	*This element sets the total gross amount for the shopper to pay.* *1.Provide the currency of the amount to pay as an integer in the smallest available unit.*
<invoice> <totalNetAmount currency="EUR">3000</totalNetAmount> <totalVatAmount rate="21" currency="EUR">210</totalVatAmount> <totalVatAmount rate="6" currency="EUR">120</totalVatAmount>	*As part of the <invoice> element the request information is used in the One Page Checkout to display the product information to the shopper on the payment menu.*

<item number="1"> <name>Kingston microSD kaart 32GB</name> <code>SDC4/4GB-2ADP</code> <quantity unitOfMeasure="PCS">1</quantity> <description>Kingston microSD kaart 32GB met adapter</description> <netAmount currency="EUR">1000</netAmount> <grossAmount currency="EUR">1210</grossAmount> <vat rate="21"> <amount currency="EUR">210</amount> </vat> <totalNetAmount currency="EUR">1000</totalNetAmount> <totalGrossAmount currency="EUR">1210</totalGrossAmount> <totalVat rate="21"> <amount currency="EUR">210</amount> </totalVat> </item>	*1.Item numbers are integers adding up by 1 for each item.* *2.Ensure to correctly calculate <netAmount> and <grossAmount> as for AfterPay and Klarna Invoice this is subject to the risk check process.*

Figure 5: Example CREATE Request request when Afterpay of Klarna is used

Note

- Net amounts, gross amounts and VAT amounts in relation to these amounts at item level should always correctly add-up.
- A sale or discount should be included in the CREATE request as a separate product item. For a discount item the amount would be negative.

4.2.2. Response

The response of the create call will contain either a success or an error status:

- Success: If the CREATE request was successfully processed, the response will contain the transaction key which is the unique identifier of the Order in the Payment system and its subsequent payment(-attempts). NOTE! A success response does not mean the order has been processed successfully, it means the CREATE request was successfully processed.
- Error: If the CREATE request was not successfully processed by the Payment system, the response will contain an error code specifying the reason of the failure. The <error code> is defined in the XSD and may be used for error handling. The description of the error code however is for information purposes only and may change without notice.

Response SUCCESS	Explanation
<createResponse xmlns="http://www.docdatapayments.com/services/paymentservice/1_3/"> <createSuccess> <success code="SUCCESS">Operation successful.</success> <key>F022EA07FC10FDA97730FBFE67453D40</key> </createSuccess> </createResponse>	*The element <createSuccess> indicates that the order is successfully accepted by the Payment system. The unique <key> is required to further initiate payment actions.*

Figure 6: Example CREATE Request response: the payment order has been accepted for processing

Response ERROR	Explanation
<createResponse xmlns="http://www.docdatapayments.com/services/paymentservice/1_3/"> <createError> <error code="REQUEST DATA INCORRECT">Invalid payment profile for the order</error> </createError> </createResponse>	*The element <createError> indicates the reason of the failure. See Figure 8 for a list of possible error codes.*

Figure 7: Example CREATE Request response: the order could not be accepted by the Payment system for processing

Error code	Explanation
REQUEST_DATA_INCORRECT	*Request data is incorrect*
REQUEST_DATA_MISSING	*Request data is missing*
SECURITY_ERROR	*Error related to security violations such as login failure or blocked IP address*
INTERNAL_ERROR	*Payment system error*
UNKNOWN_ERROR	*The error is not specified*

Figure 8: List of error codes

4.3. startRequest without a token

The START request is specific to the Webdirect scenario where a payment can be controlled without online interaction with the shopper. See Figure 10 for an overview of payment methods available for Webdirect.

4.3.1. Request

The START request requires the unique key as provided in the response of the CREATE request. In addition the START request should include the minimum required payment details of the anticipated payment methods as defined in the XSD. Note that the elements and information types for each of the elements may be different for each of the payment methods available. Therefore it is best practice to always provide the complete set of user data and order details in the CREATE request as there is no option to add details later.

Request	Explanation
`<startRequest xmlns="http://www.docdatapayments.com/services/paymentservice/13/" version="1.3"> <merchant name="webshopXYZ" password="p5V4MZqs"/> <paymentOrderKey>F022EA07FC10FDA97730FBFE67453D40</paymentOrderKey>`	*For each call this section identifies the Merchant and the order reference.* *1.The element <merchant> provides the Merchant credentials as setup in the Merchant Back Office.* *2.The <paymentOrderKey> refers to the unique <key> which is provided in the response of a successful CREATE request.*
`<payment> <paymentMethod>MASTERCARD</paymentMethod> <masterCardPaymentInput> <cardHolderName>John Doe</cardHolderName> <cardNumber>5123456789012346</cardNumber> <expiryDate month="01" year="23"/> <securityCode>123</securityCode> </masterCardPaymentInput> </payment>`	*The element <paymentMethod> provides the name of the payment method to be used to collect the amount to be paid.* *The element <masterCardPaymentInput> is specific for the payment method 'mastercard' as provided in the <paymentMethod> element of the XSD.* *For a list of payment methods and their required payment input requirements see Figure 10.*
`<returnUrl>/webshop/landingpage.jsf</returnUrl>`	*The element <returnUrl> provides the landing page of the webshop after the shopper finishes 3D secure. NOTE this shouldn't be the success page since it is*

	still unknown at this point if the payment is successful.
<shopperInfo> <shopperIp>211.111.111.1</shopperIp> <browserAccept>text/html,application/xhtml+xml,application/xml; q=0.9,*/*;q=0.8</browserAccept> <browserUserAgent>Mozilla/5.0 (Windows NT 6.1; WOW64; rv:36.0) Gecko/20100101 Firefox/36.0</browserUserAgent> </shopperInfo>	*The element <shopperInfo> contains the mandatory elements for 3D secure (these are standard HTTP header elements):* *1. The IP address of the shopper* *2. The browser acceptance criteria* *3. The browser User Agent*
<integrationInfo> <webshopPlugin>Webshop plugin</webshopPlugin> <webshopPluginVersion>Webshop plugin version</webshopPluginVersion> <integratorName>Integrator name</integratorName> <programmingLanguage>Programming language</programmingLanguage> <operatingSystem>OS: Linux</operatingSystem> <operatingSystemVersion>OS version: 3.13.0-46- generic</operatingSystemVersion> <ddpXsdVersion>1.3.1</ddpXsdVersion> </integrationInfo> </startRequest>	*This element contains the information about the integration:* *1.When a plugin is used the plugin and the version can be stated.* *2. When an external integrator is used the name of the integrator can be stated.* *3. The programming language* *4. The operating system and the OS version.* *5. The XSD version of CM that is used.*

Figure 9: Example START Request: Payment by Mastercard

<paymentMethod>	Payment method description	Required information. See XSD section: <complex Type name="paymentRequestInput">
AMEX	American Express	amexPaymentInput
BANK_TRANSFER	Bank transfer	bankTransferPaymentInput
ELV	ELV (German direct debit)	elvPaymentInput
SEPA_DIRECT_DEBIT	SEPA direct Debit	directDebitPaymentInput
MASTERCARD	Mastercard	masterCardPaymentInput
VISA	Visa	visaPaymentInput
MAESTRO	Maestro	maestroPaymentInput
MISTERCASH	Bancontact-Mistercash	misterCashPaymentInput
POINT_OF_SALE	Point2Pay	pointOfSalePaymentInput
OFFLINE	Offline payment	offlinePaymentInput
IDEAL	iDEAL	iDealPaymentInput

Figure 10: Overview of the payment methods available for Webdirect including the referral to their requirements as described in the XSD

4.3.2. Response

The response of the START request will contain either an error or a success status:

- Success: If the START request was successfully processed, the response will contain the unique *payment ID* which is required to query the status of the payment order or cancelling the specific payment order in a later stage. NOTE! A success response does not mean the order has been processed successfully, it means the START request was successfully processed.

- Error: If the START request was not successfully processed by the Payment system, the response will contain an error code specifying the reason of the failure. The <error code> is defined in the XSD and may be used for error handling. The description of the error code however is for information purposes only and may change without notice.

Response SUCCESS	Explanation
<startResponse xmlns="http://www.docdatapayments.com/services/paymentservice/1_3/"> <startSuccess> <success code="SUCCESS">Operation successful.</success>	*The element <startSuccess> indicates that the request is successfully accepted by the Payment system for processing.*
<paymentResponse> <paymentSuccess> <status>REDIRECTED_FOR_AUTHENTICATION</status> <id>4912345678</id> </paymentSuccess> </paymentResponse>	*The element <paymentResponse> provides a unique identifier specific for this payment order. This unique ID is called the Payment ID (PID). It also provides the current status of the payment.*
<redirect> <method>POST</method> <url>https://test.docdatapayments.com/ps sim/3dsecureauthentication.jsf?data=NTEyMzQ1Njc4OTAxMjM0Ng==</url> <parameters> <parameter name="MD">4912345678</parameter> <parameter name="PaReq">eJxVkVFygjAQhq/C5LUWdaxMp36oGNbparameter> <parameter name="TermUrl">/ps_sim/parameters.jsf</parameter> </parameters> </redirect> </startSuccess> </startResponse>	*The element <redirect> provides all details needed to redirect the customer to the acquirer.*

Figure 11: Example START Request response: The request is accepted.

Response ERROR	Explanation
<startResponse xmlns="http://www.docdatapayments.com/services/paymentservice/1_3/"> <startError> <error code="REQUEST_DATA_INCORRECT">Payment for order could not be authorized.</error> </startError> </startResponse>	*The element <startError> indicates the reason of the failure. See Figure 13 for a list of possible error codes.* *Note: The <error code> is defined in the XSD and may be used for error handling. The description is for information purposes only and may change without notice.*

Figure 12: Example START Request response: The request has not been accepted for processing

Error code	Explanation
REQUEST_DATA_INCORRECT	Request data is incorrect
REQUEST_DATA_MISSING	Request data is missing
SECURITY_ERROR	Error related to security violations such as login failure or blocked IP address.
INTERNAL_ERROR	Payment system error
UNKNOWN_ERROR	The error is not specified

Figure 13: Error codes and explanation

4.4. startRequest with a token

If the token API was used to create a token, then it is possible to start a payment with the token. A token will never contain the security code, so this needs to be sent separately in the START request.

4.4.1. Request

Request	Explanation
<startRequest xmlns="http://www.docdatapayments.com/services/paymentservice/1 3/" version="1.3"> <merchant name="webshopXYZ" password="p5V4MZqs"/> <paymentOrderKey>F022EA07FC10FDA97730FBFE67453D40</paymentOrder Key>	*For each call this section identifies the Merchant and the order reference.* *1.The element <merchant> provides the Merchant credentials as setup in the Merchant Back Office.* *2.The <paymentOrderKey> refers to the unique <key> which is provided in the response of a successful CREATE request.*
<token key="6a277b44520111b027ad4f3363eb977be9c261"> <masterCardTokenInput> <securityCode>123</securityCode> </masterCardTokenInput> </token>	*The <token> contains the mandatory elements for the mastercard payment:* *1. The token key to access the earlier stored payment input.* *2. The security code to authorize the payment*
<returnUrl>/webshop/landingpage.jsf</returnUrl>	*The element <returnUrl> provides the landing page of the webshop after the shopper finishes 3D secure. NOTE this shouldn't be the success page since it is still unknown at this point if the payment is successful.*
<shopperInfo> <shopperIp>211.111.111.1</shopperIp> <browserAccept>text/html,application/xhtml+xml,application/xml; q=0.9,*/*;q=0.8</browserAccept> <browserUserAgent>Mozilla/5.0 (Windows NT 6.1; WOW64; rv:36.0) Gecko/20100101 Firefox/36.0</browserUserAgent> </shopperInfo>	*The element <shopperInfo> contains the mandatory elements for 3D secure (these are standard HTTP header elements:* *1. The IP address of the shopper* *2. The browser acceptance criteria* *3. The browser User Agent*
<integrationInfo> <webshopPlugin>Webshop plugin</webshopPlugin> <webshopPluginVersion>Webshop plugin version</webshopPluginVersion> <integratorName>Integrator name</integratorName> <programmingLanguage>Programming language</programmingLanguage> <operatingSystem>OS: Linux</operatingSystem> <operatingSystemVersion>OS version: 3.13.0-46- generic</operatingSystemVersion> <ddpXsdVersion>1.3.1</ddpXsdVersion> </integrationInfo> </startRequest>	*The element <integrationInfo> contains the information about the integration:* *1.When a plugin is used the plugin and the version can be stated.* *2. When an external integrator is used the name of the integrator can be stated.* *3. The programming language* *4. The operating system and the OS version.* *5. The XSD version of CM that is used.*

Figure 14: Example START Request with token and security code: Payment by Mastercard

4.5. proceedRequest

The PROCEED request is specific to the Webdirect scenario where the shopper is redirected to an acquirer. It is used to finish the authorisation after the merchant returns from the acquirer. For example this is the case when the merchant redirected the shopper to 3D secure.

4.5.1. Request

The PROCEED request requires the unique payment ID as provided in the response of the START request. The request should further contain all minimum required information which is mandatory to successfully process the authorisation of the payment.

Request	Explanation
<proceedRequest xmlns="http://www.docdatapayments.com/services/paymentservice/1_3/" version="1.3"> <merchant name="webshopXYZ" password="p5V4MZqs"/> <paymentId>4907817914</paymentId>	*For each call this section identifies the Merchant and the payment reference.* *1.The element <merchant> provides the Merchant credentials as setup in the Merchant Back Office.* *2.The <paymentId> refers to the unique ID which is provided in the response of a successful START request.*
<threeDomainSecureAuthenticationResult> <MD>4907817914</MD> <PARes>eJzdmEmP4zqSgO/6FYXso1FPu201XNnQvlmy9sU3bZZk7dZq/fqWM1/Vq6kuNHrm1BgDiSSDwWAw gvxI6vSPpSq/TMmjz5v62xv8B/T2JamjJs7r9NubbXFfj29f+iGo46Bs6uTbW928/eP9ZGWPJGHM….. </PARes> </threeDomainSecureAuthenticationResult>	*The element <threeDomainSecureAuthenticationResult> contains the mandatory elements for authorizing a 3D secure payment:* *1. The element <MD> (Merchant Data) can be found in the response of the 3D secure request.* *2. The element <PARes> is the authentication information sent by the card issuer*
<integrationInfo> <webshopPlugin>Webshop plugin</webshopPlugin> <webshopPluginVersion>Webshop plugin version</webshopPluginVersion> <integratorName>Integrator name</integratorName> <programmingLanguage>Programming language</programmingLanguage> <operatingSystem>OS: Linux</operatingSystem> <operatingSystemVersion>OS version: 3.13.0-46-generic</operatingSystemVersion> <ddpXsdVersion>1.3.1</ddpXsdVersion> </integrationInfo> </proceedRequest>	*This element contains the information about the integration:* *1.When a plugin is used the plugin and the version can be stated.* *2. When an external integrator is used the name of the integrator can be stated.* *3. The programming language* *4. The operating system and the OS version.* *5. The XSD version of CM that is used.*

Figure 15: Example PROCEED Request: 3D secure Payment

4.5.2. Response

The response of the PROCEED request will contain either an error or a success status:

- Success: if the PROCEED request was successfully processed, the response will contain the status of the payment
- Error: If the PROCEED request was not successfully processed by the Payment system, the response will contain an error code specifying the reason of the failure. The <error code> is defined in the XSD and may be used for error handling. The description of the error code however is for information purposes only and may change without notice.

Response SUCCESS	Explanation
<proceedResponse xmlns="http://www.docdatapayments.com/services/paymentservice/1_3/" ddpXsdVersion="1.3.1"> <proceedSuccess> <success code="SUCCESS">Operation successful.</success>	*The element <proceedSuccess> indicates that the request is successfully accepted by the Payment system for processing.* *However this does not indicate a successful payment!*

<paymentResponse> <paymentSuccess> <status>AUTHORIZED</status> <id>4907817914</id> </paymentSuccess> </paymentResponse> </proceedSuccess> </proceedResponse>	*The <paymentResponse> provides the status of the payment.*

Figure 16: Example PROCEED Request response: The request is accepted.

Response ERROR	Explanation
<proceedResponse xmlns="http://www.docdatapayments.com/services/paymentservice/1_3/" ddpXsdVersion="1.3.1"> <proceedErrors> <error code="REQUEST DATA INCORRECT">Payment for order could not be authorized.</error> </proceedErrors> </proceedResponse>	*The element <proceedErrors> indicates the reason of the failure. See Figure 18 for a list of possible error codes.* *Note: The <error code> is defined in the XSD and may be used for error handling. The description is for information purposes only and may change without notice.*

Figure 17: Example PROCEED Request response: The request has not been accepted for processing

Error code	Explanation
REQUEST_DATA_INCORRECT	Request data is incorrect
REQUEST_DATA_MISSING	Request data is missing
SECURITY_ERROR	Error related to security violations such as login failure or blocked IP address.
INTERNAL_ERROR	Payment system error
UNKNOWN_ERROR	The error is not specified

Figure 18: Error codes and explanation

4.6. cancelRequest

4.6.1. Request

The purpose of the CANCEL request is to force the Payment system not to accept any payment actions for the given Order anymore. In a scenario where a shopper has cancelled the order in the web shop, the CANCEL request is to be used to synchronize the administration in both the web shop and the Payment system to ensure no payments are processed by CM for the order.

Request	Explanation
<cancelRequest xmlns="http://www.docdatapayments.com/services/paymentservice/1_3/" version="1.3"> <merchant name="webshopXYZ" password="p5V4MZqs"/> <paymentOrderKey>F022EA07FC10FDA97730FBFE67453D40</paymentOrderKey>	*The element <PaymentOrderKey> refers to the unique transaction <key> as provided in the response of the successful CREATE request.*

<integrationInfo> <webshopPlugin>Webshop plugin</webshopPlugin> <webshopPluginVersion>Webshop plugin version</webshopPluginVersion> <integratorName>Integrator name</integratorName> <programmingLanguage>Programming language</programmingLanguage> <operatingSystem>OS: Linux</operatingSystem> <operatingSystemVersion>OS version: 3.13.0-46-generic</operatingSystemVersion> <ddpXsdVersion>1.3.1</ddpXsdVersion> </integrationInfo> </cancelRequest>	*This element contains the information about the integration:* *1.When a plugin is used the plugin and the version can be stated.* *2. When an external integrator is used the name of the integrator can be stated.* *3. The programming language* *4. The operating system and the OS version.* *5. The XSD version of CM that is used*

Figure 19: Example CANCEL Request

4.6.2. Response

The response of the CANCEL request will contain either an error or a success status:

- Success: If the CANCEL request was successfully processed. The status of the *transaction* will be updated by the Payment system. NOTE! A success response does not mean the order has been cancelled successfully, it means the CANCEL request was successfully processed.
- Error: If the CANCEL request was not successfully processed by the Payment system, the response will contain an error code specifying the reason of the failure. The <error code> is defined in the XSD and may be used for error handling. The description of the error code however is for information purposes only and may change without notice.

The response of the cancelRequest does not confirm that the order indeed is cancelled. Always use the STATUS request to retrieve the actual status of the Order and its payments and update the web shop order administration accordingly.

Response SUCCESS	**Explanation**
<cancelResponse xmlns="http://www.docdatapayments.com/services/paymentservice/1_3/"> <cancelSuccess> <success code="SUCCESS">Operation successful.</success> <result>SUCCESS</result> </cancelSuccess> </cancelResponse>	*The element <cancelSuccess> indicates that the request to cancel the transaction is accepted.* *The element <result> indicates the status of the cancellation of the underlying payment(s)*

Figure 20: Example CANCEL Request response: The request is accepted.

Response ERROR	**Explanation**
<cancelResponse xmlns="http://www.docdatapayments.com/services/paymentservice/1_3/"> <cancelError> <error code="REQUEST DATA INCORRECT">Order could not be found with the given key.</error> </cancelError> </cancelResponse>	*The element <cancelError> indicates the reason of the failure. See Figure 22 for a list of possible error codes.*

Figure 21: Example CANCEL Request response: The request has not been accepted for processing

Error code	Explanation
REQUEST_DATA_INCORRECT	Request data is incorrect
REQUEST_DATA_MISSING	Request data is missing
SECURITY_ERROR	Error related to security violations such as login failure or blocked IP address.
INTERNAL_ERROR	Payment system error
UNKNOWN_ERROR	The error is not specified

Figure 22: Error codes cancelRequest

4.7. captureRequest

4.7.1. Request

The purpose of the CAPTURE request is to force the Payment system to capture a payment order which is successfully authorized for processing by the acquirer. This operation however is not required in all scenario's. Depending on the payment method, CM offers an option to automatically capture payment orders after a predefined number of days. This option supports payment methods like AfterPay and Klarna Invoice where products are only to be shipped when the products are collected for shipment.

Refer to the payment method specific information for more details whether a manual CAPTURE is required.

Request	Explanation
<captureRequest xmlns="http://www.docdatapayments.com/services/paymentservice/1 3/" version="1.3"> <merchant name="webshopXYZ" password="p5V4MZqs"/> <paymentId>4907435245</paymentId> <amount currency="EUR">1000</amount>	*The element <paymentId> refers to the unique Payment ID as returned to the web shop when initiating a STATUS call. See chapter 4.9 for more information how to initiate a STATUS call and process the Status Report details.*
<integrationInfo> <webshopPlugin>Webshop plugin</webshopPlugin> <webshopPluginVersion>Webshop plugin version</webshopPluginVersion> <integratorName>Integrator name</integratorName> <programmingLanguage>Programming language</programmingLanguage> <operatingSystem>OS: Linux</operatingSystem> <operatingSystemVersion>OS version: 3.13.0-46- generic</operatingSystemVersion> <ddpXsdVersion>1.3.1</ddpXsdVersion> </integrationInfo> </captureRequest>	*This element contains the information about the integration:* *1.When a plugin is used the plugin and the version can be stated.* *2. When an external integrator is used the name of the integrator can be stated.* *3. The programming language* *4. The operating system and the OS version.* *5. The XSD version of CM that is used*

Figure 23: Example CAPTURE Request

4.7.2. Response

The response of the CAPTURE request will contain either a success or an error status:

- Success: If the CAPTURE request was successfully processed. The status of the *transaction* will be updated by the Payment system. NOTE! A success response does not mean the capture has been processed successfully, it means the CAPTURE request was successfully processed.
- Error: If the CAPTURE request was not successfully processed by the Payment system, the response will contain an error code specifying the reason of the failure. The <error code> is defined in the XSD and may be used for error handling. The description of the error code however is for information purposes only and may change without notice.

The response of the captureRequest does not confirm that the order indeed is captured. Always use the STATUS request to retrieve the actual status of the Order and its payments and update the web shop order administration accordingly.

Response SUCCESS	Explanation
<captureResponse xmlns="http://www.docdatapayments.com/services/paymentservice/1_3/"> <captureSuccess> <success code="SUCCESS">Operation successful.</success> </captureSuccess> </captureResponse>	*The element <captureSuccess> indicates that the request to capture the transaction is accepted.*

Figure 24: Example CAPTURE Request response: The request is accepted.

Response ERROR	Explanation
<captureResponse xmlns="http://www.docdatapayments.com/services/paymentservice/1_3/"> <captureError> <error code="REQUEST DATA INCORRECT">Payment id incorrect.</error> </captureError> </captureResponse>	The element <captureError> indicates the reason of the failure. See Figure 26 for a list of possible error codes.

Figure 25: Example CAPTURE Request response: The request has not been accepted for processing

Error code	Explanation
REQUEST_DATA_INCORRECT	Request data is incorrect
REQUEST_DATA_MISSING	Request data is missing
SECURITY_ERROR	Error related to security violations such as login failure or blocked IP address.
INTERNAL_ERROR	Payment system error
UNKNOWN_ERROR	The error is not specified

Figure 26: Error codes cancelRequest

4.8. refundRequest

4.8.1. Request

In cases where a Merchant wants to refund a certain amount on a Payment Order the REFUND request can be used. The refundRequest enables the refund process to be controlled and automated end-to-end between the web shop and the Payment system.

Request	Explanation
<refundRequest xmlns="http://www.docdatapayments.com/services/paymentservice/1 2/" version="1.3"> <merchant name="webshopXYZ" password="p5V4MZqs"/> <paymentId>4907435245</paymentId> <amount currency="EUR">100</amount>	*The element <paymentId> refers to the unique Payment ID as returned to the web shop when initiating a STATUS call. See chapter Error! Reference source not found. for more information how to initiate a STATUS call and process the Status Report details.*
<refundBankAccount> <holderName>Ted Vinke</holderName> <holderCity>Nijmegen</holderCity> <holderCountry code="NL"/> <bic>INGBNL2A</bic> <iban>NL47INGB0007673523</iban> </refundBankAccount>	*The <refundBankAccount> is optional. The bank account information is only required for payment methods for which no information is known to the system yet. If provided here, but not needed, then it will be ignored (e.g. for iDEAL or credit card refunds).*
<integrationInfo> <webshopPlugin>Webshop plugin</webshopPlugin> <webshopPluginVersion>Webshop plugin version</webshopPluginVersion> <integratorName>Integrator name</integratorName> <programmingLanguage>Programming language</programmingLanguage> <operatingSystem>OS: Linux</operatingSystem> <operatingSystemVersion>OS version: 3.13.0-46- generic</operatingSystemVersion> <ddpXsdVersion>1.3.1</ddpXsdVersion> </integrationInfo> </refundRequest>	*This element contains the information about the integration:* *1.When a plugin is used the plugin and the version can be stated.* *2. When an external integrator is used the name of the integrator can be stated.* *3. The programming language* *4. The operating system and the OS version.* *5. The XSD version of Docdata that is used*

Figure 27: Example REFUND Request

4.8.2. Response

The response of the REFUND request will contain either a success or an error status:

- Success: If the REFUND request was successfully processed. The status of the *transaction* will be updated by the Payment system. NOTE! A success response does not mean the refund has been processed successfully, it means the REFUND request was successfully processed.
- Error: If the REFUND request was not successfully processed by the Payment system, the response will contain an error code specifying the reason of the failure. The <error code> is defined in the XSD and may be used for error handling. The description of the error code however is for information purposes only and may change without notice.

The response of the refundRequest does not confirm that the order indeed is refunded. Always use the STATUS request to retrieve the actual status of the Order and its payments and update the web shop order administration accordingly.

Response SUCCESS	Explanation
<refundResponse xmlns="http://www.docdatapayments.com/services/paymentservice/1 3/"> <refundSuccess> <success code="SUCCESS">Operation successful.</success> </refundSuccess> </refundResponse>	*The element <refundSuccess> indicates that the request to refund an given amount is accepted for processing.*

Figure 28: Example REFUND Request response: The request is accepted.

Response ERROR	Explanation
<refundResponse xmlns="http://www.docdatapayments.com/services/paymentservice/13/"> <refundError> <error code="REQUEST_DATA_INCORRECT">Invalid amount.</error> </refundError> </refundResponse>	The element <refundError> indicates the reason of the failure. See Figure 30 for a list of possible error codes.

Figure 29: Example REFUND Request response: The request has not been accepted for processing

Error code	Explanation
REQUEST_DATA_INCORRECT	Request data is incorrect
REQUEST_DATA_MISSING	Request data is missing
SECURITY_ERROR	Error related to security violations such as login failure or blocked IP address.
INTERNAL_ERROR	Payment system error
UNKNOWN_ERROR	The error is not specified

Figure 30: Error codes refundRequest

4.9. statusRequest

4.9.1. Request

The STATUS request can be used to retrieve a report reflecting the actual status of an Order, its payments and its captures or refunds. The statusRequest is used to determine whether an Order is considered “paid”.

Request	Explanation
<statusRequest xmlns="http://www.docdatapayments.com/services/paymentservice/1_3/" version="1.3"> <merchant name="webshopXYZ" password="p5V4MZqs"/> <paymentOrderKey>A1AECDCCCB8577A9C89A146263E38554</paymentOrderKey>	*The element <PaymentOrderKey> refers to the unique transaction <key> as provided in the response of the successful CREATE request.*
<integrationInfo> <webshopPlugin>Webshop plugin</webshopPlugin> <webshopPluginVersion>Webshop plugin version</webshopPluginVersion> <integratorName>Integrator name</integratorName> <programmingLanguage>Programming language</programmingLanguage> <operatingSystem>OS: Linux</operatingSystem> <operatingSystemVersion>OS version: 3.13.0-46-generic</operatingSystemVersion> <ddpXsdVersion>1.3.1</ddpXsdVersion> </integrationInfo> </statusRequest>	*This element contains the information about the integration:* *1.When a plugin is used the plugin and the version can be stated.* *2. When an external integrator is used the name of the integrator can be stated.* *3. The programming language* *4. The operating system and the OS version.* *5. The XSD version of CM that is used*

Figure 31: Example STATUS Request

4.9.2. Response

The response of the STATUS request will contain either a success or an error status:

- Success: If the STATUS request was successfully processed. The status of the *transaction* will be updated by the Payment system. NOTE! A success response does not mean the order has been processed successfully, it means the STATUS request was successfully processed.
- Error: If the STATUS request was not successfully processed by the Payment system, the response will contain an error code specifying the reason of the failure. The <error code> is defined in the XSD and may be used for error handling. The description of the error code however is for information purposes only and may change without notice.

Response SUCCESS	Explanation
<statusResponse xmlns="http://www.docdatapayments.com/services/paymentservice/1_3/"> <statusSuccess> <success code="SUCCESS">Operation successful.</success>	*The element <statusSuccess> indicates that the request is accepted by the Payment system for processing the data.*
<report> <approximateTotals exchangedTo="EUR" exchangeRateDate="2015-07-14 16:12:14"> <totalRegistered>3330</totalRegistered> <totalShopperPending>3330</totalShopperPending> <totalAcquirerPending>0</totalAcquirerPending> <totalAcquirerApproved>0</totalAcquirerApproved> <totalCaptured>0</totalCaptured> <totalRefunded>0</totalRefunded> <totalChargedback>0</totalChargedback> <totalReversed>0</totalReversed> </approximateTotals>	*The element <approximateTotals> provides values which can be used to determine whether an order is considered 'paid'.*
<payment> <id>4907819251</id> <paymentMethod>BANK TRANSFER</paymentMethod> <authorization> <status>AUTHORIZED</status> <amount currency="EUR">3330</amount> <confidenceLevel>SHOPPER PENDING</confidenceLevel> <capture> <status>STARTED</status> <amount currency="EUR">3330</amount> </capture> </authorization> </payment> </report> </statusSuccess> </statusResponse>	*For each Payment registered to the Payment Order as represented by the <PaymentOrderKey>, the status details of the payment are provided in the corresponding <payment> element.*

Figure 32: Example STATUS Request response

Response ERROR	Explanation
<statusResponse xmlns="http://www.docdatapayments.com/services/paymentservice/1_3/"> <statusError> <error code="REQUEST DATA INCORRECT">Order could not be found with the given key.</error> </statusError> </statusResponse>	*The element <statusError> indicates the reason of the failure. See Figure 34 for a list of possible error codes.*

Figure 33: Example STATUS Request response: The request has not been accepted for processing

Error code	Explanation
REQUEST_DATA_INCORRECT	Request data is incorrect
REQUEST_DATA_MISSING	Request data is missing
SECURITY_ERROR	Error related to security violations such as login failure or blocked IP address.
INTERNAL_ERROR	Payment system error
UNKNOWN_ERROR	The error is not specified

Figure 34: Error codes refundRequest

4.10. statusExtendedRequest

4.10.1. Request

The EXTENDED STATUS request can be used to retrieve additional information of an Order, its payments and its captures or refunds.

Request	Explanation
<extendedStatusRequest xmlns="http://www.docdatapayments.com/services/paymentservice/1_3/" version="1.3"> <merchant name="testshop" password="wdsYFFQL"/> <paymentOrderKey>625BB0DD8BF2DAAFDD36257FD2D5B9E6</paymentOrderKey>	*The element <PaymentOrderKey> refers to the unique transaction <key> as provided in the response of the successful CREATE request.*
<integrationInfo> <webshopPlugin>Webshop plugin</webshopPlugin> <webshopPluginVersion>Webshop plugin version</webshopPluginVersion> <integratorName>Integrator name</integratorName> <programmingLanguage>Programming language</programmingLanguage> <operatingSystem>OS: Linux</operatingSystem> <operatingSystemVersion>OS version: 3.13.0-46-generic</operatingSystemVersion> <ddpXsdVersion>1.3.1</ddpXsdVersion> </integrationInfo> </extendedStatusRequest>	*This element contains the information about the integration:* *1.When a plugin is used the plugin and the version can be stated.* *2. When an external integrator is used the name of the integrator can be stated.* *3. The programming language* *4. The operating system and the OS version.* *5. The XSD version of CM that is used*

Figure 35: Example EXTENDED STATUS Request

4.10.2. Response

The response of the EXTENDED STATUS request will contain either a success or an error status:

- Success: If the EXTENDED STATUS request was successfully processed. The status of the *transaction* will be updated by the Payment system. NOTE! A success response does not mean the order has been processed successfully, it means the EXTENDED STATUS request was successfully processed.
- Error: If the EXTENDED STATUS request was not successfully processed by the Payment system, the response will contain an error code specifying the reason of the failure. The <error code> is defined in the XSD and may be used for error handling. The description of the error code however is for information purposes only and may change without notice.

Response SUCCESS	Explanation
<extendedStatusResponse xmlns="http://www.docdatapayments.com/services/paymentservice/1 3/" ddpXsdVersion="1.3.1"> <statusSuccess> <success code="SUCCESS">Operation successful.</success>	*The element <StatusSuccess> indicates that the request is accepted by the Payment system for processing the data.*
<report> <approximateTotals exchangedTo="EUR" exchangeRateDate="2015-07-14 16:12:14"> <totalRegistered>3330</totalRegistered> <totalShopperPending>3330</totalShopperPending> <totalAcquirerPending>0</totalAcquirerPending> <totalAcquirerApproved>0</totalAcquirerApproved> <totalCaptured>0</totalCaptured> <totalRefunded>0</totalRefunded> <totalChargedback>0</totalChargedback> <totalReversed>0</totalReversed> </approximateTotals>	*The element <approximateTotals> provides values which can be used to determine whether an order is considered 'paid'.*
<payment> <id>4907819251</id> <paymentMethod>BANK TRANSFER</paymentMethod> <authorization> <status>AUTHORIZED</status> <amount currency="EUR">3330</amount> <confidenceLevel>SHOPPER PENDING</confidenceLevel> <capture> <status>STARTED</status> <amount currency="EUR">3330</amount> </capture> </authorization>	*For each Payment registered to the Payment Order as represented by the <PaymentOrderKey>, the status details of the payment are provided in the corresponding <payment> element.*
<extended> <riskChecks score="20"> <check score="0">name.amount.velocity</check> <check score="0">amex.avs.postal.code</check> <check score="0">amex.avs.address</check> <check score="0">allowed.issuer.ip.country.combination</check> <check score="0">address.amount.velocity</check> <check score="0">bank.account.amount.velocity</check> <check score="0">email.amount.velocity</check> <check score="0">directpay.credit.check</check> <check score="0">email.velocity</check> <check score="0">3ds.issuer.liability</check> <check score="0">bank.account.white.list</check> <check score="0">creditcard.amount.velocity</check> <check score="0">shopper.country.black.list</check> <check score="0">bank.account.black.list</check> <check score="0">ip.amount.velocity</check> <check score="0">issuer.country.absence</check> <check score="0">name.velocity</check> <check score="0">bin.ranges.session</check> <check score="0">card.account.number.count.session</check> <check score="0">address.velocity</check> <check score="0">card.number.black.list</check> <check score="0">ip.black.list</check> <check score="0">zipcode.amount.velocity</check> <check score="0">3ds.merchant.liability</check> <check score="0">ip.white.list</check> <check score="20">ship.to.zipcode.housenumber.black.list</check> <check score="0">iban.velocity</check> <check score="0">zipcode.velocity</check> <check score="0">ship.to.zipcode.housenumber.velocity</check> <check score="0">amex.avs.cm.name</check> <check score="0">ip.velocity</check> <check score="0">bank.account.velocity</check> <check score="0">email.address.black.list</check> <check score="0">card.number.white.list</check> </riskChecks>	*The element <riskChecks> provides the scores of all riskchecks that are executed for the given order.*
<bankTransferPaymentInfo>	*The element <bankTransferPaymentInfo> provides*

<beneficiaryName>St. Found. docdata payments</beneficiaryName> <beneficiaryCity>Driebergen-Rijsenburg</beneficiaryCity> <beneficiaryCountry>NL</beneficiaryCountry> <bankName>Bank</bankName> <bankCity>Utrecht</bankCity> <bic>DEUTNL2A</bic> <iban>NL45DEUT0556977556</iban> </bankTransferPaymentInfo> </extended> </payment> </report> </statusSuccess> </extendedStatusResponse>	*information specific for the used payment method.*

Figure 36: Example EXTENDED STATUS Request response

Response ERROR	Explanation
<extendedStatusResponse xmlns="http://www.docdatapayments.com/services/paymentservice/1 3/" ddpXsdVersion="1.3.1"> <statusErrors> <error code="REQUEST DATA INCORRECT">XML request does not match XSD. The data is: cvc-type.3.1.3: The value '78CDE5ACE088003687F65F81A7DDEB4' of element 'paymentOrderKey' is not valid..</error> </statusErrors> </extendedStatusResponse>	*The element <statusErrors> indicates the reason of the failure. See Figure 38 for a list of possible error codes.*

Figure 37: Example STATUS Request response: The request has not been accepted for processing

Error code	Explanation
REQUEST_DATA_INCORRECT	Request data is incorrect
REQUEST_DATA_MISSING	Request data is missing
SECURITY_ERROR	Error related to security violations such as login failure or blocked IP address.
INTERNAL_ERROR	Payment system error
UNKNOWN_ERROR	The error is not specified

Figure 38: Error codes refundRequest

4.11. Changes in the Order API

This paragraph describes what changes a merchant or integrator can expect in the XML, without having to use a new version number for the XSD.

CM uses the <_any> element in the XSD as last element for responses. This allows for new elements to be added in front of the <_any> element.

A merchant / integrator can expect the following:

- Additional elements as indicated by the <_any> element in the XSD. A merchant can test his support for this element by supplying the testExtensibility attribute in the request, which causes additional <_any> elements in the response. See figure 39 and 40 for an example request and response.
- Unknown enumerated values to be returned. For example new error codes. These should not be treated as exceptional, but rather as general cases.
- Elements in requests can become optional.
- The range of occurrences of elements in requests can be increased.

- The range of occurrences of elements in responses can be decreased.
- New optional elements become available in requests.
- A different XDS version in the response then the one used in the request. The XSD version stated in the request should reflect the XSD version the merchant based the XML requests on. The XSD version stated in the response is the latest one that is active on the server. This should not be a problem, because all minor version upgrades should be backwards compatible within the major version.
- The need to update your implementation if you want to use the new elements.

4.11.1. Request

Request	Explanation
<statusRequest xmlns="http://www.docdatapayments.com/services/paymentservice/1_3/" version="1.3" **testExtensibility="true"**> <merchant name="testshop" password="wdsYFFQL"/> <paymentOrderKey>B68D771F347490D1528B396A56CF71EC</paymentOrderKey>	*Next to the version number the attribute testExtensibility is added with a value true.*
<integrationInfo> <webshopPlugin>Webshop plugin</webshopPlugin> <webshopPluginVersion>Webshop plugin version</webshopPluginVersion> <integratorName>Integrator name</integratorName> <programmingLanguage>Programming language</programmingLanguage> <operatingSystem>OS: Linux</operatingSystem> <operatingSystemVersion>OS version: 3.13.0-46-generic</operatingSystemVersion> <ddpXsdVersion>1.3.1</ddpXsdVersion> </integrationInfo> </extendedStatusRequest>	*This element contains the information about the integration:* *1.When a plugin is used the plugin and the version can be stated.* *2. When an external integrator is used the name of the integrator can be stated.* *3. The programming language* *4. The operating system and the OS version.* *5. The XSD version of CM that is used*

Figure 39: Example STATUS Request including testExtensibility attribute

4.11.2. Response

Response SUCCESS	Explanation
<statusResponse xmlns="http://www.docdatapayments.com/services/paymentservice/1_3/"> *<statusSuccess>* *<success code="SUCCESS">Operation successful.</success>*	*The element <statusSuccess> indicates that the request is accepted by the Payment system for processing the data.*
<report> <approximateTotals exchangedTo="EUR" exchangeRateDate="2015-07-14 16:12:14"> <totalRegistered>3330</totalRegistered> <totalShopperPending>3330</totalShopperPending> <totalAcquirerPending>0</totalAcquirerPending> <totalAcquirerApproved>0</totalAcquirerApproved> <totalCaptured>0</totalCaptured> <totalRefunded>0</totalRefunded> <totalChargedback>0</totalChargedback> <totalReversed>0</totalReversed> < any> any</ any> </approximateTotals>	*The element <approximateTotals> provides values which can be used to determine whether an order is considered 'paid'.*
<payment> <id>4907819251</id> <paymentMethod>BANK TRANSFER</paymentMethod> <authorization> <status>AUTHORIZED</status> <amount currency="EUR">3330</amount> <confidenceLevel>SHOPPER PENDING</confidenceLevel> <capture> <status>STARTED</status> <amount currency="EUR">3330</amount> <_any>_any</_any>	*The element <_any> indicates the places in the XML were additional elements can be added.*

```
          </capture>
          < any> any</ any>
        </authorization>
        <_any>_any</_any>
      </payment>
      < any> any</ any>
    </report>
    < any> any</ any>
  </statusSuccess>
</statusResponse>
```

Figure 40: Example STATUS Request response including <any> element

4.12. Example of an Update Url

The Status Update Notification uses a GET operation to a web shop url which is to be preconfigured in the Merchant Backoffice. An example of the Update URL shown below:

http://www.merchantwebsite.com/update_order?order_id=

At the end of the url call a "="-sign should be set in order for CM to add the specific order reference.

An example of the call from the payment service will look as follows with a merchant order reference of 33:

http://www.merchantwebsite.com/update_order?order_id=33

CM support either sends Status Update Notifications to port 80 (HTTP) or port 443 (HTTPS).

When the GET operation returns a response other than a HTTP 200 response, CM assumes the attempt to deliver the notification was not successful. In those cases CM will retry to deliver the Status Update Notification for a maximum of 10 attempts. If still not successful after 10 attempts CM assumes there is a severe problem with the web shop and will send out the Status Update notification as an email to the contact person as configured in Merchant Backoffice.

4.13. Determining whether an order is paid

Within the logic of the Payment system, payment(-attempt)s are registered to a Payment Order which is represented by a unique transaction key. The amount received by each of the payment attempts within the Payment Order, add up to the total amount which is expected by the Merchant.

4.13.1. A calculated decision

As outlined in chapter **Error! Reference source not found.** the STATUS request is used to query the status of an Order and provides the logic to determine whether an order is paid in full.

The first section in the status report as returned by the STATUS request is the element <approximateTotals> which provides information about the amount to be paid, the amount already confirmed as being received by CM and the amount still open for the shopper to pay during the lifecycle of the payment Order.

The second part of the report structure provides the actual status for each Payment(-attempt). Each Payment(-attempt) is tagged within a separate <payment> element and is referenced by its unique <paymentID>.

More detail of the different elements in the <approximateTotals> is listed in Figure 41.

Element	Explanation
<TotalRegistered>	The total amount in the currency of the order which was submitted in the CREATE request as "TotalGrossAmount". This is the amount the Merchant is expecting the shopper to pay in full before shipping products.

<TotalAcquirerApproved>	The total amount which acquirers have confirmed being paid by means of a capture. The acquirer will transfer the amount to the Merchant. Either directly to the Merchant or through CM. This amount is the total sum of all payment(-attempt)s within the Payment Order for which the <confidenceLevel> of the <authorization> is "ACQUIRER_APPROVED". `<authorization>` `  <status>AUTHORIZED</status>` `  <amount currency="EUR">320</amount>` `  <confidenceLevel>ACQUIRER_APPROVED</confidenceLevel>` `</authorization>`
<TotalAcquirerPending>	The total amount which is still pending with the acquirer to process. Although the acquirer has confirmed to have received the instruction to settle the payment there is still no guarantee that the funds are available for capture. Depending the acquirer policies and processes CM will just await the successful capture confirmation by the acquirer. This amount is the total sum of all payment(-attempt)s within the Payment Order for which the <confidenceLevel> of the <authorization> is "ACQUIRER_PENDING". `<authorization>` `  <status>AUTHORIZED</status>` `  <amount currency="EUR">865</amount>` `  <confidenceLevel>ACQUIRER_PENDING</confidenceLevel>` `</authorization>`
<TotalShopperPending>	The total amount which the shopper is expected to pay. For example in case of a bank transfer CM has not actually received the amount yet. This amount is the total sum of all payment(-attempt)s within the Payment Order for which the <confidenceLevel> of the <authorization> is "SHOPPER_PENDING". `<authorization>` `  <status>AUTHORIZED</status>` `  <amount currency="EUR">495</amount>` `  <confidenceLevel>SHOPPER_PENDING</confidenceLevel>` `</authorization>`
<TotalCaptured>	This amount is the total sum of all Payment(-attempt)s within the Payment Order for which the <status> of the <authorization> is "AUTHORIZED" and the <status> of the <capture> is "CAPTURED". `<authorization>` `  <status>AUTHORIZED</status>` `  <amount currency="EUR">360</amount>` `  <confidenceLevel>ACQUIRER_APPROVED</confidenceLevel>` `  <capture>` `  <status>CAPTURED</status>` `  <amount currency="EUR">360</amount>` `  </capture>` `</authorization>`
<TotalRefunded>	This amount is the total sum of all payment(-attempt)s within the Payment Order for which the <status> of the <authorization> is "AUTHORIZED" and the <status> of the <refund> is "CAPTURED". `<authorization>` `  <status>AUTHORIZED</status>` `  <amount currency="EUR">360</amount>` `  <confidenceLevel>ACQUIRER_APPROVED</confidenceLevel>` `  <capture>` `  <status>CAPTURED</status>` `  <amount currency="EUR">360</amount>` `  </capture>` `  <refund>` `  <status>CAPTURED</status>` `  <amount currency="EUR">120</amount>` `  </refund>` `</authorization>` Note: • In order to (partly) refund a payment(-attempt), the Payment(-attempt) should have been successfully captured first.
<TotalChargedback>	This amount is the total sum of all Payment(-attempt)s within the Payment Order for which the <status> of the <authorization> is "AUTHORIZED" and the <status> of the <chargeback> is "CAPTURED". `<authorization>` `  <status>AUTHORIZED</status>` `  <amount currency="EUR">3310</amount>` `  <confidenceLevel>ACQUIRER_APPROVED</confidenceLevel>` `  <capture>` `  <status>CAPTURED</status>` `  <amount currency="EUR">3310</amount>` `  </capture>`

	<chargeback> <chargebackId>23</chargebackId> <status>CHARGED</status> <amount currency="EUR">3310</amount> <reason>DNOR Issuing bank is not allowed to execute direct debit's.</reason> </chargeback> </authorization>
<TotalReversed>	This amount is the total sum of all Payment(-attempt)s within the Payment Order for which the <status> of the <authorization> is "AUTHORIZED" and there are <reversal> elements. <authorization> <status>AUTHORIZED</status> <amount currency="EUR">3310</amount> <confidenceLevel>ACQUIRER_APPROVED</confidenceLevel> <capture> <status>CANCELED</status> <amount currency="EUR">3330</amount> </capture> <reversal> <amount currency="EUR">3330</amount> <reason>Cancel request through order api</reason> </reversal> </authorization>

Figure 41: Details of the <approximateTotals> element as returned in the XML response of the STATUS request.

Note

- Keep in mind that the status report does not report funds actually been transferred from CM or the acquirer to the bank account of the Merchant. This is a separate process.
- Because the Payment system supports multiple currencies within a single Payment Order, the amounts in the <approximateTotals> element may be converted values. The exchange rate is based on the daily rates as published by the European Central Bank (ECB). The amounts therefore are to be considered 'calculated values', solely available to support the decision and business logic of the Merchant to determine when to ship the products. However when all currencies are just euro's all values correctly add up.

...so how should I determine whether the order is paid?

In most cases shoppers only choose one payment method and pay the amount in full. The logic to determine if an order is paid therefore is straight forward.

4.13.2. Direct Payment: Monitor <totalAcquirerApproved> equals <totalRegistered>.

In case of a Direct Payment where the acquirer acknowledges the payment by means of either a direct authorization, or authorization and capture in one go, the merchant may compare the amount <totalAcquirerApproved> against the amount <totalRegistered>. When both amounts are equal, the order is paid in full.

This logic applies to the following Payment Methods:

- iDEAL
- Creditcard
- Debitcard
- PayPal
- Klarna Invoice
- AfterPay
- Sofort Uberweisung
- Sofort e-banking
- GiroPay
- Point2Pay
- Giftcard
- EPS

When the amounts <totalAcquirerApproved> and <totalRegistered> are not equal it is up to the business logic of the Merchant to decide whether or not to ship the products.

4.13.3. Delayed Payment: Monitor <totalShopperPending> and <totalAcquirerPending>

When the Merchant offers Delayed Payments the amount reported in <totalShopperPending> will be greater than 0.

This applies to the following Payment Method:

- Bank Transfer
- SEPA Direct Debit
- ELV

The Merchant may still have confidence that the shopper will pay and decide to ship the products. However the Payment system will only confirm the amounts actually received from the shopper or acquirer.

4.13.4. Safest route: Monitor <TotalCaptured> equals <totalRegistered>.

The safest route to determine whether or not to ship products is for the Merchants to monitor the amount reported in <TotalCaptured>. When the amount <TotalCaptured> equals the amount reported in <totalRegistered>, this means that CM has confirmation from the acquirer that they will transfer the amount to the Merchant. Either directly to the Merchant or through the bank accounts of CM.

5. Backoffice configuration

In order to manage payment options, in the Merchant Backoffice add available payment methods to a profile and determine in which sequence these methods should appear in the payment menu.

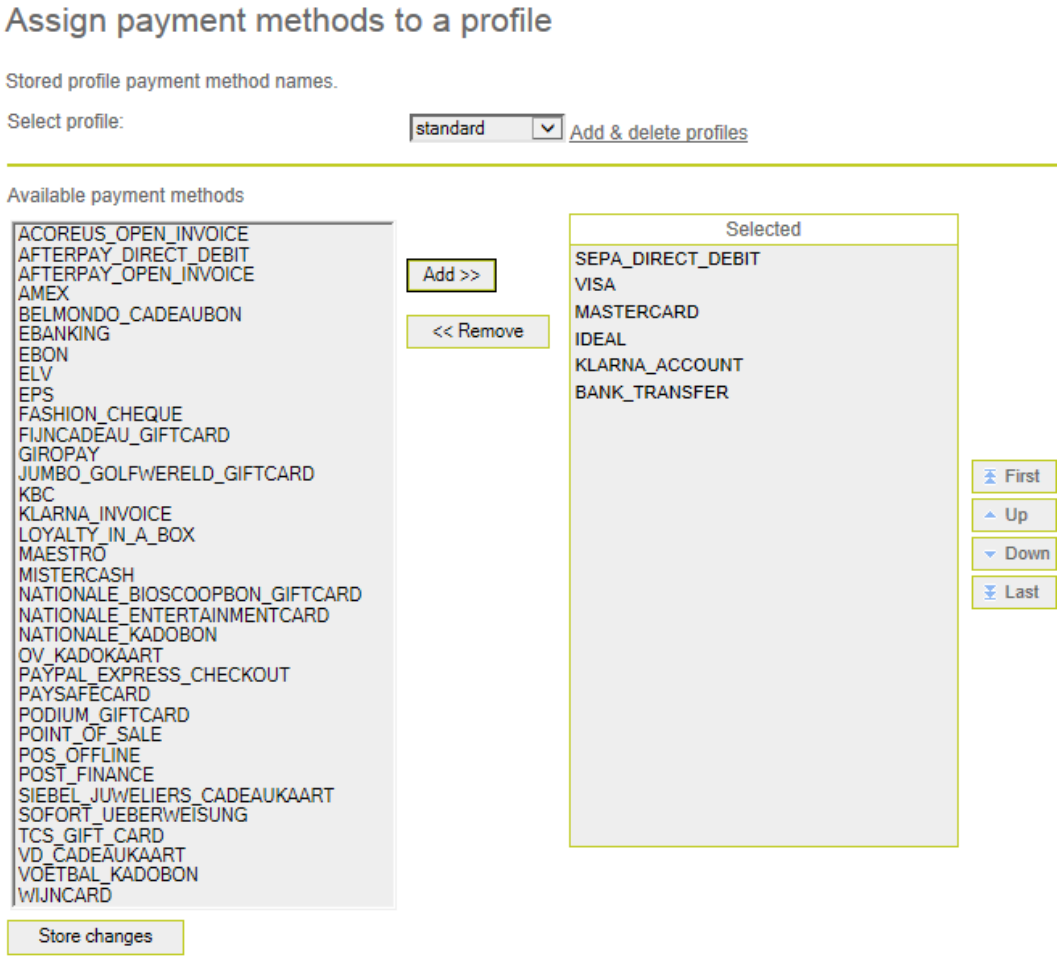

Figure 42: Assigning payment methods

To add a profile click on “Add & delete profiles”. This will open a new screen.

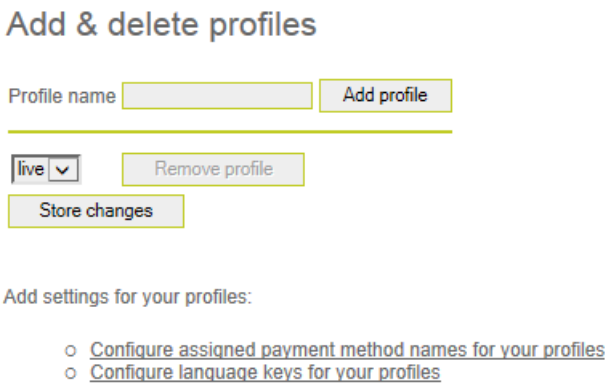

Figure 43: Profile management

Enter the profile name of your choice and save your changes by clicking the button **Store changes**. Continue by adding payment methods to the profile.

When finished return to the 'assign methods' screen, select the profile from the pull down menu and use the "add" and "remove" buttons to transfer payment methods to the box on the right hand side.

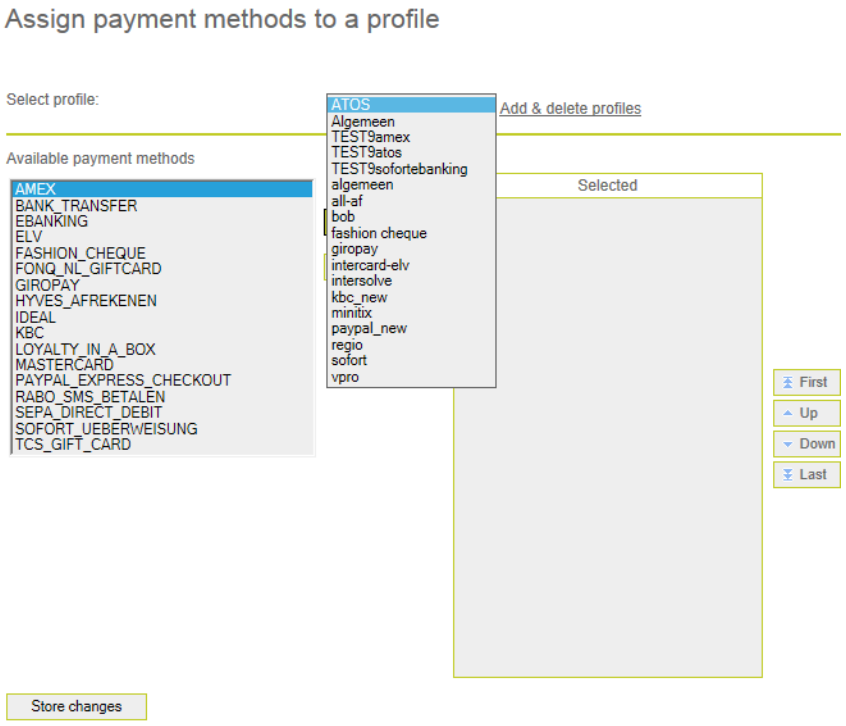

Figure 44: Profile management

Use the up and down arrow buttons First, Up, Down and Last to set the sequence of the payment methods in the payment menu.

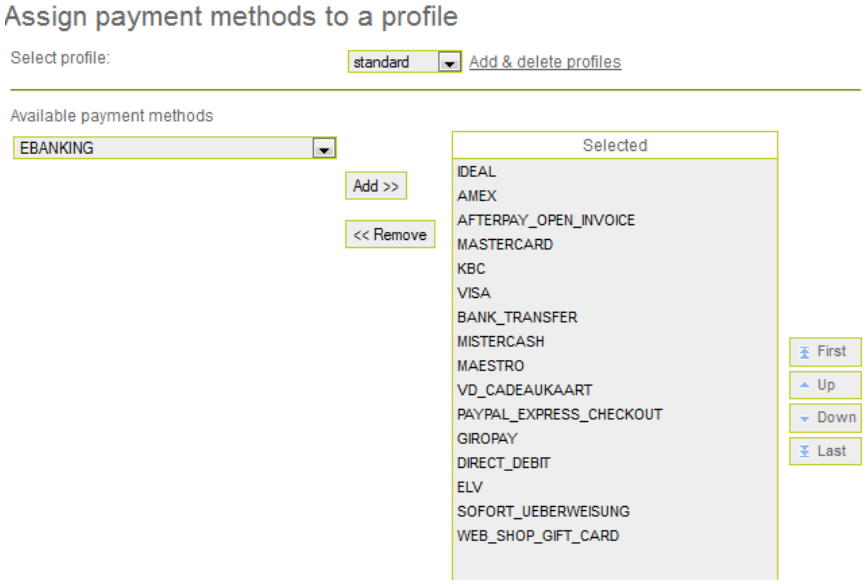

Figure 45: Profile assignment

In the above example iDEAL will be the first payment method displayed (on top) in the payment menu, followed by AMEX. Changes you apply need to be stored by the "Store changes" button.