Handleiding SDK

Disclaimer: Deze Handleiding is uitsluitend bedoeld voor derden die voor en in opdracht van klanten zorgdragen voor de technische koppeling van Rabo OmniKassa met de webshop van de klant. Rabobank is niet aansprakelijk voor mogelijke schade die voortvloeit uit fouten in of verkeerd gebruik van de Handleiding. Bijvoorbeeld maar niet beperkt tot schade die ontstaat doordat de technische koppeling van Rabo OmniKassa met de webshop van de klant niet (geheel) correct werkt. Rabobank heeft het recht deze Handleiding te wijzigen.

Versie: 1.1

Vereisten

Om gebruik te maken van de SDKs moet u aan de volgende minimale eisen voldoen:

1. Indien u gebruik maakt van de Java SDK dient u de beschikking te hebben over Java Runtime Environment versie 1.7 of hoger.
2. Indien u gebruik maakt van de PHP SDK dient u de beschikking te hebben over PHP versie 5.4.0 (of hoger tot en met versie 5.6.30).
3. Om een versleutelde verbinding te kunnen opzetten met Rabobank OmniKassa dient de Java Runtime Environment of PHP (afhankelijk van de gekozen SDK) geconfigureerd te zijn met een CA bundle waarmee de geldigheid van het certificaat van https://betalen.rabobank.nl kan worden gecontroleerd. In de regel is dit al het geval en hoeft u niets te doen. Echter, vanuit beveiligingsoogpunt strekt het tot aanbeveling om een minimale CA bundle te hanteren waarmee dit certificaat gevalideerd wordt.
   Indien u PHP 5.6.x onder Windows draait kan het nodig zijn dat u in het bestand php.ini de property openssl.cafile configureert met het pad naar de CA bundle. Raadpleeg de PHP documentatie voor meer informatie.

Introductie

Dit document beschrijft hoe een webshop aangesloten kan worden op de Rabo OmniKassa met behulp van de beschikbare SDKs. Op dit moment levert de Rabobank SDKs voor de volgende programmeertalen: PHP en Java.

Betaalstappen

De betaalstappen bestaan uit een 3 tal aanroepen van de webwinkel naar de Rabo OmniKassa, en 1 aanroep van de Rabo OmniKassa naar de webwinkel. De SDK helpt hierbij om deze stappen eenvoudiger te maken. De betaalstappen zijn uitgewerkt in de onderstaande hoofdstukken. Voor een meer high level uitleg van de stappen van het betaalproces: zie de Handleiding Rabo OmniKassa.

1 Betaalverzoek versturen

1.1 Klaarzetten order

Om een betaalverzoek te doen moet er eerst een order klaargezet worden. In de order staat alle informatie over het betaalverzoek die de Rabo OmniKassa nodig heeft om de consument door de betaalstappen heen te leiden. In de volgende codeblokken staan voorbeelden om een order aan te maken. Aan de order zijn eisen gesteld en wanneer deze hier niet aan voldoet wordt deze in principe afgekeurd.

PHP

```
use nl\rabobank\gict\payments_savings\omnikassa_sdk\model\Money;
use nl\rabobank\gict\payments_savings\omnikassa_sdk\model\OrderItem;
use nl\rabobank\gict\payments_savings\omnikassa_sdk\model\PaymentBrand;
use
nl\rabobank\gict\payments_savings\omnikassa_sdk\model\PaymentBrandForce;
use nl\rabobank\gict\payments_savings\omnikassa_sdk\model\ProductType;
use
nl\rabobank\gict\payments_savings\omnikassa_sdk\model\request\MerchantOrde
r;
use nl\rabobank\gict\payments_savings\omnikassa_sdk\model\ShippingDetails;

$merchantOrderId = '100';
$description = 'Order ID: ' . $merchantOrderId;
$itemAmount = Money::fromDecimal('EUR', 99.99);
$itemTax = Money::fromDecimal('EUR', 4.99);
$orderItems = [new OrderItem('Test product', 'Description', 1, $itemAmount,
$itemTax, ProductType::PHYSICAL)];
$amount = Money::fromDecimal('EUR', 99.99);
$shippingDetails = new ShippingDetails('Jan', 'van', 'Veen',
'Voorbeeldstraat 5', '1234AB', 'Haarlem', 'NL');
$language = 'NL';
$merchantReturnUrl = 'http://localhost/';
$paymentBrand = PaymentBrand::IDEAL;
$paymentBrandForce = PaymentBrandForce::FORCE_ONCE;

$order = new MerchantOrder($merchantOrderId, $description, $orderItems,
$amount, $shippingDetails, $language, $merchantReturnUrl, $paymentBrand,
$paymentBrandForce);
```

Java

```
Money itemAmount = Money.fromEuros(Currency.EUR, new BigDecimal("99.99"));
Money itemTax = Money.fromEuros(Currency.EUR, new BigDecimal("4.99"));
OrderItem orderItem = new OrderItem(1, "Test product", "Description",
itemAmount, itemTax, ItemCategory.PHYSICAL);

ShippingDetail shippingDetail = new ShippingDetail("Jan",
                                                   "van",
                                                   "Veen",
                                                   "Voorbeeldstraat 5",
                                                   "1234AB",
                                                   "Haarlem",
                                                   CountryCode.NL);

MerchantOrder order = new MerchantOrder.Builder()
        .withMerchantOrderId("ORDID123")
        .withDescription("An example description")
        .withOrderItems(Collections.singletonList(orderItem))
        .withAmount(Money.fromEuros(Currency.EUR, new BigDecimal("99.99")))
        .withShippingDetail(shippingDetail)
        .withLanguage(Language.NL)
        .withMerchantReturnURL("http://localhost/")
        .withPaymentBrand(PaymentBrand.IDEAL)
        .withPaymentBrandForce(PaymentBrandForce.FORCE_ONCE)
        .build();
```

1.2 Veldbeschrijving

Hieronder staan alle velden beschreven met de naam, een omschrijving, en de regels waaraan het veld moet voldoen.

`MerchantOrder`

Veld	Omschrijving	Regel
`merchantOrderId`	De identiteit van de order	Mag niet `null` of leeg zijn
		Mag alleen bestaan uit alfanumerieke karakters
		Heeft een maximale lengte van 10 karakters
`description`	Een omschrijving van de order	Optioneel
		Heeft een maximale lengte van 35 karakters
`orderItems`	Alle items die besteld gaan worden door de consument	Optioneel
		Er mogen maximaal 99 items meegeleverd worden
		De valuta van elk item moet gelijk zijn aan de valuta van het `amount` van de `MerchantOrder`
`shippingDetails`	Het afleveringsadres van deze order	Optioneel
`amount`	Het totaal bedrag van de order	Mag niet `null` zijn
		Het bedrag mag niet hoger zijn dan 99.999,99

`language`	De teksten op de betaalpagina's zullen in deze taal zijn	Optioneel, standaard hanteren we NL.
		ISO 3166-1 alpha-2, beperkt tot NL, EN, FR en DE.
		Als `NL`, `EN`, `FR`, of `DE` meegeleverd wordt: dan hanteren we `NL`, `EN`, `FR`, of `DE` respectievelijk
		Als een onbekende waarde wordt meegeleverd: dan hanteren we `EN`
`merchantReturnURL`	De `URL` waar de consument naar terug zal keren nadat de betaalstappen afgerond zijn	Mag niet `null` of leeg zijn
		Moet een valide `URL` zijn
`paymentBrand`	De betaalmethode waar de consument tot gelimiteerd wordt	Optioneel
		Moet één van de volgende waardes zijn: `IDEAL`, `PAYPAL`, `MASTERCARD`, `VISA`, `BANCONTACT`, `MAESTRO`, `V_PAY` Wanneer waarde `CARDS` meegeleverd wordt, dan worden alle kaart betaalmethoden aangeboden (MasterCard, Visa, Bancontact, Maestro, en V Pay)
`paymentBrandForce`	Manier waarop de betaalmethode geforceerd moet worden	Verplicht indien `paymentBrand` meegegeven is, anders optioneel
		Moet één van de volgende waardes zijn: `FORCE_ONCE` of `FORCE_ALWAYS`

De betaalmethode en de forceer opties werken als volgt. Wanneer de betaalmethode op `IDEAL` staat, en de forceer optie `FORCE_ONCE` is gekozen, dan betekent dit voor de consument dat zij bij aankomst op de Rabobank OmniKassa, gelijk een iDEAL betaling start en dus op het bank keuzen scherm aankomt. Zij heeft dan de mogelijkheid om de betaling af te ronden **of** voor een andere betaalmethode te kiezen door te klikken op `Kies andere betaalmethode`. Bij de optie `FORCE_ALWAYS` is het niet mogelijk voor de consument om een andere betaalmethode te kiezen en zijn de enige opties om het betaalverzoek goed te keuren of annuleren.

`Money`

Veld	Omschrijving	Regels
`currency`	De valuta	Mag niet `null` zijn
		Moet één van de volgende valuta's zijn: `AUD`, `CAD`, `CHF`, `DKK`, `EUR`, `GBP`, `JPY`, `NOK`, `SEK`, `USD`
`amount`	Het bedrag	Mag niet `null` zijn
		Moet een natuurlijk getal zijn

Het veld `currency` beïnvloedt welke betaalmethodes beschikbaar zullen zijn voor de consument. Dit komt omdat niet alle betaalmethodes alle verschillende valuta's ondersteunen. In onderstaande tabel staan **alleen** de betaalmethodes die niet alle valuta's ondersteunen.

Betaalmethode	Ondersteunde valuta's
iDEAL	EUR
Bancontact	EUR
PayPal	EUR

Om een `Money` instantie aan te maken kan gebruik gemaakt worden van de volgende code

PHP

```
//1,75 Euro
$moneyFromCents = Money::fromCents('EUR', 175);
//75,99 Euro
$moneyFromDecimal = Money::fromDecimal('EUR', 75.99);

//Rounding examples
//1000 Euro cents
$amountInCents = Money::fromDecimal('EUR', 9.999)->getAmount();
//999 Euro cents
$amountInCents = Money::fromDecimal('EUR', 9.991)->getAmount();
//1000 Euro cents
$amountInCents = Money::fromDecimal('EUR',9.995)->getAmount();
```

Java

```
//1,75 Euro
Money moneyFromEuros = Money.fromEuros(Currency.EUR,
BigDecimal.valueOf(175, 2));
Money moneyFromString = Money.fromEuros(Currency.EUR, new
BigDecimal("1.75"));
```

`ShippingDetails`

Veld	Omschrijving	Regels
`firstName`	Voornaam van de consument	Mag niet `null` of leeg zijn
		Mag alleen bestaan uit alfabet karakters
		Heeft een maximale lengte van `50` karakters
`middleName`	Tussenvoegsel van de consument	Optioneel
		Mag alleen bestaan uit alfabet karakters
		Heeft een maximale lengte van `20` karakters
`lastName`	Achternaam van de consument	Mag niet `null` of leeg zijn
		Mag alleen bestaan uit alfabet karakters
		Heeft een maximale lengte van `50` karakters
`street`	Straatnaam en huisnummer van het afleveradres	Mag niet `null` of leeg zijn
		Mag alleen bestaan uit alfabet karakters
		Heeft een maximale lengte van `100` karakters
`postalCode`	Postcode van het afleveradres	Mag niet `null` of leeg zijn
		Moet overeenkomen met het postcode formaat* van de gekozen `countryCode`

`city`	Stad van het afleveradres	Mag niet `null` of leeg zijn
		Mag alleen bestaan uit alfabet karakters
		Heeft een maximale lengte van `40` karakters
`countryCode`	Landcode van het afleveradres	Mag niet `null` of leeg zijn
		Moet één van de volgende waardes zijn: `AF, AX, AL, DZ, VI, AS, AD, AO, AI, AQ, AG, AR, AM, AW, AU, AZ, BS, BH, BD, BB, BE, BZ, BJ, BM, BT, BO, BQ, BA, BW, BV, BR, VG, IO, BN, BG, BF, BI, KH, CA, CF, CL, CN, CX, CC, CO, KM, CG, CD, CK, CR, CU, CW, CY, DK, DJ, DM, DO, DE, EC, EG, SV, GQ, ER, EE, ET, FO, FK, FJ, PH, FI, FR, TF, GF, PF, GA, GM, GE, GH, GI, GD, GR, GL, GP, GU, GT, GG, GN, GW, GY, HT, HM, HN, HU, HK, IE, IS, IN, ID, IQ, IR, IL, IT, CI, JM, JP, YE, JE, JO, KY, CV, CM, KZ, KE, KG, KI, UM, KW, HR, LA, LS, LV, LB, LR, LY, LI, LT, LU, MO, MK, MG, MW, MV, MY, ML, MT, IM, MA, MH, MQ, MR, MU, YT, MX, FM, MD, MC, MN, ME, MS, MZ, MM, NA, NR, NL, NP, NI, NC, NZ, NE, NG, NU, MP, KP, NO, NF, UG, UA, UZ, OM, AT, TL, PK, PW, PS, PA, PG, PY, PE, PN, PL, PT, PR, QA, RE, RO, RU, RW, BL, KN, LC, PM, VC, SB, WS, SM, SA, ST, SN, RS, SC, SL, SG, SH, MF, SX, SI, SK, SD, SO, ES, SJ, LK, SR, SZ, SY, TJ, TW, TZ, TH, TG, TK, TO, TT, TD, CZ, TN, TR, TM, TC, TV, UY, VU, VA, VE, AE, US, GB, VN, WF, EH, BY, ZM, ZW, ZA, GS, KR, SS, SE, CH`

De ondersteunde postcode formaten zijn als volgt. De overige landcodes hebben alleen een maximale lengte van 10 en staan in onderstaande tabel aangegeven als 'Anders'.

Landcode	Formaat	Maximale lengte
`BE`	`\p{Digit}+`	4
`DE`	`\p{Digit}+`	5
`NL`	`\p{Digit}{4}\p{Alpha}{2}`	6
Anders	N.v.t.	10

`OrderItem` regels

Veld	Omschrijving	Regels
`name`	Naam van het item	Mag niet `null` of leeg zijn
		Mag alleen bestaan uit alfabet karakters
		Heeft een maximale lengte van `50` karakters
`description`	Een omschrijving van het item	Mag niet `null` of leeg zijn
		Heeft een maximale lengte van `100` karakters
`quantity`	Hoevaak dit item besteld wordt	Mag niet `null` of leeg zijn
		Moet een natuurlijk getal zijn
`amount`	Het bedrag per stuk, inclusief BTW	Mag niet `null` zijn
`tax`	BTW per stuk	Optioneel
		Moet dezelfde valuta zijn als het `amount`
`category`	Categorie van dit item	Mag niet `null` zijn
		Moet één van de volgende waardes zijn: `DIGITAL`, `PHYSICAL`

De `quantity` beschrijft hoeveel de consument van het product wil hebben, bijvoorbeeld 2 appels, of 3 peren. Het `amount` geeft aan hoeveel **één** product kost, dus 1 appel kost €1,50, of 1 peer kost €1,75, en is **inclusief** BTW. Als we de voorbeelden uitschrijven dan wordt het.

PHP

```
$apples = new OrderItem('Apple', 'A delicious apple', 2,
Money::fromDecimal('EUR', 1.50), Money::fromDecimal('EUR', 0.11),
ProductType::PHYSICAL);
$pears = new OrderItem('Pear', 'A delicious pear', 3,
Money::fromDecimal('EUR', 1.75), Money::fromDecimal('EUR', 0.18),
ProductType::PHYSICAL);
```

Java

```
OrderItem apples = new OrderItem(2, "Apple", "A delicious apple",
Money.fromEuros(Currency.EUR, BigDecimal.valueOf("1.50")),
Money.fromEuros(Currency.EUR, BigDecimal.valueOf("0.11")),
ItemCategory.PHYSICAL);
OrderItem pears = new OrderItem(3, "Pear", "A delicious pear",
Money.fromEuros(Currency.EUR, BigDecimal.valueOf("1.75")),
Money.fromEuros(Currency.EUR, BigDecimal.valueOf("0.18")),
ItemCategory.PHYSICAL);
```

Het `amount` van de `MerchantOrder` moet, indien de `orderItems` gevuld worden, overeenkomen met de som van de `orderItems`. Als geen o `rderItems` meegeleverd worden mag elk bedrag, binnen de opgestelde regels, ingevuld worden. Je kan het totaal als volgt berekenen.

PHP

```
$orderTotalInCents = 0;
for($orderItems as $orderItem){
 $priceTaxInclusiveInCents = $orderItem->getAmount()->getAmount();
 $orderTotalInCents += $priceTaxInclusiveInCents *
$orderItem->getQuantity();
}
$orderAmount = Money::fromCents('EUR', $orderTotalInCents);
```

Java

```
BigDecimal orderTotal = BigDecimal.ZERO;
for(OrderItem orderItem : orderItems) {
 BigDecimal amount = orderItem.getAmount().getAmount();
 BigDecimal quantity = new BigDecimal(orderItem.getQuantity());
 orderTotal = orderTotal.add(amount.multiply(quantity));
}
Money orderAmount = Money::fromEuros(Currency.EUR, orderTotal);
```

Mocht het zijn dat er toch een incorrecte order wordt opgestuurd en de Rabo OmniKassa het kan repareren, dan wordt er gepoogd om de order toch te corrigeren door eventuele gegevens weg te halen of in te korten. Dit is alleen in de uiterste gevallen van toepassing.

1.3 `Endpoint` aanmaken

Naast een order is ook een instantie van een `Endpoint` nodig. Met deze `Endpoint` is het mogelijk om alle aanroepen richting de Rabo OmniKassa uit te voeren. Een `Endpoint` heeft 3 dingen nodig: een `URL` om mee te verbinden; een `Base64` gedecodeerde `SigningKey`; en een `TokenProvider` implementatie.

De URL is:

Omgeving	URL	Omschrijving
Live	https://betalen.rabobank.nl/omnikassa-api/	In deze omgeving worden alle betalingen afgeschreven van consumenten en overgemaakt naar de eigenaar van de webshop

De omgevingen zijn te configureren in het Rabo OmniDashboard - bijvoorbeeld welke betaalmethodes aangeboden moeten worden.

De `SigningKey`, of signeersleutel, is een geheime sleutel die terug te vinden is in het Rabo OmniDashboard. Deze signeersleutel is Base64 geëncodeerd en moet gedecodeerd doorgegeven worden aan de `Endpoint`. De signeersleutel wordt gebruikt om alle berichten van de aanroepen te versleutelen van de Rabo OmniKassa. Dit is een extra beveiligingslaag om aan te tonen dat berichten wel degelijk van de webshop afkomen en niet van eventuele hackers.

PHP

```
$signingKey = new SigningKey(base64_decode('Z2VoZWltZSBzbGV1dGVs'));
```

Java

```
byte[] signingKey = Base64Utils.decodeFromString("Z2VoZWltZSBzbGV1dGVs");
```

Als laatste hebben we een implementatie van de `TokenProvider` nodig. Deze implementatie is verantwoordelijk voor het aanleveren en opslaan van token informatie. Een implementatie kan bijvoorbeeld de gegevens opslaan in een database, op een file systeem, of in het geheugen gezet worden. Het is de bedoeling dat de eigen `TokenProvider` standaard een refresh token aanlevert - deze moet op te halen zijn voordat de eerste aanroep gedaan wordt. Dit token is te vinden in het Rabo OmniDashboard en is een uniek token, met lange houdbaarheid, die gebruikt wordt om een access token aan te vragen bij de Rabo OmniKassa. Het access token is een token met een korte houdbaarheid waarmee alle aanroepen worden geauthenticeerd. Dit token wordt aangeleverd bij de `TokenProvider` voor opslag en eveneens om uit te lezen.

PHP

```php
<?php

use
nl\rabobank\gict\payments_savings\omnikassa_sdk\connector\TokenProvider;

class InMemoryTokenProvider extends TokenProvider
{
    private $map = array();

    /**
     * Construct the in memory token provider with the given refresh token.
     * @param string $refreshToken The refresh token used to retrieve the
access tokens with.
     */
    public function __construct($refreshToken)
    {
        $this->setValue('REFRESH_TOKEN', $refreshToken);
    }

    /**
     * Retrieve the value for the given key.
     *
     * @param string $key
     * @return string Value of the given key or null if it does not exists.
     */
    protected function getValue($key)
    {
        return array_key_exists($key, $this->map) ? $this->map[$key] :
null;
    }

    /**
     * Store the value by the given key.
     *
     * @param string $key
     * @param string $value
     */
    protected function setValue($key, $value)
    {
        $this->map[$key] = $value;
    }

    /**
     * Optional functionality to flush your systems.
     * It is called after storing all the values of the access token and
can be used for example to clean caches or reload changes from the
database.
     */
    protected function flush()
    {
    }
}
```

Java

```
public class InMemoryTokenProvider extends TokenProvider {
    private Map<FieldName, String> map = new HashMap<>();

    public InMemoryTokenProvider(String refreshToken) {
        setValue(FieldName.REFRESH_TOKEN, refreshToken);
    }

    @Override
    public String getValue(FieldName key) {
        return map.get(key);
    }

    @Override
    public void setValue(FieldName key, String value) {
        map.put(key, value);
    }
}
```

Met deze gegevens kan dan als volgt een `Endpoint` aangemaakt worden.

PHP

```
$rokServerUrl = 'https://betalen.rabobank.nl/omnikassa-api/';
$signingKey = new SigningKey(base64_decode('Z2VoZWltZSBzbGV1dGVs'));
$inMemoryTokenProvider = new InMemoryTokenProvider('eyJra...PLFozk');
$endpoint = Endpoint::createInstance($rokServerUrl, $signingKey,
$inMemoryTokenProvider);
```

Java

```
String rokServerUrl = "https://betalen.rabobank.nl/omnikassa-api/";
byte[] signingKey = Base64Utils.decodeFromString("Z2VoZWltZSBzbGV1dGVs");
TokenProvider inMemoryTokenProvider = new
InMemoryTokenProvider("eyJra...PLFozk");
Endpoint endpoint = Endpoint.createInstance(rokServerUrl, signingKey,
inMemoryTokenProvider);
```

1.4 Versturen order

Met dit `Endpoint` kunnen we de order aankondigen en wordt de URL van de Rabo OmniKassa, waar de gebruiker de order kan betalen, terug gestuurd.

PHP

```
$redirectUrl = $endpoint->announceMerchantOrder($order);
//Redirect user to Rabo OmniKassa
header('Location: ' . $redirectUrl);
die();
```

Spring MVC - Java

```
String redirectUrl = endpoint.announceMerchantOrder(order);
//Redirect user to Rabo OmniKassa
return "redirect: " + redirectUrl;
```

2. Consument betaalt de order

In de Rabo OmniKassa kan de consument de order betalen. Wanneer dit gedaan is komt de consument terug bij de webwinkel via de `merchantReturnUrl` die opgegeven is tijdens het klaarzetten van de order. Met deze `URL` worden een aantal `GET` parameters meegestuurd die over de order gaan namelijk:

Parameter	Omschrijving
order_id	Dit is het bestelnummer van de order.
status	De huidig bekende status van de order.
signature	Met de handtekening is het mogelijk om te controleren dat de huidige aanroep authentiek is.

Om te controleren dat de handtekening authentiek is kan de volgende code gebruikt worden. Het is eveneens aan te raden om gebruik te maken van de `PaymentCompletedResponse` class zodat de `order_id`, `status`, en `signature` geschoond worden als zij invalide / onveilige waardes bevatten.

PHP

```
$orderId = $_GET['orderId'];
$status = $_GET['status'];
$signature = $_GET['signature'];
$signingKey = new SigningKey(base64_decode('Z2VoZWltZSBzbGV1dGVs'));
$paymentCompletedResponse =
PaymentCompletedResponse::createInstance($orderId, $status, $signature,
$signingKey);
if (!$paymentCompletedResponse) {
    throw new Exception('The payment completed response was invalid.');
}

// Use these variables instead of using the URL parameters ($orderId and
$status). Input validation has been performed on these values.
$validatedMerchantOrderId = $paymentCompletedResponse->getOrderID();
$validatedStatus = $paymentCompletedResponse->getStatus();

// ... complete payment
```

Java

```
void paymentCompleted(HttpServletRequest request) {
 byte[] signingKey = Base64Utils.decodeFromString("Z2VoZWltZSBzbGV1dGVs");
    String orderId = request.getParameter("order_id");
    String status = request.getParameter("status");
    String signature = request.getParameter("signature");
    PaymentCompletedResponse paymentCompletedResponse = new
PaymentCompletedResponse(orderId, status, signature);
 try {
  paymentCompletedResponse.validateSignature(signingKey);
 } catch (RabobankSdkException invalidSignatureException){
  throw new IllegalStateException("The payment completed response was
invalid.", invalidSignatureException);
 }

    // Use these variables instead of using the URL parameters (orderId and
status). Input validation has been performed on these values.
 String validatedOrderId = paymentCompletedResponse.getOrderID();
 String validatedStatus = paymentCompletedResponse.getStatus();

 //... complete payment
}
```

Mocht het antwoord invalide zijn, dan is het raadzaam om de gebruiker door te verwijzen naar een foutpagina maar de order als 'open' te beschouwen. In een later stadium kan door middel van notificaties de daadwerkelijke status van de order opgevraagd worden.

3. Updates ontvangen over orders

Alle status overgangen van een order worden bijgehouden door de Rabo OmniKassa zodat deze aangeboden kunnen worden aan de webwinkel met behulp van notificaties. De Rabo OmniKassa doet dit door een notificatie te sturen naar de `webhookUrl` via een `POST` aanvraag met de notificatie als JSON meegestuurd. De `webhookUrl` kan geconfigureerd worden in het Rabo OmniDashboard.

Een notificatie ziet er als volgt uit:

Notification

```
{
  "authentication":
"eyJraWQiOiJHS0wiLCJhbGciOiJFUzI1NiJ9.eyJwayMiOjUwMiwiY2lkIjoiYzhjYy1mMThj
IiwiZXhwIjoxNDc5MTIyODc2fQ.MEUCIQC2Z5WUVTAKcBHISsOVMJIJE8PAbVe5x1ior4bgrTc
gCwIgLNoVIWEmSbQekJTccM89sosAY-8JzN47DGjvdPGdF0w",
  "expiry": "2016-11-25T09:53:46.765+01:00",
  "eventName": "merchant.order.status.changed",
  "poiId": "123"
}
```

Met deze JSON is het mogelijk om een `AnnouncementResponse` op te bouwen waarmee de daadwerkelijke informatie van de orders opgehaald kan worden.

PHP

```
$json = $this->getJsonFromRequest();
$signingKey = new SigningKey(base64_decode('Z2VoZWltZSBzbGV1dGVs'));
$announcementResponse = new AnnouncementResponse($json, $signingKey);
$endpoint = Endpoint::createInstance(...);

do {
    $response = $endpoint->retrieveAnnouncement($announcementResponse);

 //... Update the order statuses
} while ($response->isMoreOrderResultsAvailable());
```

Java

```
@RequestMapping(value = "/webhook", method = RequestMethod.POST)
ResponseEntity webhook(@RequestBody ApiNotification apiNotification) throws
RabobankSdkException {
 byte[] signingKey = Base64Utils.decodeFromString("Z2VoZWltZSBzbGV1dGVs");
 apiNotification.validateSignature(signingKey);
 Endpoint endpoint = Endpoint.createInstance(...);

 MerchantOrderStatusResponse merchantOrderStatusResponse;
 do {
     merchantOrderStatusResponse =
endpoint.retrieveAnnouncement(apiNotification);

  //... Update the order statuses
 } while (merchantOrderStatusResponse.moreOrderResultsAvailable());
 return new ResponseEntity(HttpStatus.OK);
}
```

Ook bij deze stap wordt de handtekening van de notificatie gecontroleerd. Als dit fout gaat kan het zijn dat bijvoorbeeld een nieuwe signeersleutel gegenereerd is in het Rabo OmniDashboard en deze nog niet verwerkt is door alle systemen. De Rabo OmniKassa probeert het dan op een later moment nog een keer.