COLLECTING SOLUTION

Hosted Payment Page

Implementation Guide

Document version 3.39

Contents

1. HISTORY OF THE DOCUMENT

Version	Author	Date	Comment
3.39	Lyra Collect	9/9/2024	• Updated the values in the chapter *Authorization request validity period*. • Update of the chapter *Processing the response data*
3.38.2	Lyra Collect	8/1/2024	Data dictionary removed from this guide.
3.38.1	Lyra Collect	7/1/2024	• Updated the chapter *Preparing your environment*. Data dictionary: • Updated field values for**vads_acquirer_network**,**vads_contracts**,**vads_payment_cards**.
3.38	Lyra Collect	3/5/2024	Chapter(s) updated: • Cascading payment • Analyzing the nature of the notification Data dictionary: • **vads_submerchant_company_type** • **vads_url_check_src**
3.37.1	Lyra Collect	2/13/2024	• Updated the chapter *Defining the steps of the payment process - As seen by the buyer* Data dictionary: • **vads_threeds_mpi**

2. DEFINITIONS

2.1. Authorization request

An authorization request allows to accept or refuse a transaction.

It connects the holder's bank (SAE = Issuer Acceptance System) with the merchant's bank (SAA = Acquirer Acceptance System) and the payment gateway.

When an authorization request is accepted, the authorization limit of the card is lowered by the authorized amount.

In the CB network, an accepted authorization request is valid:

- 7 days for Visa, Mastercard, Visa Electron, e-Carte Bleue and Vpay cards
- 30 days for Maestro cards

2.2. Information request

An information request checks the validity of the card without debiting it.

It is a specific authorization request with an amount of 0 EUR.

If the acquirer does not support this request, a 1 EUR authorization request is made without captured by the bank.

Holders of prepaid and immediate debit cards will see a virtual debit of EUR 1 on their account.

The amount is restored when the issuer cancels the authorization, which can take up to 30 days for debit cards.

Information request are sent in the following cases:

- Deferred payment, the capture date is beyond the authorization lifespan,
- Creation or update of a card token without payment.

These operations appear in the Expert Back Office as a "Verification" type transaction.

2.3. Chaining of CIT/MIT transactions

The second Payment Services Directive (PSD2) requires cardholder authentication for any e-commerce transaction initiated by the cardholder.

It is essential to identify whether the payment request is initiated:

- **CIT (Customer Initiated Transaction)** : Buyer-initiated transaction with buyer interaction.

  Example: Payment or card registration requiring data entry.

- **MIT (Merchant Initiated Transaction)** : Transaction initiated by the merchant, without the presence of the buyer, linked to an CIT transaction.

  Example: Installment of an payment in installments.

Operation chaining:

For a **CIT** transaction, regulations require cardholder authentication. After the request for authorization or information, the issuer returns a unique identifier, called the "chaining reference". This reference is used in the

MIT transactions to inform the issuer that the transaction is part of a series of payments, for which the cardholder authenticated in the first payment.

Without this reference, the issuer can decline an MIT transaction due to lack of authentication (soft decline).

3. DIFFERENT TYPES OF PAYMENTS

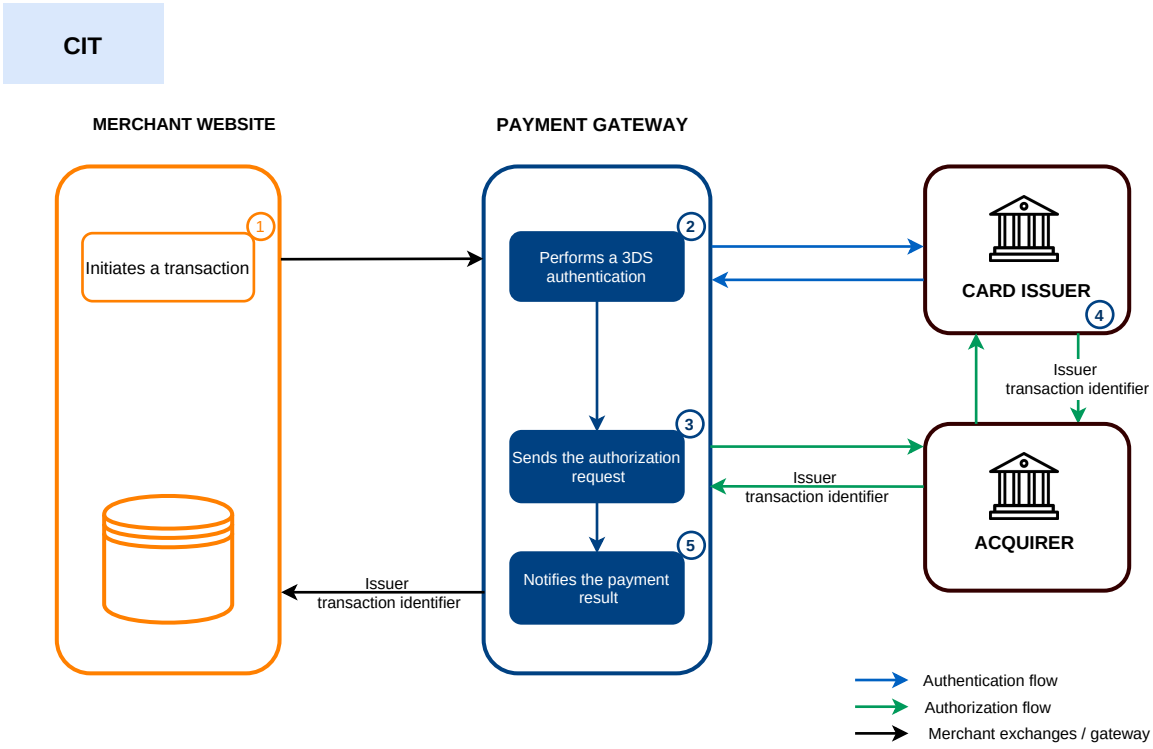
3.1. Immediate payment

A payment is considered as immediate payment if:

- The amount is debited once,
- The capture delay at the bank is 0 days.

The payment is captured at the bank as soon as possible.

Simplified diagram



1. The merchant site submits a payment request.
2. The payment gateway initiates the cardholder's authentication process with the issuer (mandatory for all CIT transactions).
3. After the authentication (challenge or frictionless), the gateway proceeds with the authorization request by providing the cardholder's authentication details.
4. The issuer generates and transmits a unique transaction identifier in their response.
5. The payment gateway notifies the merchant website about the payment result.

The payment gateway stores the issuer transaction identifier for each transaction.

If the merchant duplicates the transaction (MIT), the payment gateway uses this identifier as a chaining reference.

The chaining reference management is automatic and transparent for the merchant.

3.2. Deferred payment

A payment is considered a deferred payment if:

- The amount is debited once,
- The capture delay is strictly more than 0 days.

  The capture date cannot be more than 12 months after the payment request registration date.

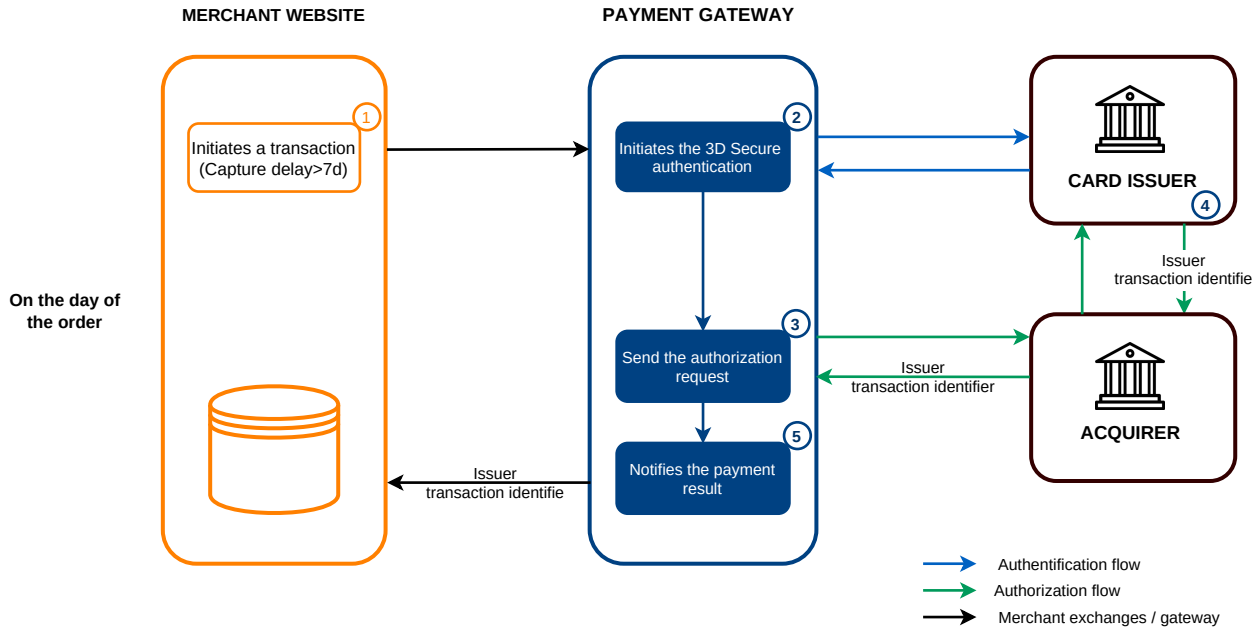
There are two types of deferred payments:

1. **Capture delay shorter than the authorization validity period** (see: Authorization request validity period on page 22)
2. **Capture delay longer than the authorization validity period** (see: Authorization request validity period on page 22)

3.2.1. Capture delay shorter than the authorization validity period

Simplified diagram

CIT



On the day of the order:

1. The merchant site submits a payment request.
2. The payment gateway initiates the cardholder's authentication process with the issuer.

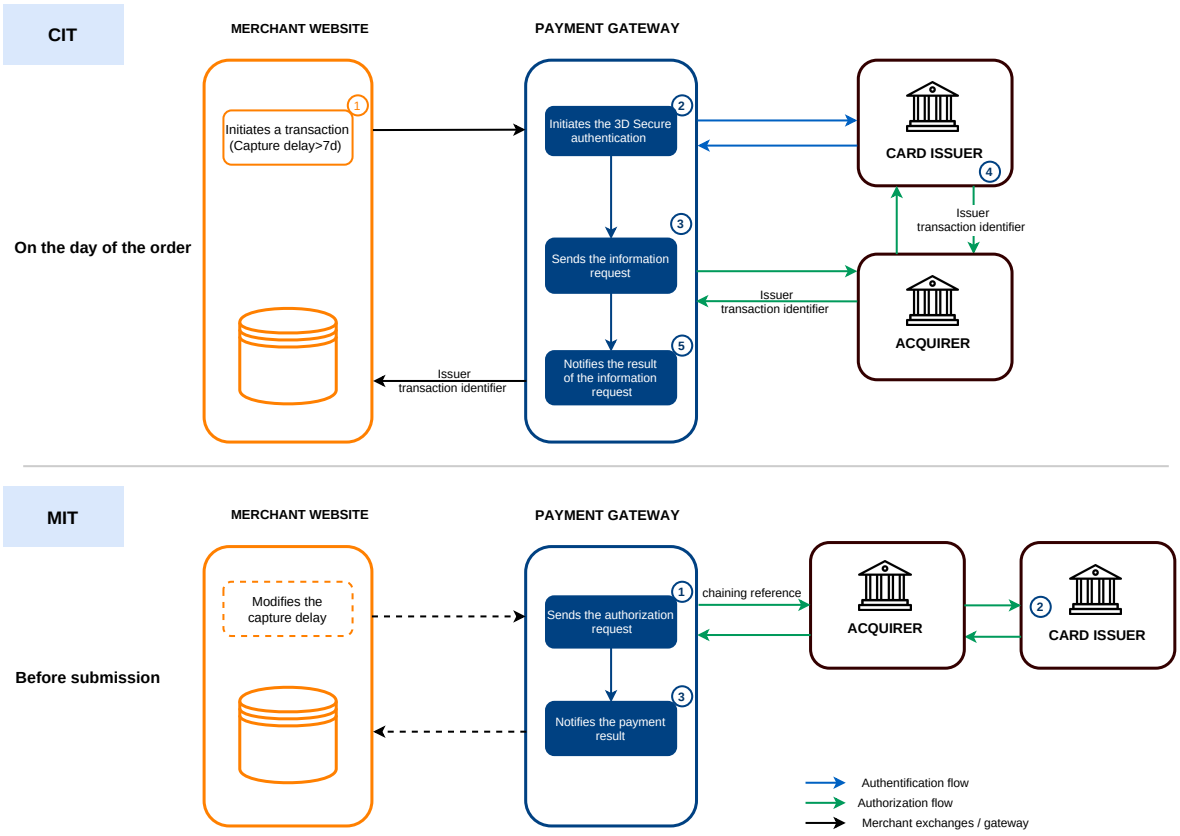
   The regulation imposes cardholder authentication for all CIT transactions.
3. After the authentication (challenge or frictionless), the gateway proceeds with the authorization request by providing the cardholder's authentication details.
4. The issuer generates a unique transaction identifier and transmits it in the response.
5. The payment gateway notifies the merchant website about the payment result.

Before submission:

1. If the transaction is submitted before the initial capture delay expires, the merchant modifies the capture date to D.
2. If no action is taken for the transaction, the transaction is captured by the bank on the initially requested date.

3.2.2. Capture delay longer than the authorization validity period

Simplified diagram



On the day of the order:

1. The merchant site submits a payment request.
2. The payment gateway initiates the cardholder's authentication process with the issuer.

   The regulation imposes cardholder authentication for all CIT transactions.
3. Once the authentication (challenge or frictionless) is completed, the gateway proceeds to an information request by providing the cardholder's authentication details.
4. The issuer generates a unique transaction identifier and transmits it in the response.
5. The payment gateway notifies the merchant website about the information request result.

Before submission:

1. If no action for the transaction, the authorization request is made on D-1 before the requested capture date.

   If the transaction is submitted before the initial capture delay expires, the merchant modifies the capture date to D.

   The payment gateway performs an authorization request, providing the initial transaction identifier (ITC) as a chaining reference.
2. The issuer recognizes the transaction as an MIT that is part of a series of payments where the cardholder has previously authenticated themselves.

   The transaction is not rejected for lack of authentication (soft decline).
3. If the merchant has enabled the notification rule "Instant Payment Notification URL on batch authorization", the payment gateway notifies the merchant site of the payment result.

3.3. Payment in installments

A payment is considered to be an “installment payment” if the amount for the purchase is debited to the buyer’s account in several installments.

The first installment works as an immediate full payment, the subsequent installments are similar to deferred full payments.

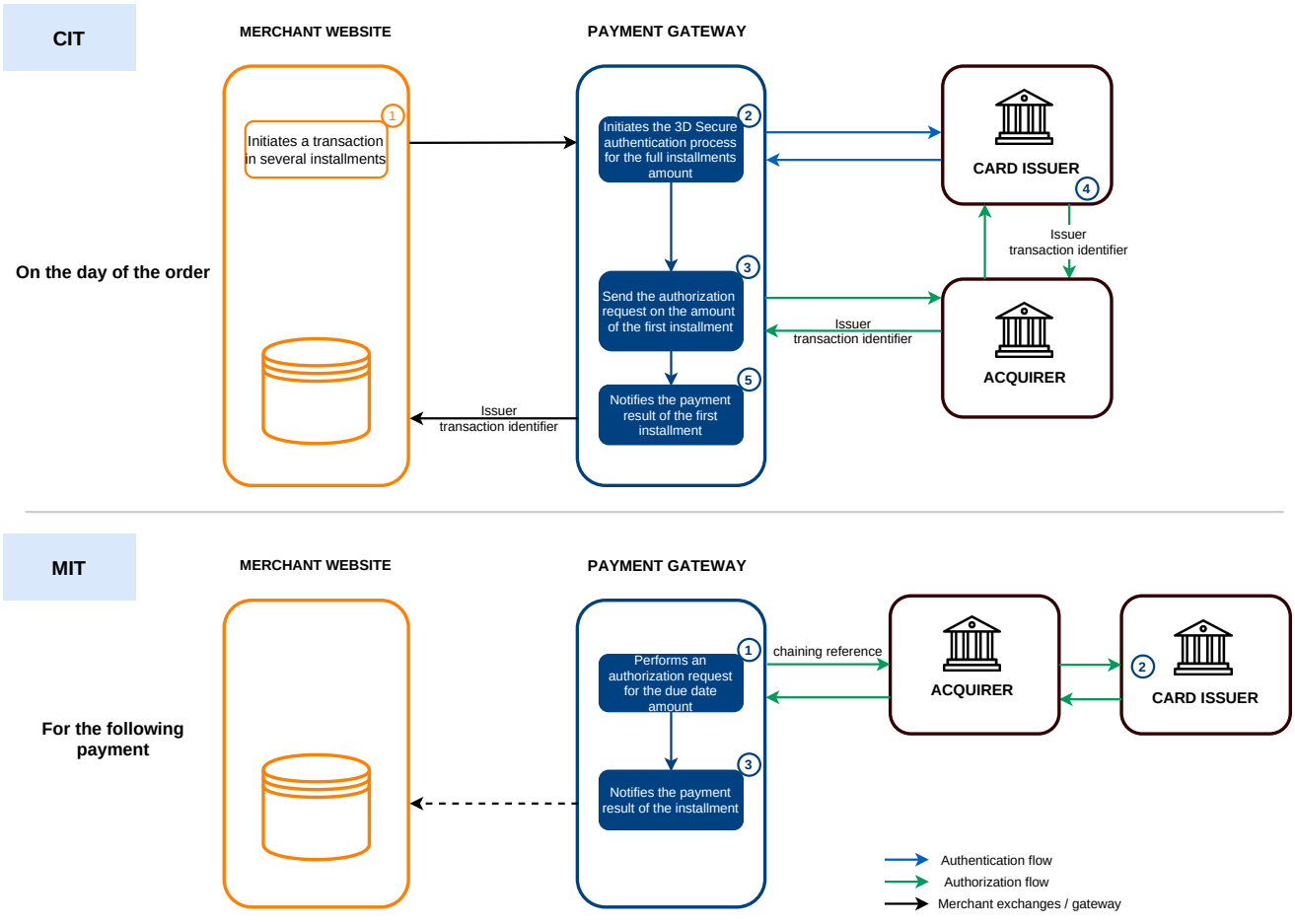
Only the first installment can be guaranteed to the merchant on the condition that the capture date for the installment is set before the authorization expiry date depending on the payment method (see: Authorization request validity period on page 22) .

As part of the implementation of PSD2, the cardholder will be required to undergo strong authentication when making the first installment payment.

If the authorization (or information) request is accepted on the day of the order, a transaction is created for each Installment payment due date.

Otherwise, only one rejected transaction is created. The **History** tab in the Expert Back Office indicates the number of initially planned installments.

Simplified diagram



On the day of the order:

1. The merchant site submits a payment request in several installments.
2. The payment gateway initiates the cardholder’s authentication process with the issuer.

   Authentication is requested for the total amount of installment. Regulations require strong authentication.

3. After the strong authentication, the gateway proceeds with the authorization request on the amount of the first installment by providing the cardholder's authentication details.
4. The issuer generates a unique transaction identifier and resend it in the response.
5. The payment gateway notifies the merchant website about the payment result.

For subsequent payments:

1. The payment gateway performs an authorization request for the installment amount, providing the initial transaction identifier (ITC) as a chaining reference.
2. The issuer recognizes the transaction as an MIT that is part of a series of payments previously authenticated by the cardholder and proceeds with the authorization request.

   The transaction is not rejected for lack of authentication (soft decline).
3. If the merchant has enabled the "Instant Payment Notification URL on batch authorization", the payment gateway notifies the merchant site of the payment result.

   In this use case, the way the chaining reference is handled is transparent to the merchant.

A verification is made to check the payment method validity throughout the payment schedule.

If the card is invalid, a warning message is shown to the buyer, who must use another payment method or abandon the payment.

However, if the card is renewed or terminated before the end of the payment schedule, the payments will be refused by the issuing bank (auto 54 return code: Payment method expired.

In this case, you will receive an e-mail notification via the "Refusal e-mail for deferred payment" notification rule.

List of payment methods compatible with payment in installments:

Network code	Payment method	Card types(vads_payment_cards)	Supports payment in installments
ACCORD	Illicado gift Card	ILLICADO	✗
ACCORD_SANDBOX	Illicado gift cards - Sandbox mode	ILLICADO_SB	✗
ALIPAY_PLUS	Akulaku PayLater ID	AKULAKU_ID	✗
ALIPAY_PLUS	Akulaku PayLater PH	AKULAKU_PH	✗
ALIPAY_PLUS	Alipay CN (China)	ALIPAY_CN	✗
ALIPAY_PLUS	Alipay HK (Hong Kong)	ALIPAY_HK	✗
ALIPAY_PLUS	BillEase	BILLEASE	✗
ALIPAY_PLUS	Boost	BOOST	✗
ALIPAY_PLUS	BPI	BPI	✗
ALIPAY_PLUS	Dana	DANA	✗
ALIPAY_PLUS	GCash	GCASH	✗
ALIPAY_PLUS	Kakao Pay	KAKAOPAY	✗
ALIPAY_PLUS	Krevido	KREDIVO_ID	✗
ALIPAY_PLUS	Maya	MAYA	✗

Network code	Payment method	Card types(vads_payment_cards)	Supports payment in installments
ALIPAY_PLUS	MPay	MPAY	✗
ALIPAY_PLUS	Rabbit LINE Pay	RABBIT_LINE_PAY	✗
ALIPAY_PLUS	Touch 'n Go eWallet	TNG	✗
ALIPAY_PLUS	Toss Pay	TOSS	✗
ALIPAY_PLUS	TrueMoney Wallet	TRUEMONEY	✗
ALMA	Alma in 2 installments	ALMA_2X	✗
ALMA	Alma in 3 installments	ALMA_3X	✗
ALMA	Alma in 4 installments	ALMA_4X	✗
ALMA	Alma in 10 installments	ALMA_10X	✗
ALMA	Alma in 12 installments	ALMA_12X	✗
AMEXGLOBAL	American Express	AMEX	✓
APPLE PAY	Apple Pay wallet payment	APPLE_PAY	✗
AURORE	Cpay card	AURORE-MULTI	✗
BIZUM	Bizum	BIZUM	✗
CB	CB	CB	✓
CB	e-Carte Bleue virtual card	E-CARTEBLEUE	✓
CB	Maestro	MAESTRO	✓
CB	Mastercard	MASTERCARD	✓
CB	Visa	VISA	✓
CB	Visa Electron	VISA_ELECTRON	✓
CB	VPay	VPAY	✓
CB	Apetiz Meal Voucher card	APETIZ	✓
CB	Chèque Déjeuner Meal Voucher card	CHQ_DEJ	✓
CB	1st generation Mastercard electronic meal voucher	EDENRED	✓
CB	Sodexo Meal Voucher card	SODEXO	✓
COFIDIS	Cofidis in 3 installments (France)	COFIDIS_3X_FR	✗
COFIDIS	Cofidis in 4 installments (France)	COFIDIS_4X_FR	✗

Network code	Payment method	Card types(vads_payment_cards)	Supports payment in installments
COFIDIS	Cofidis in 5-12 installments (France)	COFIDIS_LOAN_FR	✗
COFIDIS	Cofidis Pay Later (France)	COFIDIS_DFPAY_FR	✗
COFIDIS	Cofidis in 3 installments (Belgium)	COFIDIS_3X_BE	✗
COFIDIS	Cofidis in 5-12 installments (Belgium)	COFIDIS_LOAN_BE	✗
COFIDIS	Cofidis in 4 installments (Spain)	COFIDIS_4X_ES	✗
COFIDIS	Cofidis in 5-12 installments (Spain)	COFIDIS_LOAN_ES	✗
COFIDIS	Cofidis in 5-12 installments (Italy)	COFIDIS_LOAN_IT	✗
CONECS	Bimpli Meal Voucher card (ex Apetiz)	APETIZ	✗
CONECS	Chèque Déjeuner Meal Voucher card	CHQ_DEJ	✗
CONECS	Conecs Meal Voucher card	CONECS	✗
CONECS	Sodexo Meal Voucher card	SODEXO	✗
CVCONNECT	Chèque-Vacances Connect	CVCO	✗
EDENRED	Edenred EcoChèque Ticket	EDENRED_EC	✗
EDENRED	Edenred Compliments Ticket	EDENRED_TC	✗
EDENRED	Edenred meal voucher	EDENRED_TR	✗
EDENRED	Sport & Culture Edenred Ticket	EDENRED_SC	✗
FRANFINANCE	Franfinance payment in 3X	FRANFINANCE_3X	✗
FRANFINANCE	Franfinance payment in 4X	FRANFINANCE_4X	✗
FRANFINANCE_SB	Franfinance payment in 3X - Sandbox mode	FRANFINANCE_3X	✗
FRANFINANCE_SB	Franfinance payment in 4X - Sandbox mode	FRANFINANCE_4X	✗
FULLCB	Payment in 3 installments with no fees with BNPP PF	FULLCB3X	✗
FULLCB	Payment in 4 installments with no fees with BNPP PF	FULLCB4X	✗
GATECONEX	Bancontact	BANCONTACT	✗
GATECONEX	Diners Club	DINERS	✓

Network code	Payment method	Card types(vads_payment_cards)	Supports payment in installments
GATECONEX	Discover	DISCOVER	✗
GATECONEX	e-Carte Bleue virtual card	E-CARTEBLEUE	✓
GATECONEX	Maestro	MAESTRO	✗
GATECONEX	Mastercard	MASTERCARD	✓
GATECONEX	Visa	VISA	✓
GATECONEX	Visa Electron	VISA_ELECTRON	✗
GATECONEX	VPay	VPAY	✗
GICC_DINERS	Diners Club	DINERS	✓
GICC_DINERS	Discover	DISCOVER	✓
GICC_MAESTRO	Bancontact	BANCONTACT	✗
GICC_MAESTRO	Maestro	MAESTRO	✗
GICC_MASTERCARD	Mastercard	MASTERCARD	✓
GICC_VISA	Visa	VISA	✓
GICC_VISA	Visa Electron	VISA_ELECTRON	✗
GICC_VISA	VPay	VPAY	✗
GOOGLEPAY	Google Pay wallet payment	GOOGLEPAY	✗
JCB	JCB	JCB	✓
LYRA_COLLECT_PPRO	Alipay	ALIPAY	✗
LYRA_COLLECT_PPRO	Bancontact Mistercash	BANCONTACT	✗
LYRA_COLLECT_PPRO	iDeal Internet Banking	IDEAL	✗
LYRA_COLLECT_PPRO	Multibanco	MULTIBANCO	✗
LYRA_COLLECT_PPRO	MyBank	MYBANK	✗
LYRA_COLLECT_PPRO	Przelewy24	PRZELEWY24	✗
LYRA_COLLECT_PPRO	UnionPay	UNION_PAY	✗
LYRA_COLLECT_PPRO	WeChat	WECHAT	✗
MULTIBANCO	MB Reference	MULTIBANCO	✗
MULTIBANCO	MB Way	MB_WAY	✗
ONEY_API	Oney 3x 4x payment	ONEY_3X_4X	✗
ONEY_API	Payment 10x 12x Oney	ONEY_10X_12X	✗

Network code	Payment method	Card types(vads_payment_cards)	Supports payment in installments
ONEY_API	Payment Oney Pay Later	ONEY_PAYLATER	✗
ONEY_API	Oney partner brand cards	ONEY_ENSEIGNE	✗
ONEY_API_SANDBOX	Oney 3x 4x payment (Sandbox mode)	ONEY_3X_4X	✗
ONEY_API_SANDBOX	Oney 10x 12x payment (Sandbox mode)	ONEY_10X_12X	✗
ONEY_API_SANDBOX	Payment Oney Pay Later (Sandbox mode)	ONEY_PAYLATER	✗
ONEY_API_SANDBOX	Oney partner brand cards in Sandbox mode	ONEY_ENSEIGNE	✗
ONEY_SANDBOX	FacilyPay Oney - Mode sandbox	ONEY_SANDBOX	✗
ONEY	FacilyPay Oney	ONEY	✗
PAYDIREKT_V2	PayDirekt	PAYDIREKT	✗
PAYCONIQ	Payconiq	PAYCONIQ	✗
PAYPAL	PayPal	PAYPAL	✗
PAYPAL_SB	PayPal - Mode sandbox	PAYPAL_SB	✗
PAYPAL_BNPL	PayPal Pay Later	PAYPAL_BNPL	✗
PAYPAL_BNPL_SB	PayPal Pay Later - Mode sandbox	PAYPAL_BNPL_SB	✗
PLANET_DCC	MASTERCARD	MASTERCARD	✓
PLANET_DCC	VISA	VISA	✓
POSTFINANCEV2	PostFinance	POSTFINANCE	✗
POSTFINANCEV2	PostFinance E-finance	POSTFINANCE_EFIN	✗
PRESTO	Presto by Cetelem online credit solution	PRESTO	✗
REDSYS_REST	American Express	AMEX	✓
REDSYS_REST	Diners Club	DINERS	✓
REDSYS_REST	JCB	JCB	✓
REDSYS_REST	Maestro	MAESTRO	✗
REDSYS_REST	Mastercard	MASTERCARD	✓
REDSYS_REST	Visa	VISA	✓
REDSYS_REST	Visa Electron	VISA_ELECTRON	✗
SEPA	SEPA DIRECT DEBIT	SDD	✗

3.4. Cascading payment (split)

Multicard payment (also known as cascading or split payment) allows the buyer to pay for an order using several payment methods.

There are two use cases:

1. The buyer settles the entire payment with his or her gift or private card.
2. The buyer uses several payment methods to pay for the order.

   For example, the buyer can pay partly with a gift or private card and the remaining amount with a bank card, or use several private cards for the payment.

List of compatible payment methods:

- Sign cards

Payment method	Technical code
Accord brand card	ACCORD_STORE
Alinéa brand card	ALINEA
Auchan brand card	AUCHAN
Boulanger brand card	BOULANGER
Leroy-Merlin brand card	LEROY-MERLIN
Norauto brand card	NORAUTO
PicWic brand card	PICWIC
Villaverde brand card	VILLAVERDE
Accord brand card - Sandbox mode	ACCORD_STORE_SB
Auchan brand card - Sandbox mode	AUCHAN_SB
Boulanger brand card - Sandbox mode	BOULANGER_SB
Leroy-Merlin brand card - Sandbox mode	LEROY-MERLIN_SB
Norauto brand card - Sandbox mode	NORAUTO_SB
PicWic brand card - Sandbox mode	PICWIC_SB
Villaverde brand card - Sandbox mode	VILLAVERDE_SB

- Gifts cards

Payment method	Technical code
Alinéa gift card	ALINEA_CDX
Allobébé gift card	ALLOBEBE_CDX
BizzBee gift card	BIZZBEE_CDX
Brice gift card	BRICE_CDX
Illicado gift Card	ILLICADO
Jouéclub gift card	JOUECLUB_CDX
Allobébé gift card - Sandbox mode	ALLOBEBE_CDX_SB

Payment method	Technical code
BizzBee gift card - Sandbox mode	BIZZBEE_CDX_SB
Brice gift card - Sandbox mode	BRICE_CDX_SB
Illicado gift card - Sandbox mode	ILLICADO_SB
JouéClub gift card - Sandbox mode	JOUECLUB_CDX_SB

- Meal Voucher cards

Payment method	Technical code
Bimpli Meal Voucher card (ex Apetiz)	APETIZ
Chèque Déjeuner Meal Voucher card	CHQ_DEJ
Conecs Meal Voucher card	CONECS
Sodexo Meal Voucher card	SODEXO
EDENRED Meal Voucher card	EDENRED

- e-ticket card - Edenred Belgium

Payment method	Technical code
Ticket Restaurant	EDENRED_TR
Ticket EcoCheque	EDENRED_EC
Ticket Compliments	EDENRED_TC
Ticket Sport & Culture	EDENRED_SC

- Chèque-Vacances Connect

Payment method	Technical code
Chèque-Vacances Connect	CVCO

3.5. Offering payment in another currency

Currency payment with conversion allows merchants to offer price catalogs in different currencies without needing to manage accounting in currencies other than the one specified in their contract.

When the gateway receives the amount in a currency not managed by your MIDs, it makes a conversion to the company's currency based on the daily exchange rate provided by Visa.

The buyer is informed of the indicative rate at the time of payment, but does not really know the final amount of the transaction.

The capture at the bank does not necessarily occur on the day of the authorization and the rate may therefore vary between the date of authorization and the date of capture.

For this reason, the counter-value displayed at the time of payment is provided as an indication.

- The authorization request is sent in the currency of the contract to the card issuer.
- The capture is performed exclusively in the currency of the contract.
- The buyer is debited in the contract currency, with exchange fees applied by their bank, without managing the exchange rate.

At the end of the payment, the merchant receives a notification containing the following fields:

- **vads_amount:** the currency amount,
- **vads_currency:** the currency,
- **vads_effective_amount:** the actual amount in the currency of their contract, calculated using the exchange rate in force at the time of the authorization,
- **vads_effective_currency:** the currency that is used for the capture,
- **vads_change_rate:** the exchange rate applied for converting the amount in the currency of the contract to the buyer's currency.

3.6. The “Anticipated authorizations” service

This service allows you to make an authorization request several days before the desired capture date.

The delay depends on the authorization validity period and the used payment method (see authorization validity period).

If the issuer refuses due to a non-fraud related reason, an automatic process reiterates authorization requests until up to 2 days prior to the capture date at the bank.

The merchant may cancel the transaction or change its amount (only smaller amounts can be entered) and/or the capture date at any moment.

This process applies to:

- Recurring payments
- Deferred payments
- Installments, other than the first one, in case of payment in installments.

In case of refusal for fraud-related reasons, the transaction is permanently rejected.

Here is a list of fraud-related reasons that do not allow authorization reruns.

Network	Authorization return code	Label
CB	03	Invalid acceptor
	04	Keep the card
	05	Do not honor
	07	Keep the card, special conditions
	12	Incorrect Transaction Code
	13	Invalid amount
	14	Invalid cardholder number
	15	Unknown issuer
	31	Unknown acquirer company ID
	33	Expired card
	34	Suspected fraud
	41	Lost card
	43	Stolen card
	54	Expired card
	56	Card absent from the file
	57	Transaction not allowed for this cardholder
	59	Transaction not allowed for this cardholder
	63	Security rules unfulfilled
	76	The cardholder is already blocked, the previous record has been saved
	80	Contactless payment is not accepted by the issuer

Network	Authorization return code	Label
	81	Unsecured payment is not accepted by the issuer
	82	Revocation of recurring payment for the card of a specific Merchant or for the MCC and the card
	83	Revocation of all recurring payments for the card

Please contact sales administration if you wish to enable anticipated authorizations.

3.7. Authorization request validity period

Network code	Payment method	Card types(vads_payment_cards)	Authorization validity period (in days)
ACCORD	Illicado gift Card	ILLICADO	0
ACCORD_SANDBOX	Illicado gift cards - Sandbox mode	ILLICADO_SB	0
ALIPAY_PLUS	Akulaku PayLater ID	AKULAKU_ID	0
ALIPAY_PLUS	Akulaku PayLater PH	AKULAKU_PH	0
ALIPAY_PLUS	Alipay CN (China)	ALIPAY_CN	0
ALIPAY_PLUS	Alipay HK (Hong Kong)	ALIPAY_HK	0
ALIPAY_PLUS	BillEase	BILLEASE	0
ALIPAY_PLUS	Boost	BOOST	0
ALIPAY_PLUS	BPI	BPI	0
ALIPAY_PLUS	Dana	DANA	0
ALIPAY_PLUS	GCash	GCASH	0
ALIPAY_PLUS	Kakao Pay	KAKAOPAY	0
ALIPAY_PLUS	Krevido	KREDIVO_ID	0
ALIPAY_PLUS	Maya	MAYA	0
ALIPAY_PLUS	MPay	MPAY	0
ALIPAY_PLUS	Rabbit LINE Pay	RABBIT_LINE_PAY	0
ALIPAY_PLUS	Touch 'n Go eWallet	TNG	0
ALIPAY_PLUS	Toss Pay	TOSS	0
ALIPAY_PLUS	TrueMoney Wallet	TRUEMONEY	0
ALMA	Alma in 2 installments	ALMA_2X	0
ALMA	Alma in 3 installments	ALMA_3X	0
ALMA	Alma in 4 installments	ALMA_4X	0
ALMA	Alma in 10 installments	ALMA_10X	0
ALMA	Alma in 12 installments	ALMA_12X	0
AMEXGLOBAL	American Express	AMEX	7
APPLE PAY	Apple Pay wallet payment	APPLE_PAY	Selon la carte de paiement
AURORE	Cpay card	AURORE-MULTI	29
BIZUM	Bizum	BIZUM	0
CB	CB	CB	7
CB	e-Carte Bleue virtual card	E-CARTEBLEUE	7
CB	Maestro	MAESTRO	30

Network code	Payment method	Card types(vads_payment_cards)	Authorization validity period (in days)
CB	Mastercard	MASTERCARD	7
CB	Visa	VISA	7
CB	Visa Electron	VISA_ELECTRON	7
CB	VPay	VPAY	7
CB	Bimpli Meal Voucher card (ex Apetiz)	APETIZ	7
CB	Chèque Déjeuner Meal Voucher card	CHQ_DEJ	7
CB	1st generation Mastercard electronic meal voucher	EDENRED	7
CB	Sodexo Meal Voucher card	SODEXO	7
COFIDIS	Cofidis in 3 installments (France)	COFIDIS_3X_FR	6
COFIDIS	Cofidis in 4 installments (France)	COFIDIS_4X_FR	6
COFIDIS	Cofidis in 5-12 installments (France)	COFIDIS_LOAN_FR	6
COFIDIS	Cofidis Pay Later (France)	COFIDIS_DFPAY_FR	15/30/45
COFIDIS	Cofidis in 3 installments (Belgium)	COFIDIS_3X_BE	6
COFIDIS	Cofidis in 5-12 installments (Belgium)	COFIDIS_LOAN_BE	6
COFIDIS	Cofidis in 4 installments (Spain)	COFIDIS_4X_ES	6
COFIDIS	Cofidis in 5-12 installments (Spain)	COFIDIS_LOAN_ES	6
COFIDIS	Cofidis in 5-12 installments (Italy)	COFIDIS_LOAN_IT	6
CONECS	Bimpli Meal Voucher card (ex Apetiz)	APETIZ	30
CONECS	Chèque Déjeuner Meal Voucher card	CHQ_DEJ	30
CONECS	Conecs Meal Voucher card	CONECS	30
CONECS	Sodexo Meal Voucher card	SODEXO	30
CVCONNECT	Chèque-Vacances Connect	CVCO	6
DFS	Diners Club	DINERS	28
DFS	Discover	DISCOVER	28
EDENRED	Edenred EcoChèque Ticket	EDENRED_EC	0

Network code	Payment method	Card types(vads_payment_cards)	Authorization validity period (in days)
EDENRED	Edenred Compliments Ticket	EDENRED_TC	0
EDENRED	Edenred meal voucher	EDENRED_TR	0
EDENRED	Sport & Culture Edenred Ticket	EDENRED_SC	0
FRANFINANCE	Franfinance payment in 3X	FRANFINANCE_3X	0
FRANFINANCE	Franfinance payment in 4X	FRANFINANCE_4X	0
FRANFINANCE_SB	Franfinance payment in 3X - Sandbox mode	FRANFINANCE_3X	0
FRANFINANCE_SB	Franfinance payment in 4X - Sandbox mode	FRANFINANCE_4X	0
FULLCB	Payment in 3 installments with no fees with BNPP PF	FULLCB3X	7
FULLCB	Payment in 4 installments with no fees with BNPP PF	FULLCB4X	7
GATECONEX	Bancontact	BANCONTACT	30
GATECONEX	Diners Club	DINERS	3
GATECONEX	Discover	DISCOVER	5
GATECONEX	e-Carte Bleue virtual card	E-CARTEBLEUE	7
GATECONEX	Maestro	MAESTRO	30
GATECONEX	Mastercard	MASTERCARD	7
GATECONEX	Visa	VISA	7
GATECONEX	Visa Electron	VISA_ELECTRON	7
GATECONEX	VPay	VPAY	7
GICC_DINERS	Diners Club	DINERS	3
GICC_DINERS	Discover	DISCOVER	5
GICC_MAESTRO	Bancontact	BANCONTACT	30
GICC_MAESTRO	Maestro	MAESTRO	30
GICC_MASTERCARD	Mastercard	MASTERCARD	7
GICC_VISA	Visa	VISA	7
GICC_VISA	Visa Electron	VISA_ELECTRON	7
GICC_VISA	VPay	VPAY	7
GOOGLEPAY	Google Pay wallet payment	GOOGLEPAY	Selon la carte de paiement
IP	Virement bancaire	IP_WIRE	90
IP	Virement bancaire instantané	IP_WIRE_INST	0

Network code	Payment method	Card types(vads_payment_cards)	Authorization validity period (in days)
JCB	JCB	JCB	7
LYRA_COLLECT_PPRO	Alipay	ALIPAY	0
LYRA_COLLECT_PPRO	Bancontact	BANCONTACT	0
LYRA_COLLECT_PPRO	iDeal Internet Banking	IDEAL	0
LYRA_COLLECT_PPRO	Multibanco	MULTIBANCO	0
LYRA_COLLECT_PPRO	MyBank	MYBANK	0
LYRA_COLLECT_PPRO	Przelewy24	PRZELEWY24	0
LYRA_COLLECT_PPRO	UnionPay	UNION_PAY	0
LYRA_COLLECT_PPRO	WeChat	WECHAT	0
MULTIBANCO	MB Reference	MULTIBANCO	0
MULTIBANCO	MB Way	MB_WAY	0
ONEY_API	Oney 3x 4x payment	ONEY_3X_4X	0
ONEY_API	Payment 10x 12x Oney	ONEY_10X_12X	0
ONEY_API	Payment Oney Pay Later	ONEY_PAYLATER	0
ONEY_API	Oney partner brand cards	ONEY_ENSEIGNE	0
ONEY_API_SANDBOX	Oney 3x 4x payment (Sandbox mode)	ONEY_3X_4X	0
ONEY_API_SANDBOX	Oney 10x 12x payment (Sandbox mode)	ONEY_10X_12X	0
ONEY_API_SANDBOX	Payment Oney Pay Later (Sandbox mode)	ONEY_PAYLATER	0
ONEY_API_SANDBOX	Oney partner brand cards in Sandbox mode	ONEY_ENSEIGNE	0
ONEY_SANDBOX	FacilyPay Oney - Mode sandbox	ONEY_SANDBOX	255
ONEY	FacilyPay Oney	ONEY	255
PAYDIREKT_V2	PayDirekt	PAYDIREKT	7
PAYCONIQ	Payconiq	PAYCONIQ	0
PAYPAL	PayPal	PAYPAL	3
PAYPAL	PayPal Pay Later	PAYPAL_BNPL	3
PAYPAL_SB	PayPal - Mode sandbox	PAYPAL_SB	3
PAYPAL_SB	PayPal Pay Later - Mode sandbox	PAYPAL_BNPL_SB	3
PLANET_DCC	MASTERCARD	MASTERCARD	0
PLANET_DCC	VISA	VISA	0
POSTFINANCEV2	PostFinance	POSTFINANCE	1

Network code	Payment method	Card types(vads_payment_cards)	Authorization validity period (in days)
POSTFINANCEV2	PostFinance E-finance	POSTFINANCE_EFIN	1
PRESTO	Presto by Cetelem online credit solution	PRESTO	0
REDSYS_REST	American Express	AMEX	28
REDSYS_REST	Diners Club	DINERS	28
REDSYS_REST	JCB	JCB	28
REDSYS_REST	Maestro	MAESTRO	28
REDSYS_REST	Mastercard	MASTERCARD	28
REDSYS_REST	Visa	VISA	28
REDSYS_REST	Visa Electron	VISA_ELECTRON	28
SEPA	SEPA DIRECT DEBIT	SDD	15
NPCIUPI	BHIM UPI	UPI	0

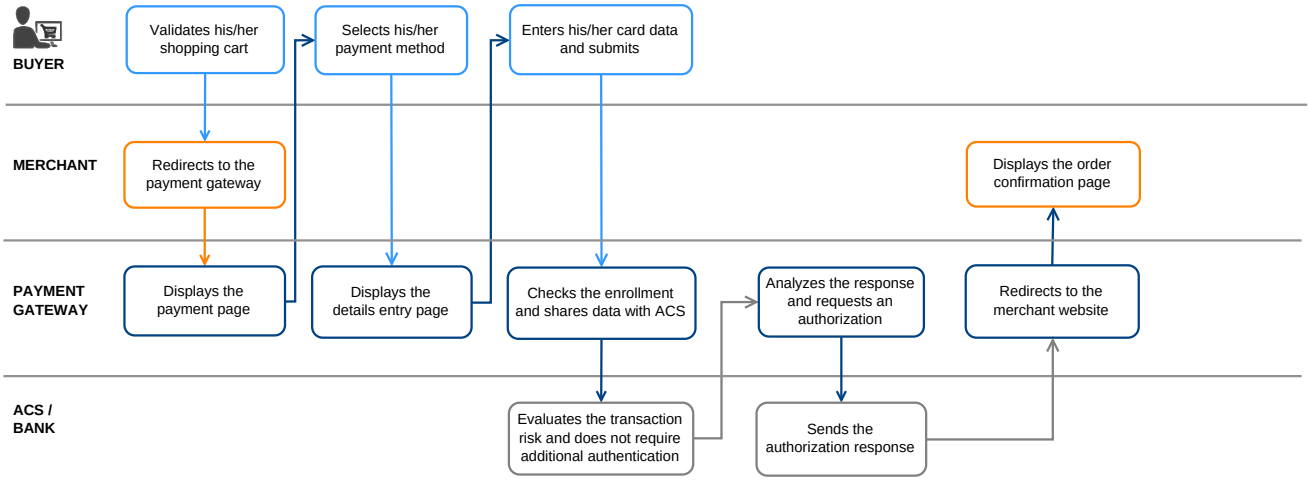
4. THE 3D SECURE AUTHENTICATION

You can find all useful information about 3DS authentication in the 3D Secure guide.

4.1. “Frictionless” flow

In frictionless flow (without interaction with the buyer), based on the received information, the issuer can determine:

- No additional authentication is required.
  The payment gateway proceeds with the payment and issues the authorization request.
- The analyzed information does not allow to move on with the payment.
  The payment gateway notifies the merchant website and the buyer about the payment rejection and redirects the buyer to the merchant website.



4.2. “Challenge” flow

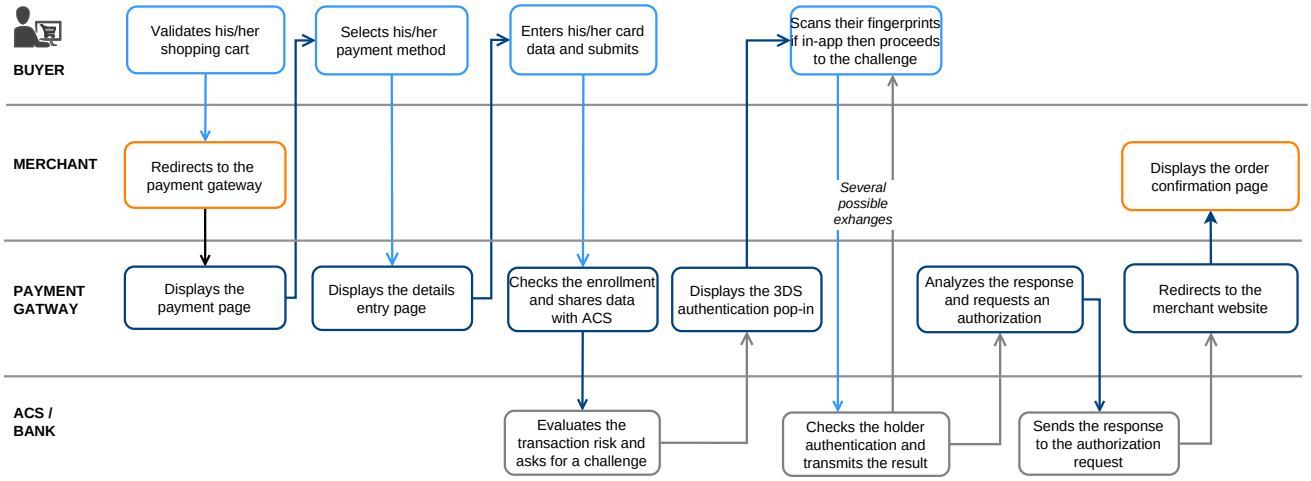
In a challenge flow, based on the received information, the issuer determines that it is necessary for the buyer to provide the following elements:

- A biometric element, such as a device fingerprint,
- A strong two-factor authentication.

For in-app solutions, the device fingerprint is systematically requested before proceeding to the challenge.

Once the challenge has been successfully completed, the payment gateway proceeds with the payment and issues the authorization request.

In case of a technical or authentication error, the payment stops. The payment gateway notifies the merchant about the payment rejection and redirects the buyer to the merchant website



4.3. Increasing the chances of a frictionless payment

The use of these fields is optional. It is always the issuing bank deciding if a two-factor authentication is made or not.

Name/Description	Format/Values
vads_cust_address_number Street number - Billing address	Format: ans..64
vads_cust_address2 2nd line of the address - Billing address	Format: ans..255
vads_cust_address 1st line of the address - Billing address	Format: ans..255
vads_cust_cell_phone Buyer's cell phone number	Format: an..32
vads_cust_city City - Billing address	Format: an..128
vads_cust_email Cardholder's e-mail address	Format: ans..150
vads_cust_national_id National identifier. Allows to identify each citizen of a country in a unique way	Format: ans..255
vads_cust_phone Shipping buyer's phone number	Format: an..32
vads_cust_state State/Region - Billing address	Format: ans..127
vads_cust_zip Zip code - Billing address	Format: an..64
vads_ship_to_city City - Shipping address	Format: an..128
vads_ship_to_email Shipping e-mail address in case of an e-ticket order.	Format: an..128
vads_ship_to_type Transport type	Format: enum 3DS2 value: • **"CARD_HOLDER_ADDRESS"** • **"VERIFIED_ADDRESS"** • **"NOT_VERIFIED_ADDRESS"** • **"SHIP_TO_STORE"** • **"DIGITAL_GOOD"** • **"ETRAVEL_OR_ETICKET"** • **"OTHER"** • **"PICKUP_POINT"** • **"AUTOMATED_PICKUP_POINT"**

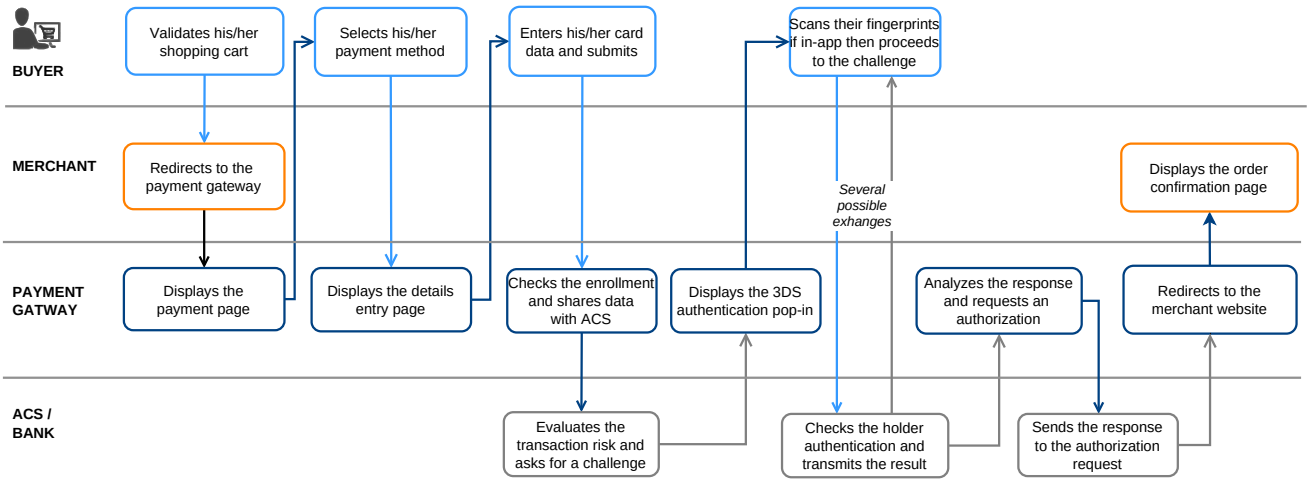
Name/Description	Format/Values
vads_ship_to_state State/Region - Shipping address	Format: ans..127
vads_ship_to_street2 2nd line of the address - Shipping address.	Format: ans..255
vads_ship_to_street 1st line of the address - Shipping address.	Format: ans..255
vads_ship_to_speed Shipping speed	Format: enum 3DS2 value: • "**ELECTRONIC_DELIVERY**" • "**SAME_DAY_SHIPPING**" • "**OVERNIGHT_SHIPPING**" • "**TWO_DAYS_OR_MORE_SHIPPING**"
vads_ship_to_zip Zip code - Shipping address	Format: ans..64

5. UNDERSTANDING THE PAYMENT FLOW

The online payment process appears differently when viewed from the point of view of the buyer or of the merchant.

5.1. Defining the steps of the payment process - As seen by the buyer

Payment process from the buyer's point of view:



1. The buyer validates the shopping cart.
2. The merchant website redirects the buyer to the payment gateway via a secure HTTPS HTML POST form.

   The parameters to include in this form are detailed in the chapter Generating a payment form.
3. When the parameters and their signature have been verified, presents the payment process to the buyer.

   There are two journey, depending on how you fill in the payment form:

   - Journey 1: You have specified only one payment method (e.g. bank card). The gateway directly displays the Payment method data entry page (step 5)
   - Journey 2: You have specified all the available payment methods. The gateway then displays the payment method selection page.

     Example:

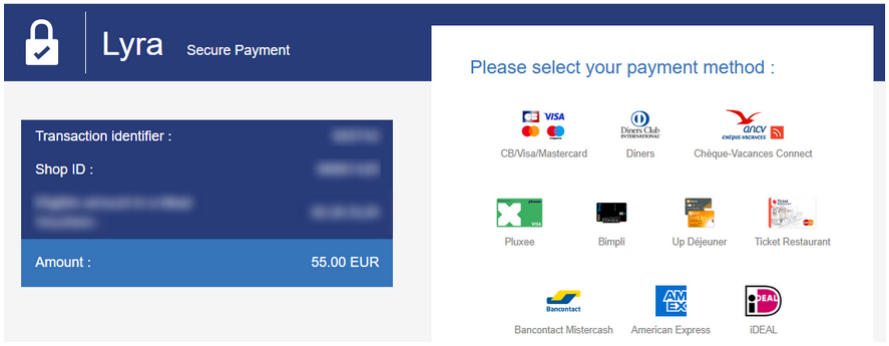


Figure 1: Selecting a payment method

A system called unique logo includes CB, Visa and MasterCard.

It allows the buyer to access the card data entry page directly if the merchant has only a CB contract.

In this case, CB, e-Carte bleue, Visa, Visa Electron, MasterCard and Maestro cards are displayed with the same logo on the payment.

4. The buyer selects his payment method if the gateway displays journey 2.

5. The buyer enters the number and the expiry date of their card.

   If the card has a security (CVV) code, it must be specified.

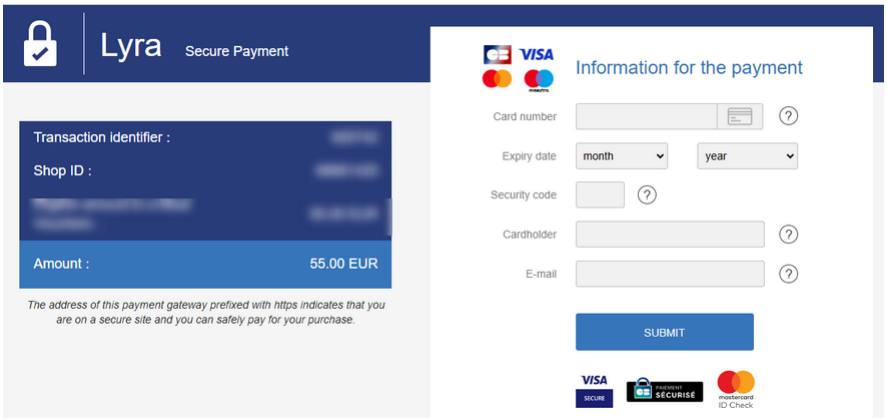


Figure 2: Entering payment method details

6. The buyer click on **Validate**.
7. If the merchant and the buyer's card are enrolled in the 3D Secure program, the payment will be authenticated with 3D Secure.
8. The gateway runs anti-fraud checks and sends an authorization request to the buyer's bank.
9. If the authorization is accepted, the buyer sees the transaction summary.

   A button allowing to return to the shop is presented.

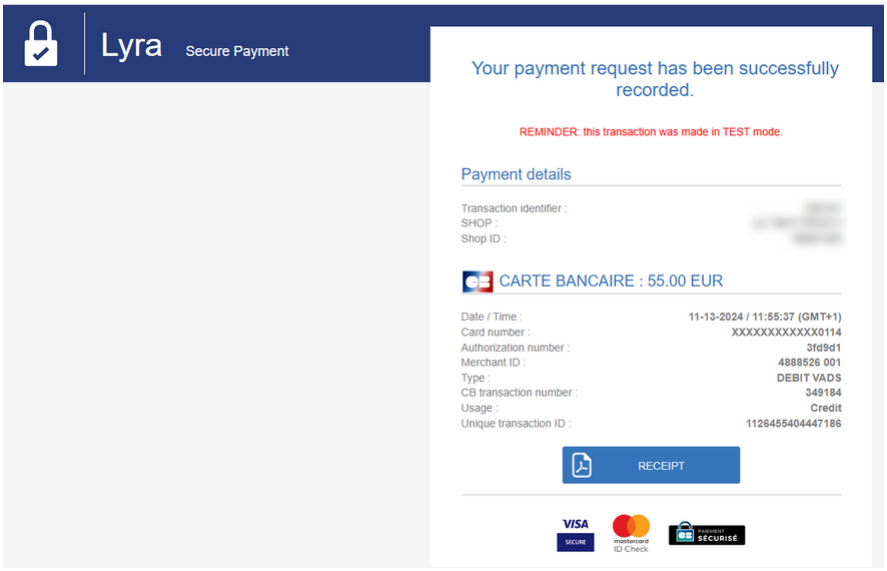


Figure 3: Transaction summary

In the event of failure, the platform informs the buyer and displays a button to cancel and return to the shop.

If the additional attempts are configured in your Expert Back Office, the buyer can make another attempt. The payment process then resumes at the payment method selection.

When all attempts are exhausted, the payment is definitively declined.

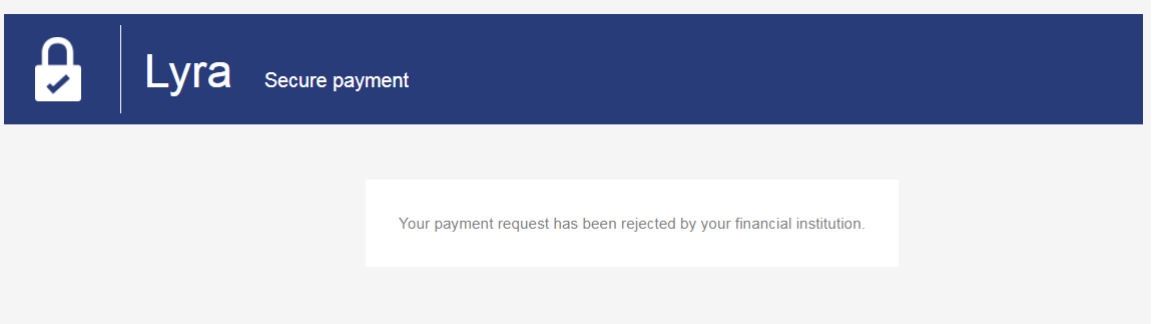


Figure 4: Summary page in case of a failed transaction

5.2. Defining the steps of the payment process - As seen by the merchant

The online payment process on the merchant side follows these steps:

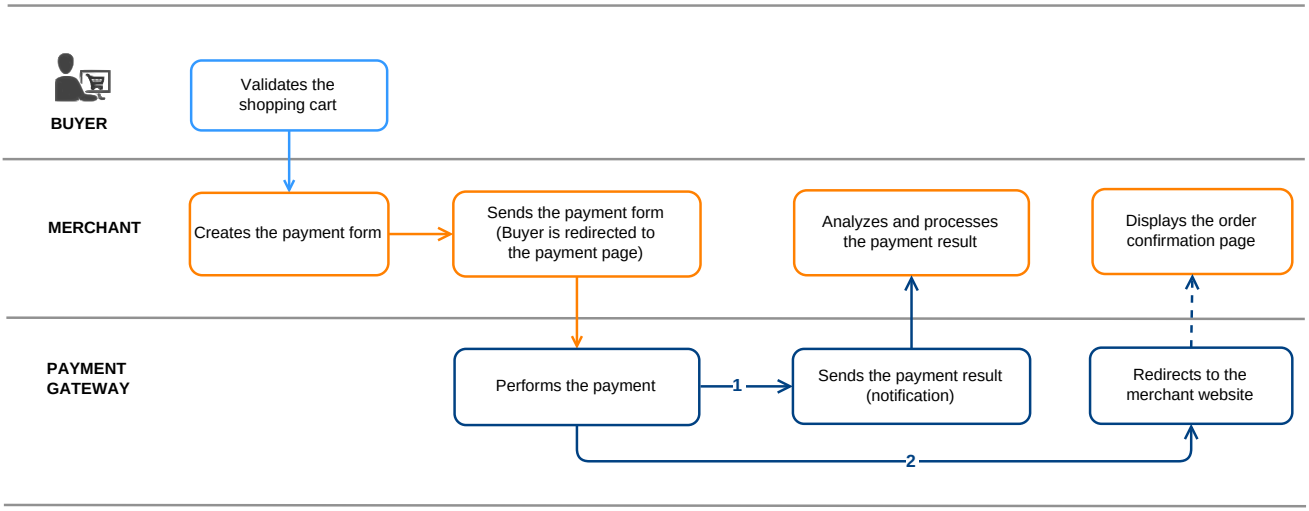


Figure 5: Payment process - as seen by the merchant

1. The buyer validates the shopping cart.
2. The merchant website creates a form using the data from the buyer's cart.
3. The merchant website redirects the buyer to the payment gateway via an HTTPS HTML POST form. The parameters are described in the chapter Generating a payment form.
4. The buyer enters their payment information, and the gateway proceeds to the payment.
5. The gateway automatically informs the merchant website about the result (depending on the configuration), see chapter Setting up notifications.
6. The merchant website analyzes and processes the payment result.
7. The merchant updates their database (order status, stock status, etc.).
8. The buyer is informed about the result and can return to the merchant website to see their order status.

6. OFFERING ADDITIONAL PAYMENT ATTEMPTS

Configure the number of additional attempts in case of a rejected payment:

1. Go to **Settings** > **Shop** and then click the shop where the configuration needs to be modified.
2. Select **Configuration**.
3. Enter the additional number of attempts allowed.

   Example: 2 attempts equal 3 in total.
4. Check **Instant Payment Notification URL on a declined attempt** if you wish to receive a notification (IPN) for each refusal.
5. Click **Save**.

Additional attempts are not allowed for installment payments.

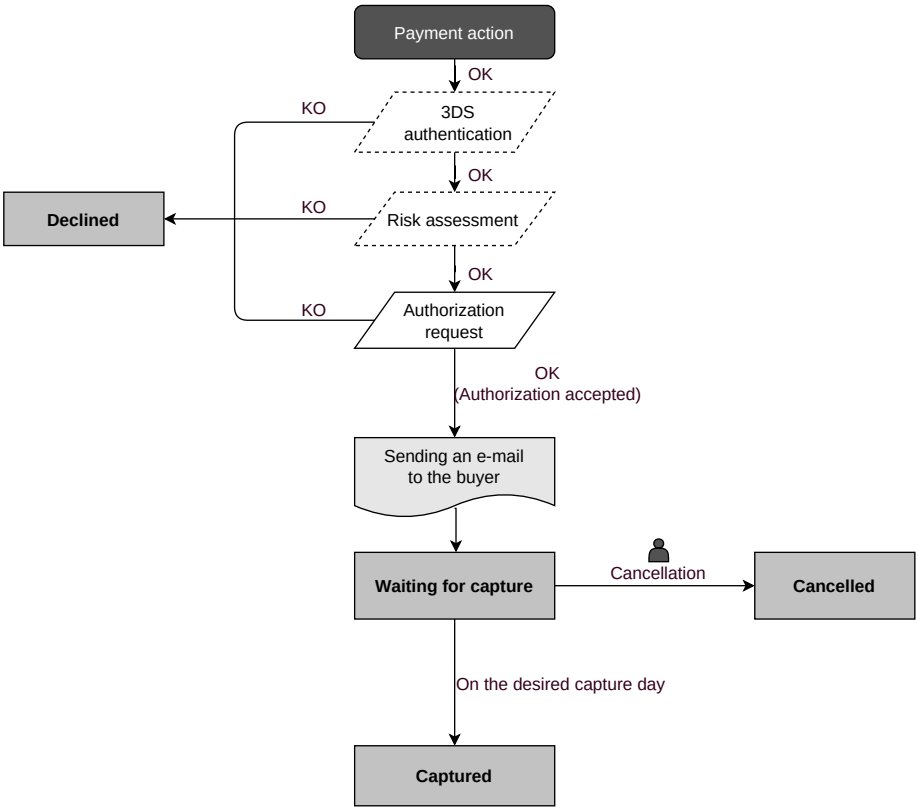
7. TRANSACTION LIFECYCLE

In all the following diagrams, the following caption is used:

Action required from the merchant - manual (Expert Back Office) or automatic (Web Services)

7.1. Immediate payment

7.1.1. Automatic validation



Once the payment request has been made, several verification processes start automatically:

- The 3D Secure authentication.
- Different verification processes performed by the payment gateway (these potentially include local checks, risk rules configured by the merchant) or by an external risk analyzer.
- An authorization request is also made by the buyer's bank on the day of payment, independently of the requested capture date at the bank.

If one of the verification processes fails, the payment request will not be accepted. The buyer is informed of the rejection on the screen. In the Expert Back Office, the transaction appears with the **Refused** status.

Otherwise, the transaction takes the **Waiting for capture** status.

The buyer is informed about the acceptance of the payment request and receives a confirmation e-mail.

The transaction will be automatically submitted for capture on the day requested by the merchant and will take the **Captured** status. The **Captured** status is final.

Once the capture is made, the arrival of the transaction to the merchant account depends on the interbank processing time.

Before the capture date, the buyer can modify it together with the amount (only smaller amounts can be entered in case of partial delivery by the merchant).

If necessary, the buyer can also cancel the transaction: the transaction will then appear with the **Cancelled** status.

7.1.2. Manual validation

Following a payment request, the verification process starts automatically:

- The 3D Secure authentication.
- Different verification processes performed by the payment gateway (these potentially include local checks, risk rules configured by the merchant) or by an external risk analyzer.
- An authorization request is also made by the buyer's bank.

If one of the verification processes fails, the payment request will not be accepted. The buyer is informed of the rejection on the screen. In the Expert Back Office, the transaction appears with the **Refused** status.

Otherwise, the payment is accepted and the transaction appears in the Expert Back Office with the **To be validated** status.

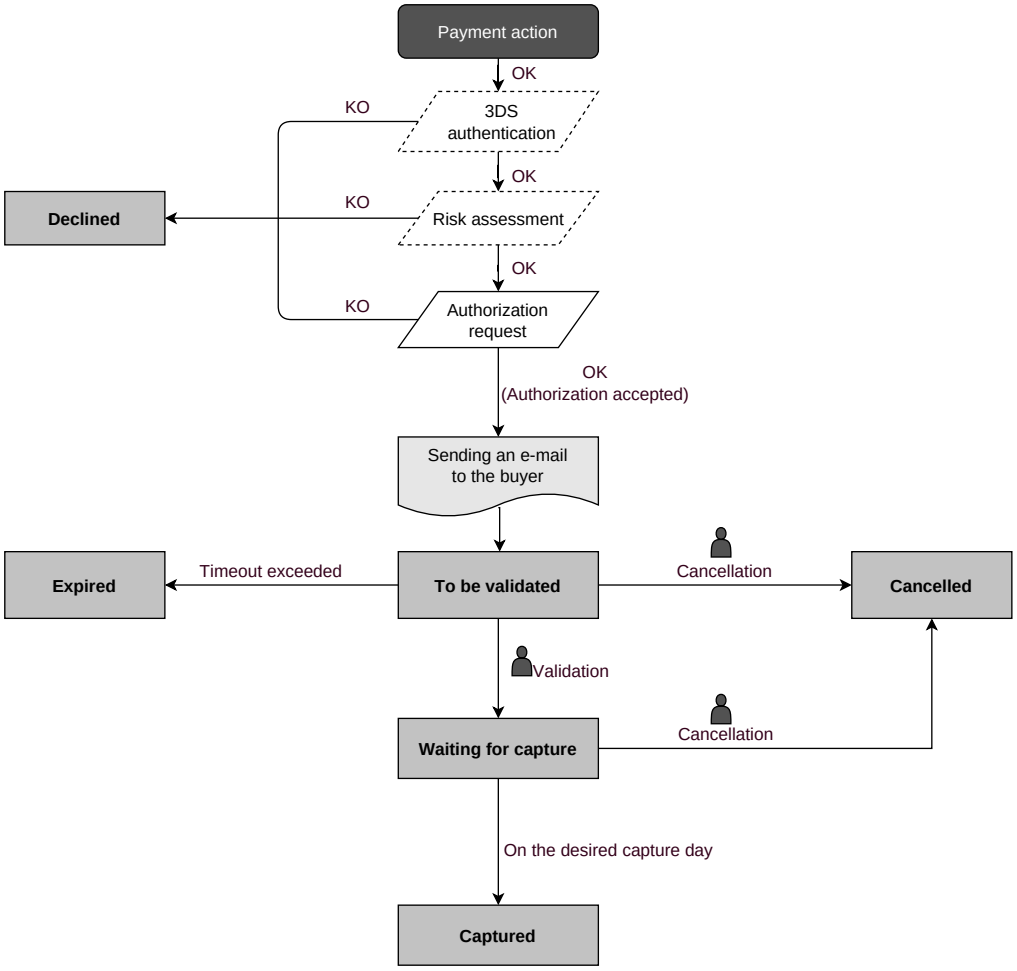
In this case, the merchant must validate the transaction before the expiry date of the authorization request. If the validation is made after this date, the transaction appears as **Expired** and cannot be captured in the bank.

As soon as the transaction is validated, its status changes to **Waiting for capture**.

The transaction will be automatically submitted for capture on the day requested by the merchant and will take the **Captured** status. The **Captured** status is final.

Once the capture is made, the arrival of the transaction to the merchant account depends on the interbank processing time.

The merchant can also cancel the transaction, if necessary. In this case, the transaction takes the **Cancelled** status.



7.2. Deferred payment

7.2.1. Automatic validation

Capture delay shorter than the authorization validity period

(See the diagram “The life cycle of an immediate payment transaction”).

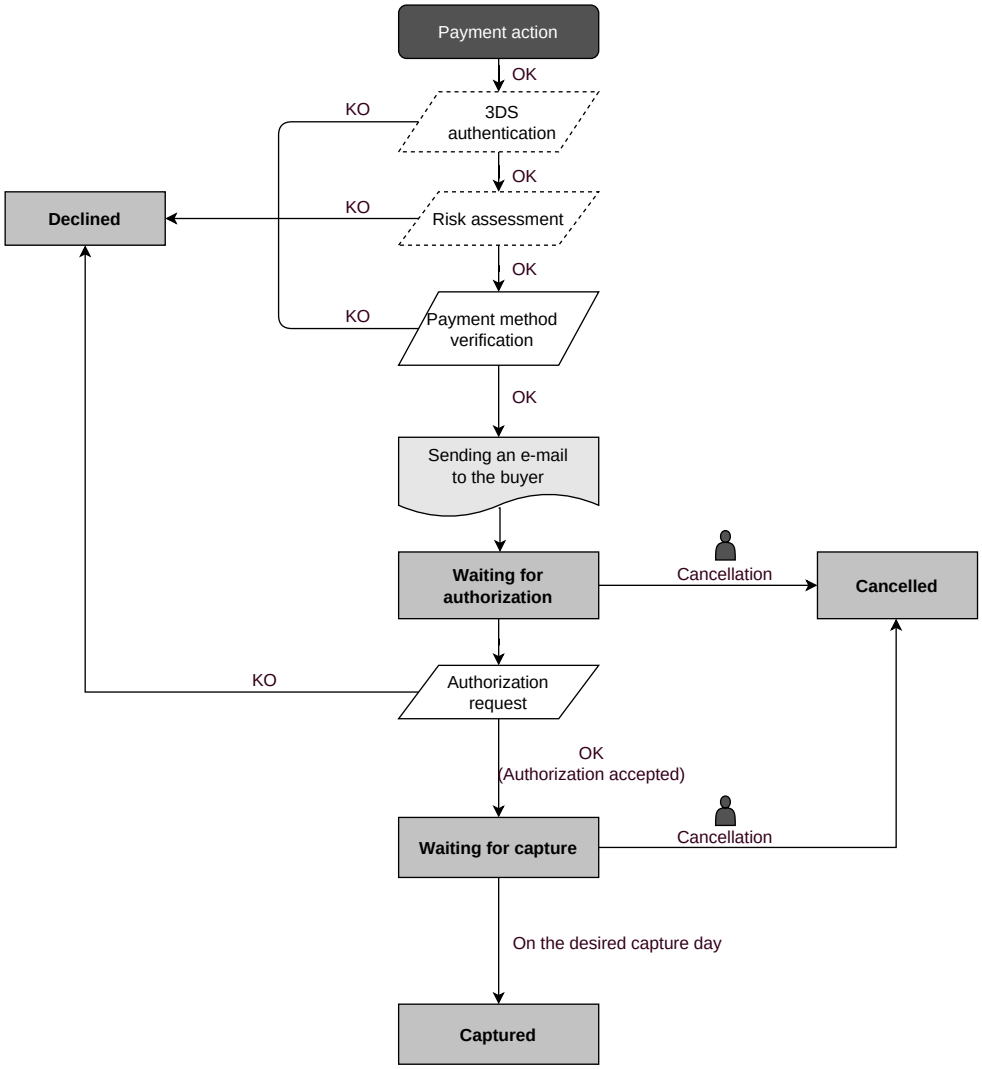
Capture delay longer than the authorization validity period

All the transactions for deferred payments made in automatic validation mode with a successfully completed verification request can be viewed in the Expert Back Office with the **Waiting for authorization** status.

The authorization request is automatically sent:

- By default: the day before the desired capture date,
- With anticipated authorization: depending on the selected payment method, on D-Δ before the desired capture date (see chapter The “Anticipated authorizations” service on page 20).

A deferred payment goes through the steps in the diagram below:



7.2.2. Manual validation

Capture delay shorter than the authorization validity period

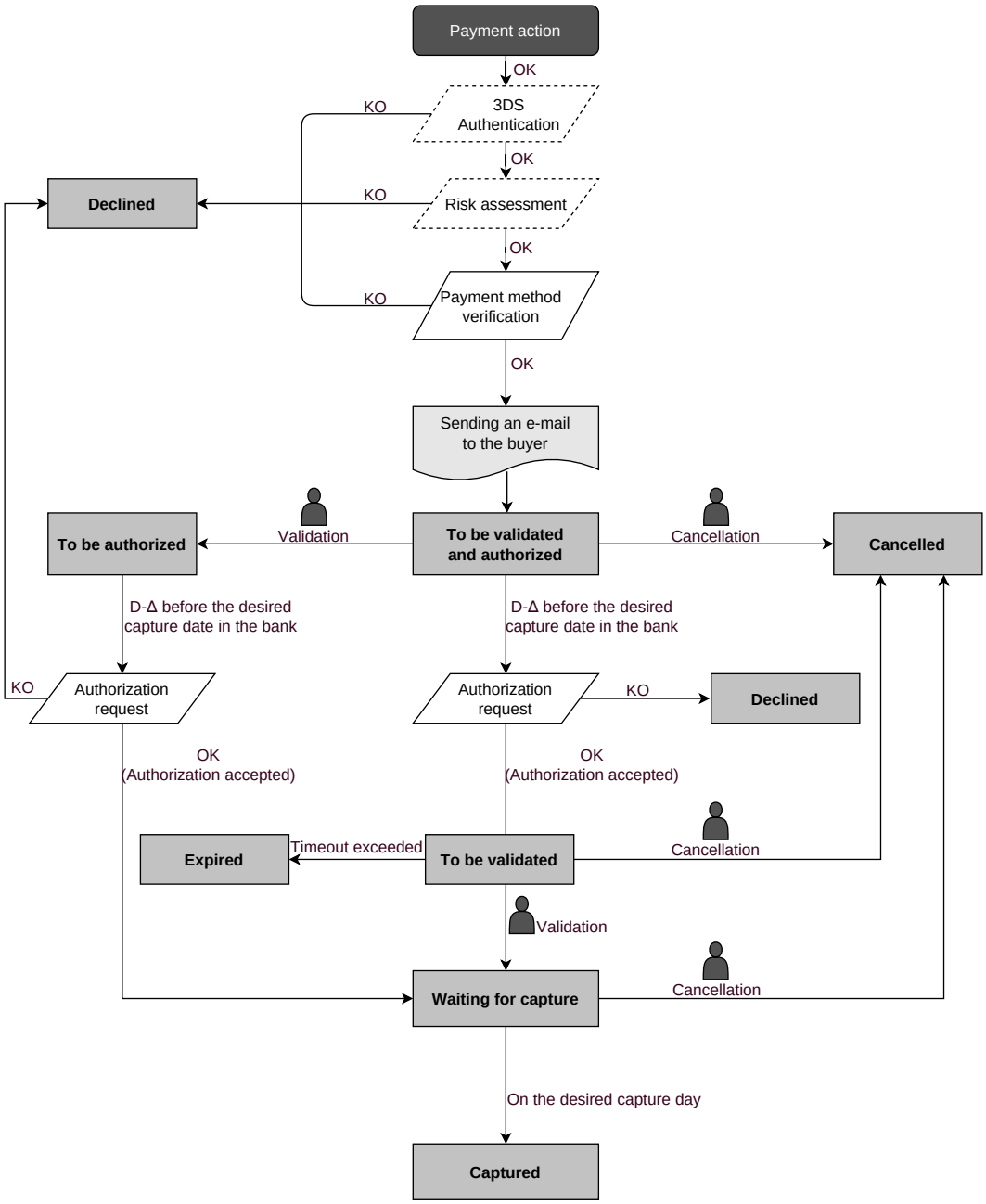
(See the diagram “The life cycle of an immediate payment transaction”).

Capture delay longer than the authorization validity period

All the transactions for deferred payments made in automatic validation mode with a successfully completed authorization request for EUR 1 (or information request about the CB network if the acquirer supports it) can be viewed in the Expert Back Office with the **To be validated and authorized** status.

The authorization request is automatically sent on the requested capture day, on the condition that the merchant has already validated the transaction.

In the meantime, the merchant may cancel the transaction or change its amount (only smaller amounts can be entered) and/or the capture date. These transactions go through the steps in the diagram below:



7.3. Payment in installments

7.3.1. Automatic validation

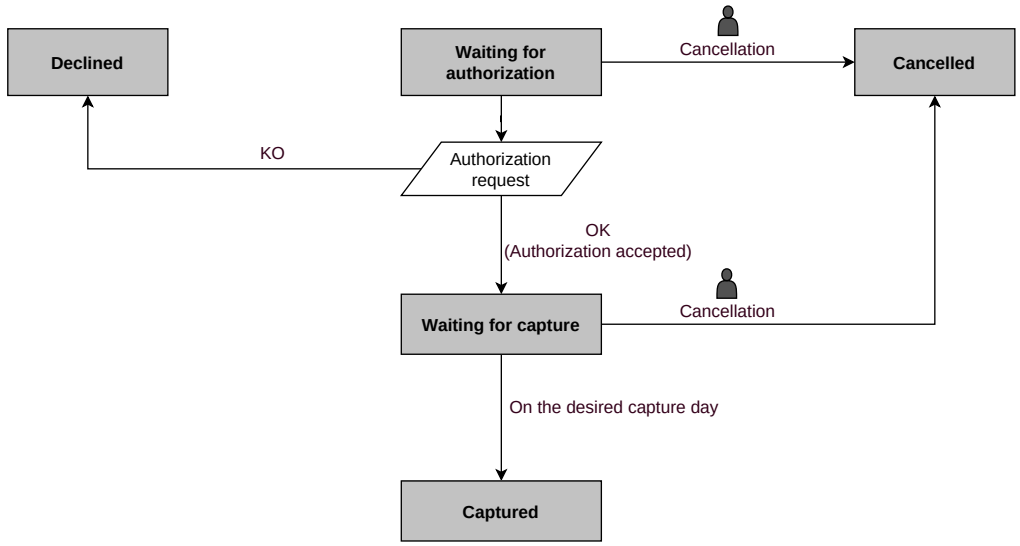
Depending on the capture date, the first installment will have exactly the same features as an immediate payment or a deferred payment.

By default, the following installments will have the **Waiting for authorization** status. The buyer's bank will be able to reject the authorization request. The payment gateway will then inform the merchant by e-mail that the transaction has been declined.

The authorization requests for the upcoming installments are automatically sent as a transaction for a deferred payment, with two possible dates:

- By default: the day before the desired capture date,
- With anticipated authorization: depending on the selected payment method, several days before the desired capture date (see chapter The "Anticipated authorizations" service on page 20).

The following installments go through the steps specified in the diagram below (case of an authorization request that is not resent):



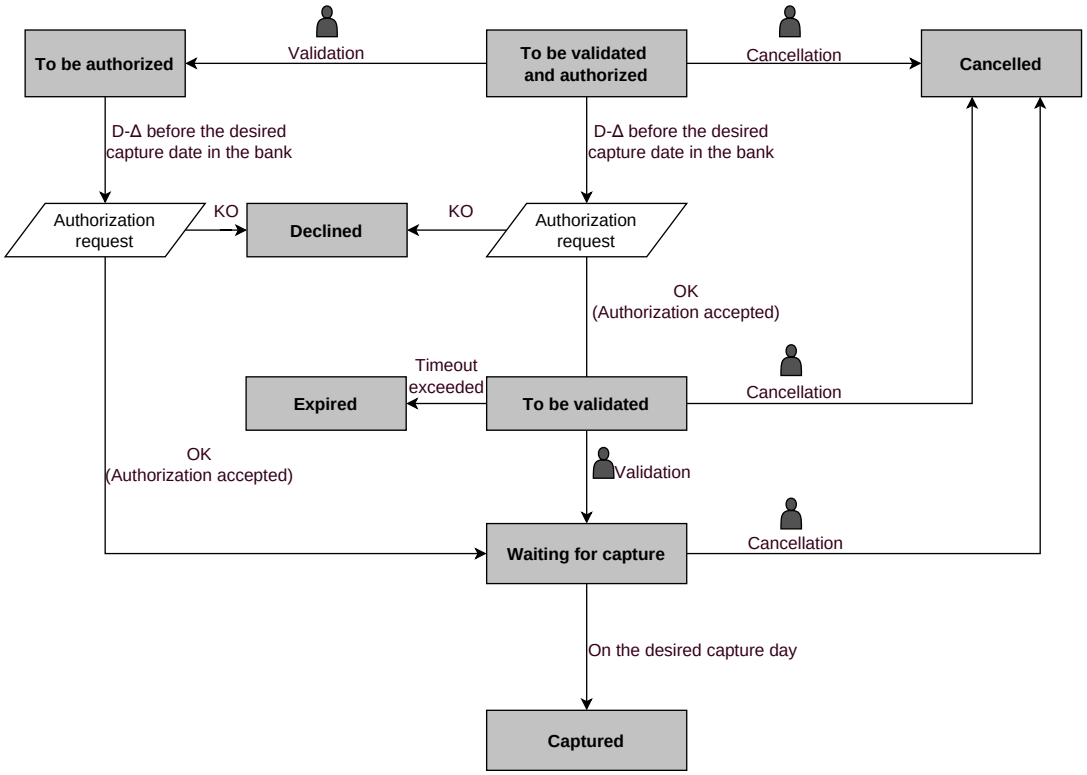
In any case, canceling an installment never implies that the upcoming installments will be canceled.

7.3.2. Manual validation

Depending on the capture date, the first installment will have exactly the same features as an immediate payment or a deferred payment.

By default, the upcoming installments have the **To be validated and authorized** status as long as the first installment has not been validated by the merchant. The successful execution of the installments is not guaranteed to the merchant. The buyer's bank may reject the authorization request.

Validation of the first installment implies that all the other installments will be validated as well. However, canceling an installment does not cancel the upcoming installments.



8. ESTABLISHING INTERACTION WITH THE PAYMENT GATEWAY

The merchant website and the payment gateway interact by exchanging data.

To create a payment, this data is sent in an HTML form via the buyer's browser.

At the end of the payment, the result is transmitted to the merchant website in two ways:

- Automatically by means of notifications called Instant Notification URLs (also called IPN or Instant Payment Notification), see chapter **Setting up notifications**.
- Via the browser, when the buyer clicks the button to return to the merchant website, see chapter **Managing the return to the merchant website**.

To guarantee the security of the exchange, the data is signed with a key known only to the merchant and the payment gateway.

8.1. Setting up the payment page URL

The merchant website interacts with the payment gateway by redirecting the buyer to the following URL:

https://secure.lyra.com/vads-payment/

8.2. Identifying yourself when exchanging with the payment gateway

To be able to interact with the payment gateway, the merchant needs to have:

- **The shop ID:** allows to identify the merchant website during the exchange. Its value is transmitted in the **vads_site_id** field.
- **The key:** allows to compute the alphanumeric signature transmitted in the **signature** field.

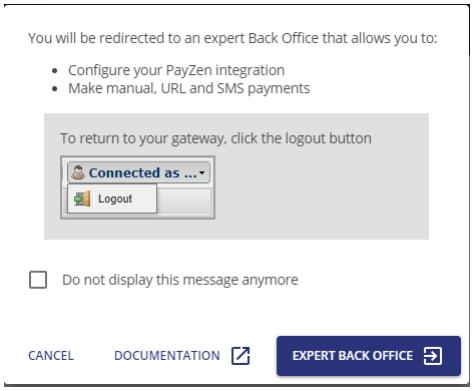
To retrieve these values:

1. Sign in to the **Lyra Collect Back Office**: https://secure.lyra.com/portal/
2. Enter your user name.
3. Enter your password.
4. Click **Login**.

   In case of an entry error of the user name and/or password, the error message "*Invalid username or password*" will appear.

   You can correct your entry or click on the link **Forgotten password or locked account**.
5. Click **Other actions**.

   The following window appears:



6. Click on **Expert Back Office** to access your Expert Back Office.

7. Click **Settings > Shop**.

8. Select **Keys**.

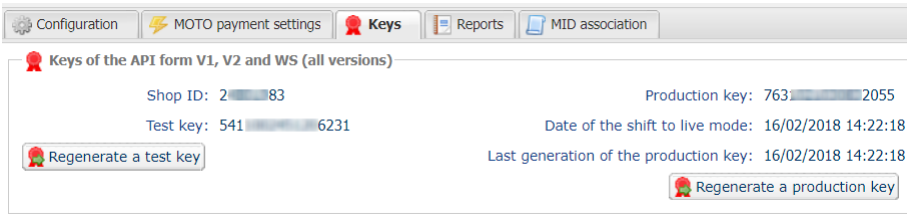


Figure 6: Keys tab

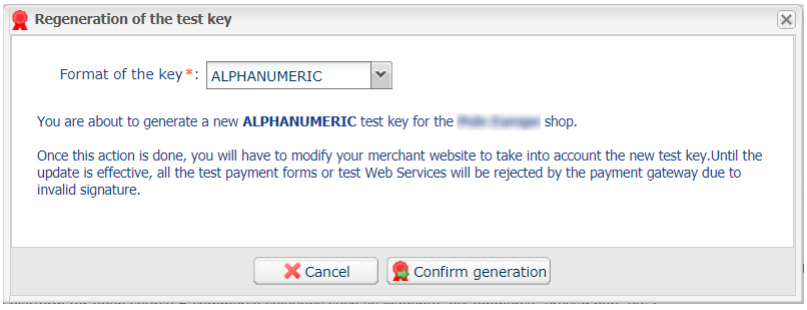
Two types of keys are available:

- The **test key** that allows to generate the form signature in test mode.
- The **production key** that allows you to generate the form signature in production mode.

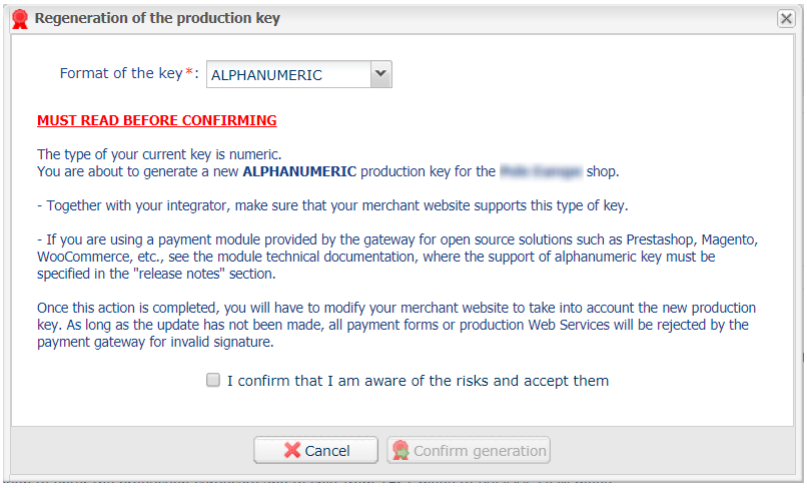
These keys can be numeric or alphanumeric.

For maximum security, it is recommended to use an alphanumeric key.

To change the format of your test key, click the **Regenerate a test key** button and select the format ("ALPHANUMERIC" or "NUMERIC").



To change the format of your production key, click the **Regenerate a production key** button and select the format ("ALPHANUMERIC" or "NUMERIC").



8.3. Choosing between Test and Production modes

The choice between **TEST** or **PRODUCTION** modes can be made using the **vads_ctx_mode** field (See chapter Generating a payment form on page 62).

- The **TEST** mode allows to make test payments.

  It is available at all times, even after the generation of the production key.

  If you create a new merchant website (or have access to the acceptance testing environment), you can make tests without impacting the website that is currently in production.

  TEST transactions can be viewed in the Expert Back Office via **Management** > **TEST transactions**.

- The **PRODUCTION** mode will become available only once the production key has been generated.

  It allows to make real payments.

  PRODUCTION transactions can be viewed in the Expert Back Office via **Management** > **Transactions**.

8.4. Managing interaction with the merchant website

Two types of URLs are used to manage the dialog with the merchant website:

- **Instant Payment Notification**, also called the IPN,
- **Return URL** to the merchant website.

Instant Payment Notification - IPN

The **Notification URL** is the URL of a specific page on the merchant website that is **automatically** called by the payment gateway when certain events take place.

By default, the rules are created to manage the events below:

- end of payment (accepted or rejected),
- payment abandoned or canceled,
- token creation or update,
- recurring payment creation,
- new installment date,
- authorization made in case of a deferred payment,
- update of a transaction status by the acquirer,
- operation made via the Expert Back Office (cancellation, refund, duplication, manual payment, etc.).

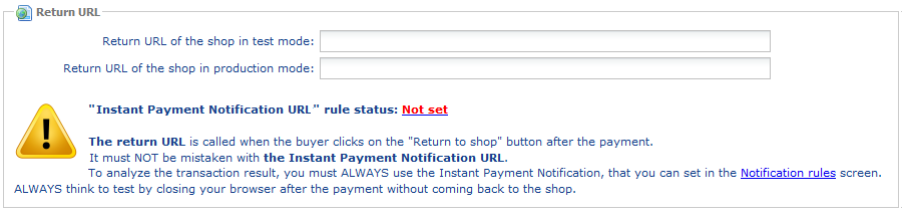
These rules must be enabled and configured according to the needs of the merchant.

With each call, the payment gateway transmits transaction details to the merchant website. It is called instant notification (or **IPN** as in Instant Payment Notification).

To guarantee the security of the exchange, the data is signed with a key known only to the merchant and the payment gateway.

URL of return to the merchant website

In the Expert Back Office, the merchant can configure the "default" return URLs via the menu **Settings** > **Shop** > **Configuration** tab:



The merchant can set up a different return URL for each mode.

By default, the buyer is redirected to the URL regardless of the payment result.

If no URL has been set up, the main URL of the shop will be used for redirection (**URL** parameter defined in the **Details** section of the shop).

The merchant will be able to override this setting in his/her payment form (see chapter **Setting up return URLs**).

The status of the "Instant Payment Notification at the End of Payment" (IPN) rule is displayed in this window. If the URL has not been set up, make sure to specify it (see chapter **Setting up notifications**).

8.5. Managing security

There are several ways to guarantee the security of online payments.

8.5.1. Ensuring interaction integrity

The integrity of exchanged information is preserved by the exchange of alphanumeric signatures between the payment platform and the merchant website.

The payment gateway and the merchant website interact via HTML forms.

A form contains a list of specific fields (see chapter **Generating a payment form**) used to generate a chain.

This chain is then converted to a smaller chain using a hash function (SHA-1, HMAC-SHA-256).

*The merchant will be able to choose the hash algorithm in their Expert Back Office (see chapter **Choosing the hash algorithm**).*

The resulting chain is referred to as the **digest** (*empreinte* in French) of the initial chain.

The digest must be transmitted in the **signature** field (see chapter **Computing the signature**).

Modeling security mechanisms:

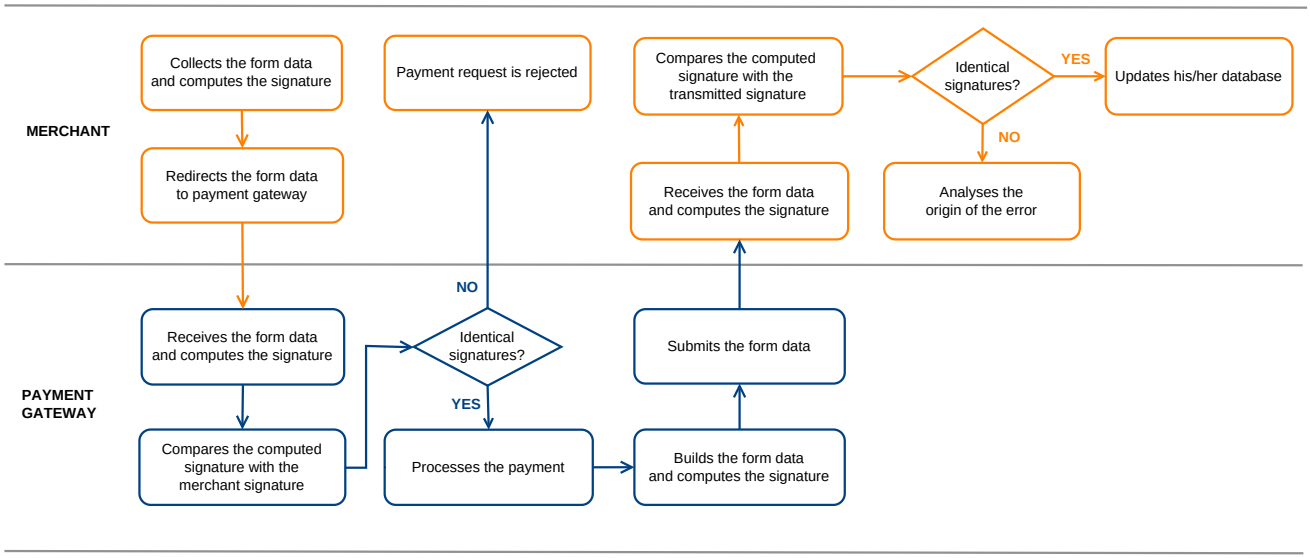


Figure 7: Diagram of a security mechanism

1. The merchant website builds the form data and computes the signature.
2. The merchant website submits the form to the gateway.
3. The gateway receives the form data and computes the signature.
4. The gateway compares the computed signature with the signature that was transmitted by the merchant website.
5. If the signatures are different, the payment request is rejected.

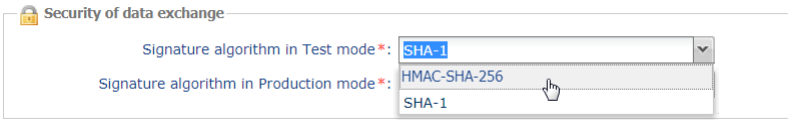
   If not, the gateway proceeds to payment.
6. The gateway builds the result data and computes the response signature.
7. Depending on the shop configuration (see chapter **Setting up notifications**), the payment gateway transmits the payment result to the merchant website.

8. The merchant website receives the data and computes the signature. It compares the computed signature with the signature that was transmitted by the payment gateway.
9. If the signatures are different, the merchant analyses the source of the error (computation error, attempted fraud, etc.).

   If not, the merchant proceeds to update their database (stock status, order status, etc.).

8.5.2. Selecting the hash algorithm

In the Expert Back Office (**Settings** > **Shop** > **Keys**), the merchant can choose the hash function to use for generating signatures.



HMAC-SHA-256 signature algorithm is applied by default.

You can select a different signature algorithm for TEST mode and for PRODUCTION mode.

However, be sure to use the same method to generate your payment forms and to analyze the data transmitted by the gateway during notifications.

In order to facilitate changing the algorithm, the SHA-1 or HMAC-SHA-256 signatures will be accepted without generating rejections due to signature error for 24h.

8.5.3. Storing the production key

For security reasons, the production key will be masked after the first real payment made with a real card.

It is strongly recommended to store the key in a safe place (encrypted file, database etc.).

In case of losing the key, the merchant will be able to regenerate a new one via their Expert Back Office.

Remember that the production key can be viewed in the Expert Back Office via **Settings** > **Shop** > **Keys** tab.

8.5.4. Managing sensitive data

Online payment transactions are regulated by strict rules (PCI-DSS certification).

As a merchant, you have to make sure to never openly transcribe data that could resemble a credit card number. Your form will be rejected (code 999 - Sensitive data detected).

Special attention should be paid to order numbers containing between 13 and 16 numeric characters and beginning with 3, 4 or 5.

8.6. Managing shop settings via a configuration file

Using a configuration file allows to avoid including hard-coded values in the code.

The configuration files may contain:

- the payment page URL,
- the test and production keys,
- the shop ID,
- etc.

These files allow to sort the data to be saved.

The program that generates the payment form interrogates the configuration file to know the value of a parameter.

It is the merchant's responsibility to do anything in his or her power to limit the access to the configuration file (.htaccess file, rewrite the URL, etc.).

Example of "conf.txt" configuration file:

```
vads_site_id = 11111111
TEST_key = 2222222222222222
PROD_key = 3333333333333333
vads_ctx_mode = TEST
```

Example of a call to configuration file in the payment form:

```
$conf_txt = parse_ini_file("conf.txt");
 if ($conf_txt['vads_ctx_mode'] == "TEST") $conf_txt['key'] = $conf_txt['TEST_key'];
 if ($conf_txt['vads_ctx_mode'] == "PRODUCTION") $conf_txt['key'] = $conf_txt['PROD_key'];
```


9. SETTING UP NOTIFICATIONS

The Expert Back Office allows to manage the events that will generate a notification to the merchant website and to configure the URL of the contact page.

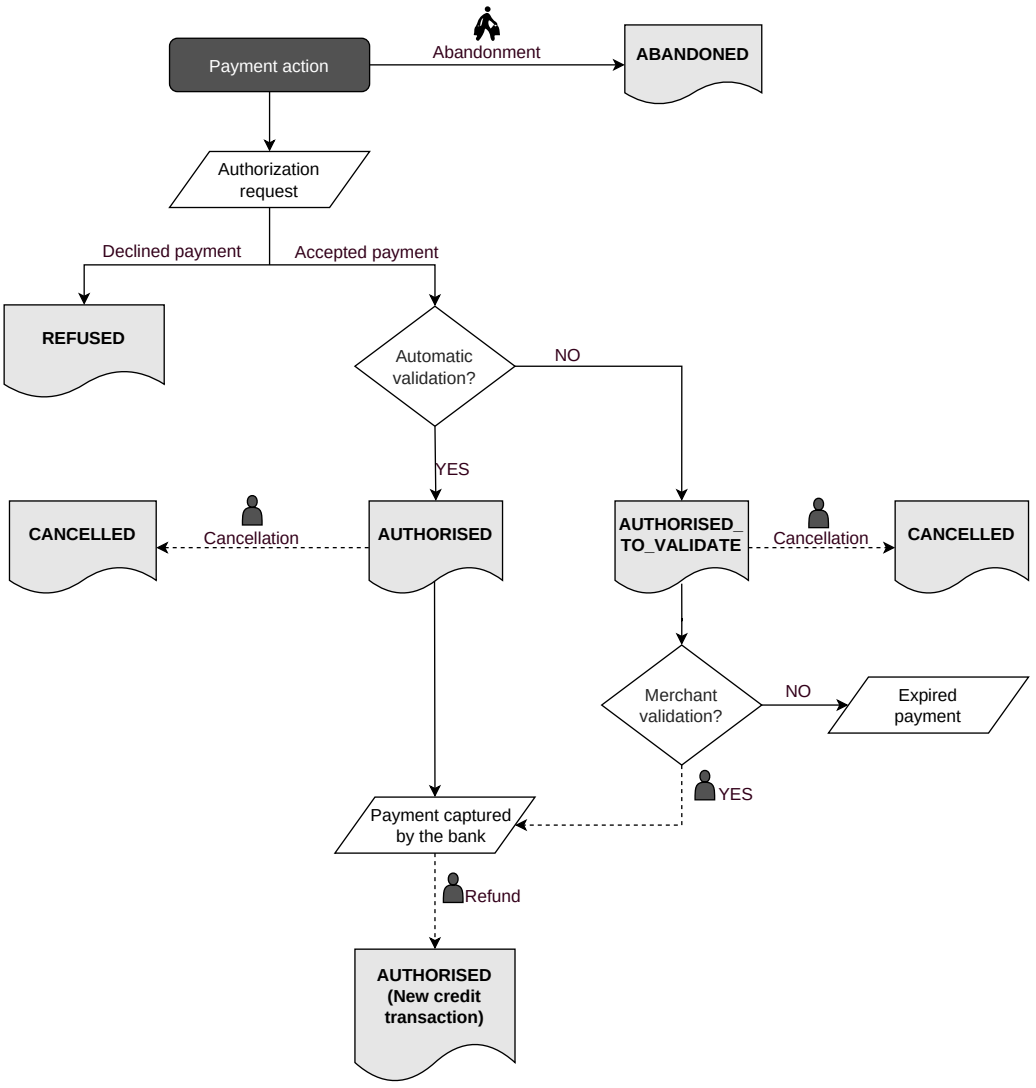
The following diagrams illustrate the transaction status sent in the notification for each event.

The following caption is used for each event:

Action required from the merchant - manual (Expert Back Office) or automatic (via Web Services)

Action performed by the buyer

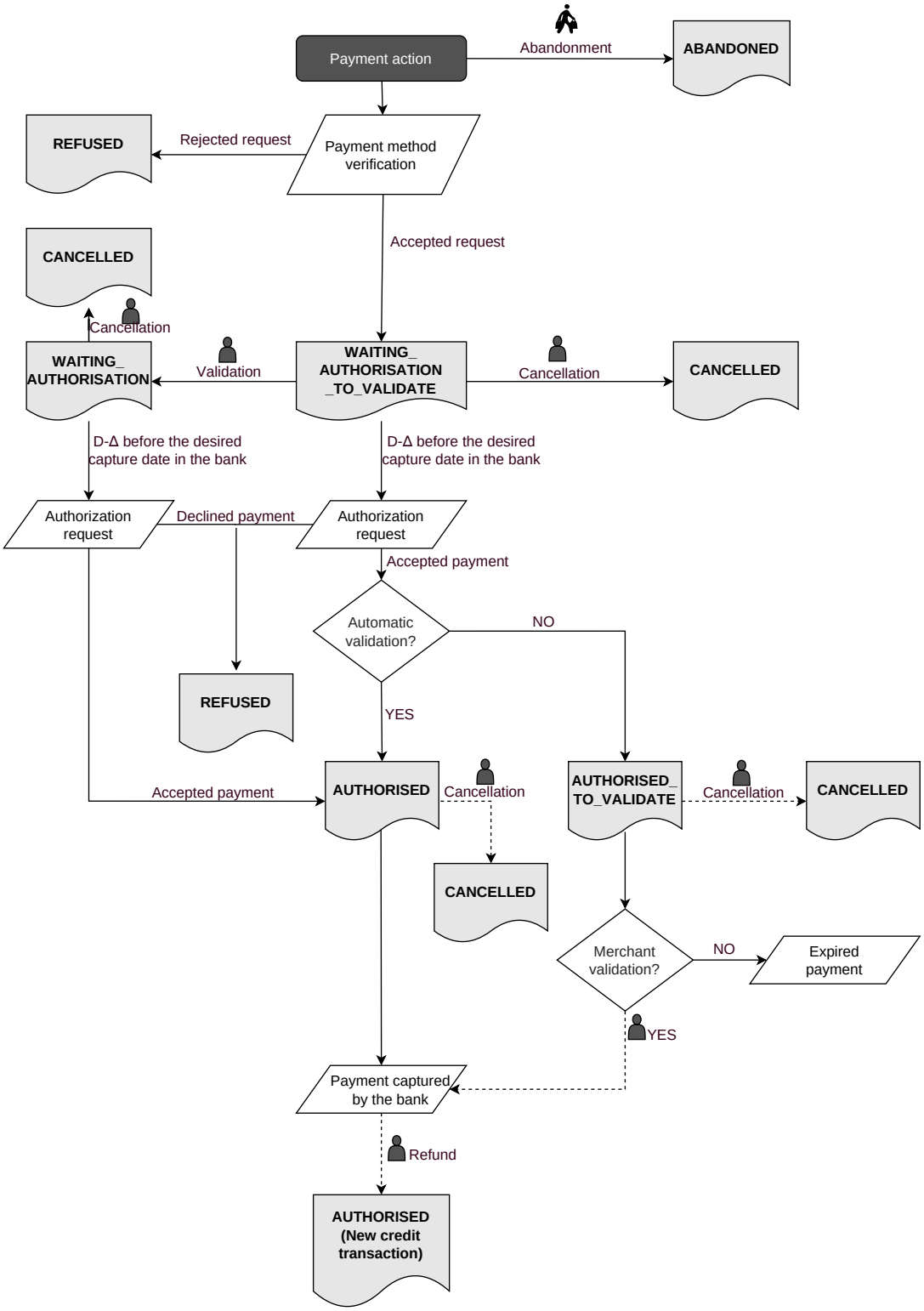
9.1. Notifications about the various statuses of an immediate payment



Event	Notified status	Name of the rule to configure
Abandoned by the buyer	ABANDONED	Instant Payment Notification URL on cancellation
Cancellation by the merchant	CANCELLED	Instant Payment Notification URL on an operation coming from the Back Office

Event	Notified status	Name of the rule to configure
Response to the authorization request	AUTHORISED_TO_VALIDATE, AUTHORISED, REFUSED	Instant Payment Notification URL at the end of the payment

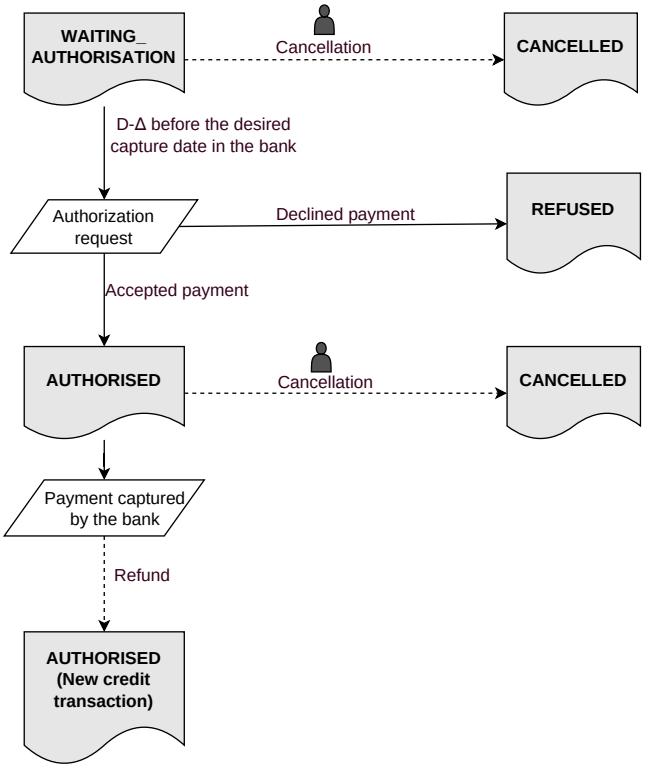
9.2. Notifications about the different statuses of a deferred payment



Δ: authorization validity period

Event	Notified status	Name of the rule to configure
Abandoned by the buyer	ABANDONED	Instant Payment Notification URL on cancellation
Cancellation by the merchant	CANCELLED	Instant Payment Notification URL on an operation coming from the Back Office
Cancellation by the merchant	WAITING_AUTHORISATION	Instant Payment Notification URL on an operation coming from the Back Office
Response to the authorization request for EUR 1 (or information request about the CB network if the acquirer supports it)	REFUSED, WAITING_AUTHORISATION, WAITING_AUTHORISATION_TO_VALIDATE	Instant Payment Notification URL at the end of the payment
Response to the authorization request	AUTHORISED, REFUSED, AUTHORISED_TO_VALIDATE	Instant Payment Notification URL on batch authorization

9.3. Notifications about the various statuses of installments



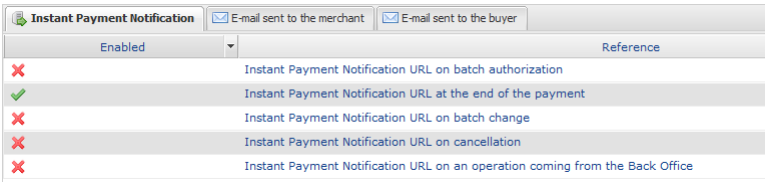
Δ: authorization validity period

Event	Notified status	Name of the rule to configure
Cancellation by the merchant	CANCELLED	Instant Payment Notification URL on an operation coming from the Back Office
Response to the authorization request	AUTHORISED, REFUSED	Instant Payment Notification URL on batch authorization

9.4. Accessing the notification center

Open the **Settings** > **Notification rules** menu.

The rule configuration tab of "Instant Payment Notification URL call" type is displayed.



9.5. Setting up the Instant Payment Notification

The payment gateway notifies on the merchant website in the following cases:

- Payment accepted
- Payment refused
- Token creation or update
- Creation of a recurring payment

The **Payment accepted** event corresponds to the creation of a transaction in one of the (vads_trans_status) statuses below:

- **ACCEPTED**
- **AUTHORISED**
- **AUTHORISED_TO_VALIDATE**
- **CAPTURED**
- **INITIAL**
- **UNDER_VERIFICATION**
- **WAITING_AUTHORISATION**
- **WAITING_AUTHORISATION_TO_VALIDATE**
- **WAITING_FOR_PAYMENT**

This notification is required to communicate the result of the payment request.

It informs the merchant website of the payment result even if your client has not clicked the **Return to the shop** button.

1. Right-click **Instant Payment Notification URL at the end of the payment**.
2. Select **Manage the rule**.
3. Enter the **E-mail address(es) to notify in case of failure** field in the "General settings" section. To specify several e-mail addresses, separate them with a semi colon (;).
4. To allow the platform to automatically resend the notification in the event of failure, check the **Automatic retry in case of failure** box. This mechanism allows up to 4 attempts to be made.

   For more information, see Automatic retry in case of failure on page 59.

5. If you wish to receive notifications in API Form format, enter the URL of your page in **URL to notify in TEST mode** and **URL to notify in PRODUCTION mode** field in the "Instant Payment Notification URL of the API form V1, V2" section.

6. If you're using the clientJavaScript, specify the URL of your page in the fields **Target URL of the IPN to notify in TEST mode** and **Target URL of the IPN to notify in PRODUCTION mode** in the "REST API Instant Payment Notification URL" section.

7. Save the changes.

9.6. Setting up a notification on batch authorization

This notification is required for communicating the result of a deferred payment:

- If the payment has been accepted,
- If the payment has been refused.

It allows the merchant website to be notified when the authorization request is not made on the payment day.

Example:

For a deferred payment with a capture delay of 60 days, the authorization request is not made at the moment of the payment. The merchant website will be contacted at the moment of the authorization request by the **Instant Payment Notification URL on batch authorization** rule.

This rule is **disabled by default**.

1. Right-click **Instant Payment Notification URL on batch authorization**.
2. Select **Manage the rule**.
3. Enter the **E-mail address(es) to notify in case of failure** field in the "General settings" section. To specify several e-mail addresses, separate them with a semi colon (;).
4. To allow the platform to automatically resend the notification in the event of failure, check the **Automatic retry in case of failure** box. This mechanism allows up to 4 attempts to be made.
   For more information, see Automatic retry in case of failure on page 59.
5. If you wish to receive notifications in API Form format, enter the URL of your page in **URL to notify in TEST mode** and **URL to notify in PRODUCTION mode** field in the "Instant Payment Notification URL of the API form V1, V2" section.
6. If you're using the clientJavaScript, specify the URL of your page in the fields **Target URL of the IPN to notify in TEST mode** and **Target URL of the IPN to notify in PRODUCTION mode** in the "REST API Instant Payment Notification URL" section.
7. Save the changes.
8. Enable the rule by right-clicking **Instant Payment Notification URL on batch authorization** and select **Enable the rule**.

9.7. Setting up notifications in case of abandoned or canceled payments

The payment gateway notifies on the merchant website in the following cases:

- When the buyer abandons/cancels a payment - via the **Cancel and return to shop** button.
- When the buyer has not completed the payment and the payment session has expired.

  The maximum length of a payment session is 10 minutes.

This customization is **mandatory** if you are using the **FacilyPay Oney** payment method.

This rule is **disabled by default**.

1. Right-click **Instant Payment Notification URL on cancellation**.
2. Select **Manage the rule**.
3. Enter the **E-mail address(es) to notify in case of failure** field in the "General settings" section. To specify several e-mail addresses, separate them with a semi colon (;).
4. To allow the platform to automatically resend the notification in the event of failure, check the **Automatic retry in case of failure** box. This mechanism allows up to 4 attempts to be made.

   For more information, see Automatic retry in case of failure on page 59.
5. If you wish to receive notifications in API Form format, enter the URL of your page in **URL to notify in TEST mode** and **URL to notify in PRODUCTION mode** field in the "Instant Payment Notification URL of the API form V1, V2" section.
6. If you're using the clientJavaScript, specify the URL of your page in the fields **Target URL of the IPN to notify in TEST mode** and **Target URL of the IPN to notify in PRODUCTION mode** in the "REST API Instant Payment Notification URL" section.
7. Save the changes.
8. Enable the rule by right-clicking **Instant Payment Notification URL on cancellation** and select **Enable the rule**.

9.8. Instant Payment Notification URL on an operation coming from the Back Office

This rule allows to notify the merchant website about every operation made via the Expert Back Office:

- Creation of a manual payment (accepted or rejected)
- Transaction update
- Transaction duplication
- Transaction refund
- Transaction cancellation
- Transaction validation
- Token creation
- Token update

1. Right-click **Instant Payment Notification URL on an operation coming from the Back Office**.
2. Select **Manage the rule**.
3. Enter the **E-mail address(es) to notify in case of failure** field in the "General settings" section. To specify several e-mail addresses, separate them with a semi colon (;).
4. To allow the platform to automatically resend the notification in the event of failure, check the **Automatic retry in case of failure** box. This mechanism allows up to 4 attempts to be made.
   For more information, see Automatic retry in case of failure on page 59.
5. If you wish to receive notifications in API Form format, enter the URL of your page in **URL to notify in TEST mode** and **URL to notify in PRODUCTION mode** field in the "Instant Payment Notification URL of the API form V1, V2" section.
6. If you're using the clientJavaScript, specify the URL of your page in the fields **Target URL of the IPN to notify in TEST mode** and **Target URL of the IPN to notify in PRODUCTION mode** in the "REST API Instant Payment Notification URL" section.
7. Save the changes.
8. Enable the rule by right-clicking **Instant Payment Notification URL on an operation coming from the Back Office** and select **Enable the rule**.

9.9. Setting up a notification on batch change

The payment gateway notifies on the merchant website in the following cases:

- When a transaction expires.

  This is the case of transactions created in manual validation mode and that have not been validated in time by the merchant. The status of these transactions changes to "Expired" (EXPIRED).

- When a **PayPal** transaction that has been blocked due to suspected fraud is finally accepted or refused.

  The status of the concerned transactions changes from "Control in progress" (UNDER_VERIFICATION) to "Captured" (CAPTURED) or"Refused" (REFUSED).

- When a **3x 4x Oney** transaction is accepted after the analysis of the financial statement.
- When a **Franfinance** transaction is accepted or refused.
- For transactions made with the following payment methods:

  Alipay, Bancontact, Giropay, iDeal, Multibanco, MyBank, Przelewy24, UnionPay, WeChat Pay.

This rule is **disabled by default**.

1. Right-click **Instant Payment Notification URL on batch change**.
2. Select **Manage the rule**.
3. Enter the **E-mail address(es) to notify in case of failure** field in the "General settings" section. To specify several e-mail addresses, separate them with a semi colon (;).
4. To allow the platform to automatically resend the notification in the event of failure, check the **Automatic retry in case of failure** box. This mechanism allows up to 4 attempts to be made.

   For more information, see Automatic retry in case of failure on page 59.
5. If you wish to receive notifications in API Form format, enter the URL of your page in **URL to notify in TEST mode** and **URL to notify in PRODUCTION mode** field in the "Instant Payment Notification URL of the API form V1, V2" section.
6. If you're using the clientJavaScript, specify the URL of your page in the fields **Target URL of the IPN to notify in TEST mode** and **Target URL of the IPN to notify in PRODUCTION mode** in the "REST API Instant Payment Notification URL" section.
7. Save the changes.
8. Enable the rule by right-clicking **Instant Payment Notification URL on batch change** and select **Enable the rule**.

9.10. Automatic retry in case of failure

Automatic retry does not apply to notifications manually triggered via the Expert Back Office.

The merchant can enable a mechanism that allows the payment gateway to automatically return notifications when the merchant website is temporarily unavailable. This mechanism allows up to 4 attempts to be made.

A notification will be considered as failed if the HTTP code returned by the merchant site is not on the following list: **200**, **201**, **202**, **203**, **204**, **205**, **206**, **301**, **302**, **303**, **307**, **308**.

Call attempts are scheduled at fixed intervals every 15 minutes (00, 15, 30, 45).

After each failed attempt, a notification e-mail is sent to the e-mail address specified in the configuration of the notification rule in question.

In this case, the subject of the e-mail contains the number corresponding to the notification retry attempt. It is presented as `attempt #` followed by the attempt number.

- Example of an e-mail subject following a first notification failure at the end of payment:

```
[MODE TEST] My Shop - Tr. ref. 067925 / FAILURE during the call to your IPN URL
 [unsuccessful attempt #1]
```

- Example of an e-mail subject following a second failure:

```
[MODE TEST] My Shop - Tr. ref. 067925 / FAILURE during the call to your IPN URL
 [unsuccessful attempt #2]
```

- Example of an e-mail subject following a third failure:

```
[MODE TEST] My Shop - Tr. ref. 067925 / FAILURE during the call to your IPN URL
 [unsuccessful attempt #3]
```

- Example of an e-mail subject following the last failure:

```
[MODE TEST] My Shop - Tr. ref. 067925 / FAILURE during the call to your IPN URL
 [unsuccessful attempt #last]
```

  To notify the merchant website of the last notification attempt, the e-mail subject will contain the mention `attempt #last`.

During the automatic retry, certain details are not stored in the database or are modified.

Examples of fields not available/not registered in the database:

Field name	Description
vads_page_action	Completed operation
vads_payment_config	Payment type (immediate or installment).
vads_action_mode	Acquisition mode for payment method data.

Examples of fields sent with different values:

Field name	New value
vads_url_check_src	Always set to **RETRY** in case of automatic retry.
vads_trans_status	The transaction status may vary between the initial call and the automatic retry (cancellation by the merchant, transaction capture at the bank, etc.).
vads_hash	The value of this field is regenerated with each call.
signature	The signature value depends on the different statuses that may vary between the initial call and the automatic retry.

These e-mails contain:

- The encountered problem;
- Parts of analysis depending on the error;
- Its consequences;
- Instructions for manually triggering the notification from the Expert Back Office.

After the fourth attempt, it is still possible to retry the IPN URL **manually** via your Expert Back Office.

Note that during the automatic retry, any manual call to the IPN URL will affect the number of automatic attempts:

- A successful manual call will stop the automatic retry;
- A failed manual call will have no impact on the current automatic retry.

9.11. Configuring e-mails sent to the merchant

In the **E-mail sent to the merchant** tab:

1. Right-click the rule to be modified and select **Enable the rule**.

2. Right-click the rule again and select **Manage the rule**.
   The rule management wizard appears.

3. Customize the label of the rule and the address to notify in the General settings section.
   To specify several e-mail addresses, separate them with a semi-colon.

4. In order to customize the body of the e-mail.
   a. Go to **E-mail settings**.
   b. Select the template of the e-mail to apply
   c. Click **Customize default text values** if you wish to edit the body and the subject of the “default” e-mail message.
   d. Click on **Fields to include** to display the list of fields available for e-mail customization.
   e. Select the fields that you wish to include. A detailed summary of the request processing will be added to the body of the e-mail.

   To preview the changes, click **Preview the e-mail** at the bottom of the dialog box.

5. In order to change the events that trigger the notification:
   a. Click the **Rule conditions** tab.
      A condition is composed of a variable, a comparison operator and a reference value.
      Example: "mode = TEST", "amount exceeding 1000". During the execution of a rule, the value of a variable is retrieved and compared to the reference value.
   b. Double-click on an existing condition to edit it.
   c. Click **Add** to create a new condition.
      All the conditions must be validated for the rule to be executed.

6. Click **Save**.

9.12. Configuring e-mails sent to the buyer

From the **E-mail sent to the buyer** tab:

1. Right-click the rule to be modified and select **Enable the rule**.

2. Right-click the rule again and select **Manage the rule**.
   The rule management wizard appears.

3. In the General settings section, you can customize the label of the rule.

4. To customize the e-mail content:
   a. Click **Buyer e-mail settings**.
   b. Select the template of the e-mail to apply
   c. Select the language that you would like to update
   d. Click **Customize default text values** if you wish to edit the body and the subject of the “default” e-mail message.
   e. Click on **Fields to include** to display the list of fields available for e-mail customization.
   f. Select the fields that you wish to include. A detailed summary of the request processing will be added to the body of the e-mail.

   To preview the changes, click **Preview the e-mail** at the bottom of the dialog box.

5. In order to change the events that trigger the notification:
   a. Click the **Rule conditions** tab.
      A condition is composed of a variable, a comparison operator and a reference value.
      Example: "mode = TEST", "amount exceeding 1000". During the execution of a rule, the value of a variable is retrieved and compared to the reference value.
   b. Double-click on an existing condition to edit it.
   c. Click **Add** to create a new condition.
      All the conditions must be validated for the rule to be executed.

6. Click **Save**.

10. GENERATING A PAYMENT FORM

You must build an HTML form as follows:

```
<form method="POST" action="https://secure.lyra.com/vads-payment/">
 <input type="hidden" name="parameter1" value="value1" />
 <input type="hidden" name="parameter2" value="value2" />
 <input type="hidden" name="parameter3" value="value3" />
 <input type="hidden" name="signature" value="signature"/>
 <input type="submit" name="pay" value="Pay"/>
</form>
```

It contains:

- The technical elements:
  - the <form> and </form> tags that allow to create an HTML form;
  - the method="POST" attribute that defines the method used for sending data;
  - the action="https://secure.lyra.com/vads-payment/" attribute that defines where to send the form data.
- Form data
  - the shop ID;
  - information about the payment depending on the use case;
  - additional information depending on your needs;
  - the signature that ensures the integrity of the form.

  This data is added to the form by using the <input> tag:

  <input type="hidden" name="parametre1" value="valeur1"/>

  For setting the name and value attributes, see chapter Data dictionary.

  All the data in the form must be encoded in UTF-8.

  This will allow for the special characters (accents, punctuation marks, etc.) to be correctly interpreted by the payment gateway. Otherwise, the signature will be computed incorrectly and the form will be rejected.

- The **Pay** button for submitting the data

  <input type="submit" name="pay" value="Pay"/>

The use cases presented in the following chapters will enable you to build your payment form according to your needs.

Indications on the different possible formats when building your form:

Notation	Description
a	Alphabetic characters (from 'A' to 'Z' and from 'a' to 'z')
n	Numeric characters
s	Special characters
an	Alphanumeric characters
ans	Alphanumeric and special characters (except < and >)
3	Fixed length of 3 characters
..12	Variable length up to 12 characters
json	JavaScript Object Notation. Object containing key/value pairs separated by commas , .

Notation	Description
	It starts with a left brace { and ends with a right brace } . Each key/value pair contains the key name in quotation marks followed by a colon, followed by a value in quotation marks "name" : "value". The name of the key must be alphanumeric. The value can be: • a string of characters (in this case it must be framed by straight quotes "); • a number; • an object; • a table; • a boolean; • empty. Example: {"name1":45,"name2":"value2", "name3":false}
bool	Boolean. Can take the value true or false.
enum	Defines a field with a complete list of values. The list of possible values is given in the field definition.
Enum list	List of values separated by a " ; ". The list of possible values is given in the field definition. Example: vads_available_languages=fr;en
map	List of key / value pairs separated by a " ; ". Each key / value pair contains the name of the key followed by " = ", followed by a value. The value can be: • a chain of characters; • a boolean; • a json object; • an xml object. The list of possible values for each key/value pair is provided in the field definition. Example: vads_theme_config=SIMPLIFIED_DISPLAY=true;RESPONSIVE_MODEL=Model_1

10.1. Creating an immediate payment

In the immediate payment mode, the buyer pays the total amount for the purchase at once.

The payment is captured by the bank on the same day.

1. Use all the fields presented in the table below to create your payment form.

Field name	Description	Format	Value
vads_site_id	Shop ID	n8	E.g.: 12345678
vads_ctx_mode	Mode of interaction with the payment gateway	enum	TEST or PRODUCTION
vads_trans_id	Transaction number. Must be unique within the same day (from 00:00:00 UTC to 23:59:59 UTC). **Warning: this field is not case sensitive.**	an6	E.g.: xrT15p
vads_trans_date	Date and time of the payment form in UTC format.	n14	Respect the YYYYMMDDHHMMSS format E.g.: 20200101130025
vads_amount	Payment amount in the smallest currency unit (cents for euro).	n..12	E.g.: 4525 for EUR 45.25
vads_currency	Numeric currency code to be used for the payment, in compliance with the ISO 4217 standard (numeric code).	n3	E.g.: 978 for euro (EUR)
vads_action_mode	Acquisition mode for payment method data	enum	INTERACTIVE
vads_page_action	Action to perform	enum	PAYMENT
vads_version	Version of the exchange protocol with the payment gateway	enum	**V2**
vads_payment_config	Payment type	enum	SINGLE
vads_payment_cards	Allows to force the card type to be used. It is recommended to provide a different payment button for each payment method on the merchant website. **It is recommended not to leave the field empty**. See the chapter Managing the payment methods offered to the buyer on page 93 for more information.	enum	E.g.: • CB • CVCONNECT • MASTERCARD • VISA • SDD
vads_capture_delay	Delay before capture in the bank.	n..3	

Field name	Description	Format	Value
vads_validation_mode	Validation mode	n1	**0** (Automatic)
signature	Signature guaranteeing the integrity of the requests exchanged between the merchant website and the payment gateway.	ans..44	Compute the value of the field **signature** using all the fields of your form that start with **vads_** (see chapter **Computing the signature**).

2. Value the field **vads_payment_config** to **SINGLE**.
3. Value the field **vads_capture_delay** to **0**.
4. Value the field **vads_validation_mode** to **0** for automatic validation (the payment will be automatically captured in the bank).
5. Value the field **vads_currency** with the code of the desired currency using the currency table.
6. Add **the fields recommended for increasing chances of frictionless** during the payment.
7. Add optional fields according to your requirements (see chapter **Using additional features**).

Example of a form for an immediate payment:

```
<form method="POST" action="https://secure.lyra.com/vads-payment/">
<input type="hidden" name="vads_action_mode" value="INTERACTIVE" />
<input type="hidden" name="vads_amount" value="15000" />
<input type="hidden" name="vads_capture_delay" value="0" />
<input type="hidden" name="vads_ctx_mode" value="TEST" />
<input type="hidden" name="vads_currency" value="978" />
<input type="hidden" name="vads_order_id" value="CX-1254" />
<input type="hidden" name="vads_page_action" value="PAYMENT" />
<input type="hidden" name="vads_payment_cards" value="CB" />
<input type="hidden" name="vads_payment_config" value="SINGLE" />
<input type="hidden" name="vads_site_id" value="12345678" />
<input type="hidden" name="vads_trans_date" value="20190626101407" />
<input type="hidden" name="vads_trans_id" value="pt156G" />
<input type="hidden" name="vads_version" value="V2" />
<input type="hidden" name="signature" value="0WaYrONo3L0VZqMcvyVf8vT/g8KfZKJ+1jqiAs3Ehiw="/>
<input type="submit" name="pay" value="Pay"/>
</form>
```


10.2. Creating a deferred payment

A deferred payment is a payment debited all at once with a capture delay that is strictly greater than 0 days.

An information request will be made if the capture delay is greater than the validity period of an authorization request (see chapter Authorization request validity period on page 22).

The information request is made in order to check the card validity. For acquirers who do not support information requests, an authorization request for EUR 1 will be made.

1. Use all the fields presented in the table below to create your payment form.

Field name	Description	Format	Value
vads_site_id	Shop ID	n8	E.g.: 12345678
vads_ctx_mode	Mode of interaction with the payment gateway	enum	TEST or PRODUCTION
vads_trans_id	Transaction number. Must be unique within the same day (from 00:00:00 UTC to 23:59:59 UTC). **Warning: this field is not case sensitive.**	an6	E.g.: xrT15p
vads_trans_date	Date and time of the payment form in UTC format.	n14	Respect the YYYYMMDDHHMMSS format E.g.: 20200101130025
vads_amount	Payment amount in the smallest currency unit (cents for euro).	n..12	E.g.: 4525 for EUR 45.25
vads_currency	Numeric currency code to be used for the payment, in compliance with the ISO 4217 standard (numeric code).	n3	E.g.: 978 for euro (EUR)
vads_action_mode	Acquisition mode for payment method data	enum	INTERACTIVE
vads_page_action	Action to perform	enum	PAYMENT
vads_version	Version of the exchange protocol with the payment gateway	enum	**V2**
vads_payment_config	Payment type	enum	SINGLE
vads_payment_cards	Allows to force the card type to be used. It is recommended to provide a different payment button for each payment method on the merchant website. **It is recommended not to leave the field empty**. See the chapter Managing the payment methods offered to the buyer on page 93 for more information.	enum	E.g.: • CB • CVCONNECT • MASTERCARD • VISA • SDD

Field name	Description	Format	Value
vads_capture_delay	Delay before capture in the bank, **the value must be greater than 0**	n..3	E.g.: 3
vads_validation_mode	Specifies the validation mode of the transaction (manually by the merchant, or automatically by the payment gateway).	n1	**0**or **1** or **absent** or **empty**
signature	Signature guaranteeing the integrity of the requests exchanged between the merchant website and the payment gateway.	ans..44	Compute the value of the field signature using all the fields of your form that start with **vads_** (see chapter **Computing the signature**).

2. Value the field **vads_payment_config** to **SINGLE**.
3. Value the field **vads_capture_delay** to a value **greater than 0**.
4. Value the field **vads_validation_mode** to **0** for an automatic validation (the payment will be automatically captured at the bank) or to **1** for a manual validation (the payment will be captured in the bank after a manual validation in the Expert Back Office).
5. Value the field **vads_currency** with the code of the desired currency using the currency table.
6. Add **the fields recommended for increasing chances of frictionless** during the payment.
7. Add optional fields according to your requirements (see chapter **Using additional features**).

Example of a form for a deferred payment:

```
<form method="POST" action="https://secure.lyra.com/vads-payment/">
<input type="hidden" name="vads_action_mode" value="INTERACTIVE" />
<input type="hidden" name="vads_amount" value="3000" />
<input type="hidden" name="vads_capture_delay" value="3" />
<input type="hidden" name="vads_ctx_mode" value="TEST" />
<input type="hidden" name="vads_currency" value="978" />
<input type="hidden" name="vads_page_action" value="PAYMENT" />
<input type="hidden" name="vads_payment_cards" value="CB" />
<input type="hidden" name="vads_payment_config" value="SINGLE" />
<input type="hidden" name="vads_site_id" value="12345678" />
<input type="hidden" name="vads_trans_date" value="20190629130025" />
<input type="hidden" name="vads_trans_id" value="Hu92ZQ" />
<input type="hidden" name="vads_version" value="V2" />
<input type="hidden" name="signature" value="NrHSHyBBBc+TtcauudspNHQ5cYcy4tS4IjvdC0ztFe8="/>
<input type="submit" name="pay" value="Pay"/>
</form>
```


10.3. Creating an installment payment

Under PSD2, strong authentication is required upon the payment of the first installment. Thevads_threeds_mpifield is ignored and theCHALLENGE_MANDATEvalue is automatically applied.

This payment mode allows the merchant to offer payment facilities to the buyer.

The payment form defines the number of installments and the interval between them.

The first installment works the same way as an immediate payment.

The next installment(s) is similar to (a) deferred payment(s).

Reminder:

Notification rules have to be activated depending on the installment. See chapter **Setting up notifications** for more information.

Details:

The field **vads_amount** contains the total order amount. This is the amount that will be split according to the value of the field **vads_payment_config**.

On the payment day, the total amount is not credited to the merchant's account and the payment guarantee cannot apply to future installments.

The date of the last installment cannot exceed one year after the date of the form submission. Otherwise, an error message will appear and the form will be rejected.

1. Use all the fields below to create your payment form.

Field name	Description	Format	Value
vads_site_id	Shop ID	n8	E.g.: 12345678
vads_ctx_mode	Mode of interaction with the payment gateway	enum	TESTorPRODUCTION
vads_trans_id	Transaction number. Must be unique within the same day (from 00:00:00 UTC to 23:59:59 UTC). **Warning: this field is not case sensitive.**	an6	E.g.: xrT15p
vads_trans_date	Date and time of the payment form in UTC format.	n14	Respect the YYYYMMDDHHMMSS format E.g.: 20200101130025
vads_amount	Payment amount in the smallest currency unit (cents for euro).	n..12	E.g.: 4525 for EUR 45.25
vads_currency	Numeric currency code to be used for the payment, in compliance with the ISO 4217 standard (numeric code).	n3	E.g.: 978 for euro (EUR)
vads_action_mode	Acquisition mode for payment method data	enum	INTERACTIVE
vads_page_action	Action to perform	enum	PAYMENT

Field name	Description	Format	Value
vads_version	Version of the exchange protocol with the payment gateway	enum	**V2**
vads_payment_config	Payment type	enum	See step 2.
vads_payment_cards	Allows to force the card type to be used. It is recommended to provide a different payment button for each payment method on the merchant website. **It is recommended not to leave the field empty**. See the chapter Managing the payment methods offered to the buyer on page 93 for more information.	enum	E.g.: • CB • MASTERCARD • VISA
vads_capture_delay	Delay before capture in the bank.	n..3	
vads_validation_mode	Specifies the validation mode of the transaction (manually by the merchant, or automatically by the payment gateway).	n1	**0**or **1** or **absent** or **empty**
signature	Signature guaranteeing the integrity of the requests exchanged between the merchant website and the payment gateway.	ans..44	Compute the value of the field**signature**using all the fields of your form that start with**vads_**(see chapter **Computing the signature**).

2. Value the field **vads_payment_config** respecting the following syntax:

   - Fixed payment amounts and dates:

     MULTI:first=1000;count=3;period=30, where:

     "first" corresponds to the amount (in the smallest currency unit) of the first installment made on the day of payment,

     "count" represents the total number of installments,

     "period" determines the interval between each installment.

   - Custom installment amounts and dates:

     MULTI_EXT:date1=amount1;date2=amount2;date3=amount3, where:

     date1=amount1 defines the date and the amount of the first transfer.

     The amounts are presented in the smallest currency unit. The total amount must be equal to the value of the field **vads_amount**.

     The dates are presented in the YYYYMMDD format.

3. Value the field **vads_capture_delay** to **0**. The first payment will be captured in the bank on the same day.

4. Value the field **vads_validation_mode** to **0** for automatic validation (the payment will be automatically captured in the bank) or to **1** for manual validation (manual operation performed via the Expert Back Office).

   The validation mode applies to all the installments.

5. Value the field **vads_currency** with the code of the desired currency using the currency table.

6. Add **the fields recommended for increasing chances of frictionless** during the payment.

7. Add optional fields according to your requirements (see chapter **Using additional features**).

Example of installment payment form (fixed amounts and payment dates):

```
<form method="POST" action="https://secure.lyra.com/vads-payment/">
<input type="hidden" name="vads_action_mode" value="INTERACTIVE" />
<input type="hidden" name="vads_amount" value="3000" />
<input type="hidden" name="vads_ctx_mode" value="TEST" />
<input type="hidden" name="vads_currency" value="978" />
<input type="hidden" name="vads_page_action" value="PAYMENT" />
<input type="hidden" name="vads_payment_config" value="MULTI:first=1000;count=3;period=30"/>
<input type="hidden" name="vads_site_id" value="12345678" />
<input type="hidden" name="vads_trans_date" value="20190629180150" />
<input type="hidden" name="vads_trans_id" value="1N015m" />
<input type="hidden" name="vads_version" value="V2" />
<input type="hidden" name="signature" value= "zrhUNkAciZSEl6mS4BbhV3qkYUBB9RYJQCdg1kU0ELU="/>
<input type="submit" name="pay" value="Pay" />
</form>
```

Example of installment payment form (custom amounts and payment dates):

```
<form method="POST" action="https://secure.lyra.com/vads-payment/">
<input type="hidden" name="vads_action_mode" value="INTERACTIVE" />
<input type="hidden" name="vads_amount" value="3000" />
<input type="hidden" name="vads_capture_delay" value="0" />
<input type="hidden" name="vads_ctx_mode" value="TEST" />
<input type="hidden" name="vads_currency" value="" />
<input type="hidden" name="vads_page_action" value="PAYMENT" />
<input type="hidden" name="vads_payment_config" value="
MULTI_EXT:20140201=1000;20140301=1000;20140401=1000" />
<input type="hidden" name="vads_site_id" value="12345678" />
<input type="hidden" name="vads_trans_date" value="20190629130025" />
<input type="hidden" name="vads_trans_id" value="130025" />
<input type="hidden" name="vads_version" value="V2" />
<input type="hidden" name="signature" value="7Sds6Z+R1Q1axRsblpChyQh5OU3oCle5FOirD4V/Bzk="/>
<input type="submit" name="pay" value="Pay"/>
</form>
```


10.4. Creating an authorization without capture

This payment mode allows to make sure that the buyer's card data is correct without debiting it.

If needed, the merchant can debit the desired amount from the card account using the **Duplicate** function of the **Expert Back Office** and proceed to manual validation.

1. Use all the fields of the table below to create your payment form.

Field name	Description	Format	Value
vads_site_id	Shop ID	n8	E.g.: 12345678
vads_ctx_mode	Mode of interaction with the payment gateway	enum	TEST or PRODUCTION
vads_trans_id	Transaction number. Must be unique within the same day (from 00:00:00 UTC to 23:59:59 UTC). **Warning: this field is not case sensitive.**	an6	E.g.: xrT15p
vads_trans_date	Date and time of the payment form in UTC format.	n14	Respect the YYYYMMDDHHMMSS format E.g.: 20200101130025
vads_amount	Payment amount in the smallest currency unit (cents for euro).	n..12	E.g.: 4525 for EUR 45.25
vads_currency	Numeric currency code to be used for the payment, in compliance with the ISO 4217 standard (numeric code).	n3	E.g.: 978 for euro (EUR)
vads_action_mode	Acquisition mode for payment method data	enum	INTERACTIVE
vads_page_action	Action to perform	enum	PAYMENT
vads_version	Version of the exchange protocol with the payment gateway	enum	**V2**
vads_payment_config	Payment type	enum	SINGLE
vads_capture_delay	Delay before capture in the bank.	n..3	
vads_validation_mode	Validation mode	n1	**1** (Manual)
signature	Signature guaranteeing the integrity of the requests exchanged between the merchant website and the payment gateway.	ans..44	Compute the value of the field signature using all the fields of your form that start with **vads_** (see chapter **Computing the signature**).

2. Set the **vads_amount** field to a small amount. It will not affect the authorization limit of the card.
3. Value the field **vads_validation_mode** to **1**.
4. Value the field **vads_currency** with the code of the desired currency using the currency table.
5. Add optional fields according to your requirements (see chapter **Using additional features**).

Example of a form for an authorization without capture:

```
<form method="POST" action="https://secure.lyra.com/vads-payment/">
<input type="hidden" name="vads_action_mode" value="INTERACTIVE" />
<input type="hidden" name="vads_amount" value="100" />
<input type="hidden" name="vads_capture_delay" value="0" />
<input type="hidden" name="vads_ctx_mode" value="TEST" />
<input type="hidden" name="vads_currency" value="978" />
<input type="hidden" name="vads_page_action" value="PAYMENT" />
<input type="hidden" name="vads_validation_mode" value="1"/>
<input type="hidden" name="vads_site_id" value="12345678" />
<input type="hidden" name="vads_trans_date" value="20190628073753" />
<input type="hidden" name="vads_trans_id" value="3jj7A8" />
<input type="hidden" name="vads_version" value="V2" />
<input type="hidden" name="signature" value= "DvltInRYXRroOZ/KnNdJSlpVr++29ZGty4nj1Y7yczU="/>
<input type="submit" name="pay" value="Pay" />
</form>
```


11. USING ADDITIONAL FEATURES

11.1. Managing the return to the merchant website

At the end of payment, the buyer has the possibility to return to the merchant website via a return URL. This URL is called **Return URL**.

It is not to be confused with **Instant notification URL (IPN)** (see chapter **Managing the interaction with the merchant website**).

11.1.1. Defining the Return URLs

In the payment form, the merchant can override the configuration of the Expert Back Office. To do so, the merchant can:

- Use 4 different URLs depending on the payment result:
  - Payment accepted
  - Payment declined
  - Payment abandoned
  - Payment error
- Or use a single URL independently of the payment result.

11.1.1.1. Defining the return URLs depending on the payment result

Use the optional fields presented in the table below to create a customized payment form.

If no URL is specified in the form, the value populated in the Expert Back Office will be used.

Field name	Description	Format	Value
vads_url_cancel	URL to which the buyer will be redirected upon clicking on “Cancel and return to shop” before proceeding to the payment.	ans..1024	E.g.: http://demo.com/cancel.php
vads_url_error	URL to which the buyer will be redirected in case of a processing error on the payment gateway.	ans..1024	E.g.: http://demo.com/error.php
vads_url_refused	URL to which the buyer will be redirected in case of a declined payment after having clicked on “Return to shop”.	ans..1024	E.g.: http://demo.com/refused.php
vads_url_success	URL to which the buyer will be redirected in case of an accepted payment after having clicked on “Return to shop”.	ans..1024	E.g.: http://demo.com/success.php

Example of a payment form with configuration of a return URL depending on the payment result:

```
<form method="POST" action="https://secure.lyra.com/vads-payment/">
<input type="hidden" name="vads_action_mode" value="INTERACTIVE" />
<input type="hidden" name="vads_amount" value="3000" />
<input type="hidden" name="vads_capture_delay" value="0" />
<input type="hidden" name="vads_ctx_mode" value="PRODUCTION" />
<input type="hidden" name="vads_currency" value="978" />
<input type="hidden" name="vads_page_action" value="PAYMENT" />
<input type="hidden" name="vads_payment_config" value="SINGLE" />
<input type="hidden" name="vads_site_id" value="12345678" />
<input type="hidden" name="vads_trans_date" value="20191126101407" />
<input type="hidden" name="vads_trans_id" value="pm197W" />
<input type="hidden" name="vads_url_cancel" value="http://demo.com/cancel.php" />
<input type="hidden" name="vads_url_error" value="http://demo.com/error.php" />
<input type="hidden" name="vads_url_refused" value="http://demo.com/refused.php" />
<input type="hidden" name="vads_url_success" value="http://demo.com/success.php" />
<input type="hidden" name="vads_version" value="V2" />
<input type="hidden" name="signature" value="lZIHzigiwCc6+uLStp8I5DQnbSqXu63Jtfo6Saeq3Mc="/>
<input type="submit" name="pay" value="Pay"/>
</form>
```

11.1.1.2. Setting up a unique return URL regardless of the payment outcome

Use the optional field **vads_url_return** to set up a redirection URL at the end of payment.

If no URL is specified in the form, the value populated in the Expert Back Office will be used.

Example of a payment form with a unique return URL regardless of the payment result:

```
<form method="POST" action="https://secure.lyra.com/vads-payment/">
<input type="hidden" name="vads_action_mode" value="INTERACTIVE" />
<input type="hidden" name="vads_amount" value="3000" />
<input type="hidden" name="vads_capture_delay" value="0" />
<input type="hidden" name="vads_ctx_mode" value="PRODUCTION" />
<input type="hidden" name="vads_currency" value="978" />
<input type="hidden" name="vads_page_action" value="PAYMENT" />
<input type="hidden" name="vads_payment_config" value="SINGLE" />
<input type="hidden" name="vads_site_id" value="12345678" />
<input type="hidden" name="vads_trans_date" value="20191126101407" />
<input type="hidden" name="vads_trans_id" value="xrTYh2" />
<input type="hidden" name="vads_url_return" value="http://demo.com/return.php" />
<input type="hidden" name="vads_version" value="V2" />
<input type="hidden" name="signature" value="ZI/GhI0GbeqqoXGeoZuPOy55SKQSYzRO1i6r5ku6vOs="/>
<input type="submit" name="pay" value="Pay"/>
</form>
```


11.1.2. Defining the method for receiving data

For statistical purposes or to display customized pages, the merchant site must be able to analyze certain data transmitted to the buyer's browser.

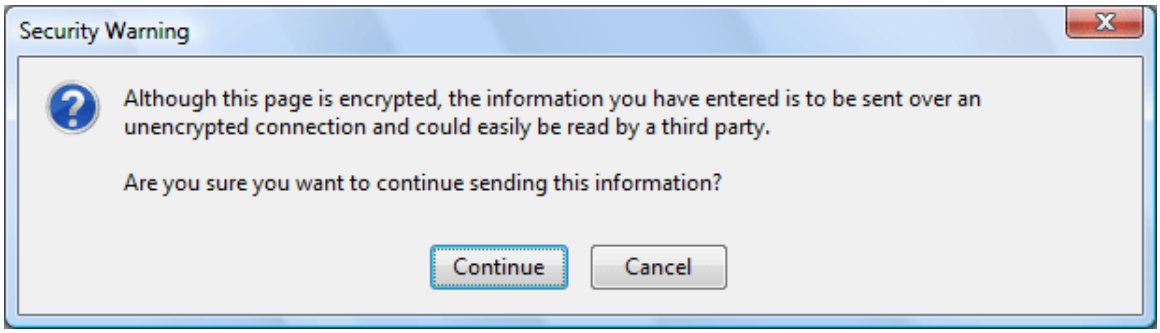
By default, the payment gateway does not transmit any data when redirecting to the return URL.

However, the merchant website may activate the transmission of data to the return URL via the payment form.

Use the **vads_return_mode** optional field to specify the method for submitting data to the merchant website.

Value	Description
Absent, empty or **NONE**	No data is transmitted.
GET	The data is transmitted in the browser URL.
POST	The date is transmitted via an HTTP in request **POST**.

The method **GET** allows to keep a notification message from appearing when the return is done from an **insecure environment (http)**.



Example of a payment form with definition of the mode for data transmission:

```
<form method="POST" action="https://secure.lyra.com/vads-payment/">
<input type="hidden" name="vads_action_mode" value="INTERACTIVE" />
<input type="hidden" name="vads_amount" value="3000" />
<input type="hidden" name="vads_capture_delay" value="0" />
<input type="hidden" name="vads_ctx_mode" value="PRODUCTION" />
<input type="hidden" name="vads_currency" value="978" />
<input type="hidden" name="vads_page_action" value="PAYMENT" />
<input type="hidden" name="vads_payment_config" value="SINGLE" />
<input type="hidden" name="vads_return_mode" value="GET" />
<input type="hidden" name="vads_site_id" value="12345678" />
<input type="hidden" name="vads_trans_date" value="20190626101407" />
<input type="hidden" name="vads_trans_id" value="239848" />
<input type="hidden" name="vads_url_return" value="http://demo.com/return.php" />
<input type="hidden" name="vads_version" value="V2" />
<input type="hidden" name="signature" value="oTCT+7Oc+xttdGmcp9qa6/0pSSfNxoMtl8U1J1l+LtE="/>
<input type="submit" name="pay" value="Pay"/>
</form>
```

11.2. Enabling automatic return to the merchant website

In the payment form, the merchant can indicate if he/she wishes to automatically redirect the buyer to the merchant website at the end of payment.

If you use a tracking code (e.g. Google Analytics™) on your website, you must implement this function.

1. Use optional fields according to your requirements.

Field name	Description
vads_redirect_success_timeout	Defines the delay before redirection following an accepted payment. This delay is presented in seconds and must be between 0 and 300 sec.
vads_redirect_success_message	Defines the message that appears before redirection following a successful payment.
vads_redirect_error_timeout	Defines the delay before the redirection that follows a declined payment. This delay is presented in seconds and must be between 0 and 300 sec.
vads_redirect_error_message	Defines the message that appears before the redirection that follows a declined payment.

If you set the timeout to zero (= 0 delay) your redirection will be done as follows:

- *For an **accepted payment**, the buyer will be redirected to **vads_url_success**.*
- *For an **cancelled payment**, the buyer will be redirected to **vads_url_cancel**if the parameter is defined.*
  - *If the parameter is not defined, the buyer will be redirected to the return URL entered in the **vads_url_return** field or to the return URL specified in the Expert Back Office.*
  - *If the return URL is not set, the buyer will be redirected to the merchant website.*
- *For an **Declined payment**, the buyer will be redirected to **vads_url_refused** if the parameter is defined.*

2. Set the **vads_return_mode** field to **GET**.

Example of a payment form:

```
<form method="POST" action="https://secure.lyra.com/vads-payment/">
<input type="hidden" name="vads_action_mode" value="INTERACTIVE" />
<input type="hidden" name="vads_amount" value="3000" />
<input type="hidden" name="vads_capture_delay" value="0" />
<input type="hidden" name="vads_ctx_mode" value="PRODUCTION" />
<input type="hidden" name="vads_currency" value="978" />
<input type="hidden" name="vads_page_action" value="PAYMENT" />
<input type="hidden" name="vads_payment_config" value="SINGLE" />
<input type="hidden" name="vads_redirect_error_message" value=" You will be redirected to your
 merchant website" />
<input type="hidden" name="vads_redirect_error_timeout" value="0" />
<input type="hidden" name="vads_redirect_success_message" value=" You will be redirected to
 your merchant website" />
<input type="hidden" name="vads_redirect_success_timeout" value="0" />
<input type="hidden" name="vads_return_mode" value="GET" />
<input type="hidden" name="vads_site_id" value="12345678" />
<input type="hidden" name="vads_trans_date" value="20190626101407" />
<input type="hidden" name="vads_trans_id" value="AL3d8Q" />
<input type="hidden" name="vads_url_return" value="http://demo.com/return.php" />
<input type="hidden" name="vads_version" value="V2" />
<input type="hidden" name="signature" value="AzTJmizS5N0muYzu63nVvCUWo0ixnMJfpqQmuEa4CSY="/>
<input type="submit" name="pay" value="Pay"/>
</form>
```


11.3. Defining the capture mode (automatic/manual)

In the Expert Back Office, the merchant can configure how payments are sent to the bank (**Settings** > **Shop** menu > **Configuration** tab):

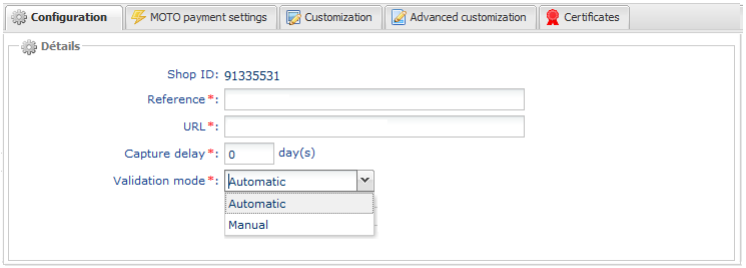


Figure 8: Defining the capture mode

- **Automatic:** no action is necessary, the payments are captured in the bank once the capture delay has been reached.
- **Manual:** the merchant must validate each payment via their Expert Back Office or Web Services, so that it is captured in the bank before the authorization request expires.

  Each transaction that has not been validated by the expected date is considered as expired and will never be captured in the bank.

  By default, the Expert Back Office is configured to automatically submit all payments to the bank.

  The merchant can override this configuration in their payment form.

  The merchant must implement the desired criteria (stock status, delay for stock replenishment, etc.) allowing the buyer to decide whether the transaction must be captured automatically or not.

  Use field **vads_validation_mode** to configure the capture mode (manual or automatic).

  This field will be resent with the response and will include the value transmitted in the form.

Value	Description
Missing or empty	Takes the value specified in the Expert Back Office.
0	Automatic capture. Transaction is automatically validated by the payment gateway.
1	Manual capture. The transaction must be validated manually by the merchant via their Expert Back Office (or automatically via the **Transaction/Validate** Web Service function).

Example of a payment form with a definition of the capture mode in INTERACTIVE mode:

```
<form method="POST" action="https://secure.lyra.com/vads-payment/">
<input type="hidden" name="vads_action_mode" value="INTERACTIVE" />
<input type="hidden" name="vads_amount" value="4000" />
<input type="hidden" name="vads_capture_delay" value="0" />
<input type="hidden" name="vads_ctx_mode" value="TEST" />
<input type="hidden" name="vads_currency" value="978" />
<input type="hidden" name="vads_page_action" value="PAYMENT" />
<input type="hidden" name="vads_payment_config" value="SINGLE" />
<input type="hidden" name="vads_site_id" value="12345678" />
<input type="hidden" name="vads_trans_date" value="20190626164147" />
<input type="hidden" name="vads_trans_id" value="164147" />
<input type="hidden" name="vads_version" value="V2" />
<input type="hidden" name="vads_validation_mode" value="1" />
<input type="hidden" name="signature" value="cJFhNTLXQ4o6BgbW1pMMoM2yMilw90OIqmFjJ6DeUmA= />
<input type="submit" name="pay" value="Pay"/>
</form>
```


11.4. Transmitting buyer details

Buyer details (e-mail address, title, phone number, etc.) constitute billing information.

All transmitted data is displayed in the transaction details (tab **Client**) in the Merchant Back Office.

Use optional fields according to your requirements. *These fields will be returned with the response and will include the value transmitted in the form.*

Field name	Description	Format	Value
vads_cust_email	Buyer's e-mail address	ans..150	E.g.: abc@example.com
vads_cust_id	Buyer reference on the merchant website	an..63	E.g.: C2383333540
vads_cust_national_id	National identifier	ans..255	E.g.: 940992310285
vads_cust_title	Buyer's title	an..63	E.g.: M
vads_cust_status	Status	enum	PRIVATE: for private clients **COMPANY**: for companies
vads_cust_first_name	First name	ans..63	E.g.: Laurent
vads_cust_last_name	Name	ans..63	E.g.: Durant
vads_cust_legal_name	Buyer's legal name	ans..100	E.g.: D. & Cie
vads_cust_phone	Phone number	an..32	E.g.: 0467330222
vads_cust_cell_phone	Cell phone number	an..32	E.g.: 06 12 34 56 78
vads_cust_address_number	Street number	ans..64	E.g.: 109
vads_cust_address	Postal address	ans..255	E.g.: Rue de l'innovation
vads_cust_address2	Address line 2	ans..255	E.g.:
vads_cust_district	District	ans..127	E.g.: Centre ville
vads_cust_zip	Zip code	an..64	E.g.: 31670
vads_cust_city	City	an..128	E.g.: Labège
vads_cust_state	State / Region	ans..127	E.g.: Occitanie
vads_cust_country	Country code in compliance with the ISO 3166 alpha-2 standard	a2	E.g.: "FR"for France,"PF"for French Polynesia,"NC"for New Caledonia, "US"for the United States.

Example of payment form with buyer details:

```
<form method="POST" action="https://secure.lyra.com/vads-payment/">
<input type="hidden" name="vads_action_mode" value="INTERACTIVE" />
<input type="hidden" name="vads_amount" value="4000" />
<input type="hidden" name="vads_capture_delay" value="0" />
<input type="hidden" name="vads_ctx_mode" value="PRODUCTION" />
<input type="hidden" name="vads_currency" value="978" />
<input type="hidden" name="vads_cust_country" value="FR" />
<input type="hidden" name="vads_cust_email" value="smith.john@example.com" />
<input type="hidden" name="vads_cust_first_name" value="John" />
<input type="hidden" name="vads_cust_last_name" value="Smith" />
<input type="hidden" name="vads_cust_title" value="Mr" />
<input type="hidden" name="vads_page_action" value="PAYMENT" />
<input type="hidden" name="vads_payment_config" value="SINGLE" />
<input type="hidden" name="vads_site_id" value="12345678" />
<input type="hidden" name="vads_trans_date" value="20190627133115" />
<input type="hidden" name="vads_trans_id" value="522754" />
```



```
<input type="hidden" name="vads_version" value="V2" />
<input type="hidden" name="signature" value="rEFhNTLXQ4o6BgbW1pTMoM2yMilw90OIqmFjJ6DeCxP="/>
<input type="submit" name="pay" value="Pay"/>
</form>
```


11.5. Transmitting shipping details

The buyer's shipping details are the address, title, phone number, etc.

This information can be found in the transaction details in the Merchant Back Office (**Client tab**).

Use optional fields according to your requirements.

These fields will be returned with the response and will include the value transmitted in the form.

Field name	Description	Format	Value
vads_ship_to_city	City	an..128	E.g.: Bordeaux
vads_ship_to_country	Country code in compliance with the ISO 3166 standard (required for triggering one or more actions if the **Shipping country control** profile is enabled).	a2	E.g.: FR
vads_ship_to_district	District	ans..127	E.g.: La Bastide
vads_ship_to_first_name	First name	ans..63	E.g.: Albert
vads_ship_to_last_name	Name	ans..63	E.g.: Durant
vads_ship_to_legal_name	Legal name	an..100	E.g.: D. & Cie
vads_ship_to_phone_num	Phone number	ans..32	E.g.: 0460030288
vads_ship_to_state	State / Region	ans..127	E.g.: Nouvelle aquitaine
vads_ship_to_status	Allows to specify the type of the shipping address.	enum	**PRIVATE** : for shipping to a private individual **COMPANY** : for shipping to a company
vads_ship_to_street_number	Street number	ans..64	E.g.: 2
vads_ship_to_street	Postal address	ans..255	E.g.: Rue Sainte Catherine
vads_ship_to_street2	Address line 2	ans..255	
vads_ship_to_zip	Zip code	an..64	E.g.: 33000

Example of payment form with shipping details:

```
<form method="POST" action="https://secure.lyra.com/vads-payment/">
<input type="hidden" name="vads_action_mode" value="INTERACTIVE" />
<input type="hidden" name="vads_amount" value="4000" />
<input type="hidden" name="vads_capture_delay" value="0" />
<input type="hidden" name="vads_ctx_mode" value="PRODUCTION" />
<input type="hidden" name="vads_currency" value="978" />
<input type="hidden" name="vads_page_action" value="PAYMENT" />
<input type="hidden" name="vads_payment_config" value="SINGLE" />
<input type="hidden" name="vads_ship_to_city" value=" city of the shipping address" />
<input type="hidden" name="vads_ship_to_country" value="FR" />
<input type="hidden" name="vads_ship_to_name" value="location name of the shipping address" />
<input type="hidden" name="vads_ship_to_street" value="street of the shipping address" />
<input type="hidden" name="vads_ship_to_street_number" value="10" />
<input type="hidden" name="vads_ship_to_zip" value="31670" />
<input type="hidden" name="vads_site_id" value="12345678" />
<input type="hidden" name="vads_trans_date" value="20190627143509" />
<input type="hidden" name="vads_trans_id" value="561095" />
<input type="hidden" name="vads_version" value="V2" />
<input type="hidden" name="signature" value="bOIxHAgm4vYUq3oIDCdEPKOWgrB9bHzkfDBEAr1i10A="/>
<input type="submit" name="pay" value="Pay"/>
</form>
```


11.6. Transmitting order details

The merchant can indicate in their payment form if they wish to transfer the order details (order reference, description, shopping cart contents, etc.).

1. Use optional fields according to your requirements.

Field name	Description	Format	Value
vads_order_id	Order ID Can contain uppercase or lowercase characters, numbers or hyphens ([A-Z] [a-z], 0-9, _, -).	ans..64	E.g.: 2-XQ001
vads_order_info	Additional order info	ans..255	E.g.: Door code 3125
vads_order_info2	Additional order info	ans..255	E.g.: No elevator
vads_order_info3	Additional order info	ans..255	E.g.: Express
vads_nb_products	Number of items in the cart	n..12	E.g.: 2
vads_product_ext_idN	Product barcode on the merchant website. N corresponds to the index of the item (0 for the first one, 1 for the second one, etc.).	an..100	E.g.: vads_product_ext_id0 = "0123654789123654789" vads_product_ext_id1 = "0223654789123654789" vads_product_ext_id2 = "0323654789123654789"
vads_product_labelN	Item name. N corresponds to the index of the item (0 for the first one, 1 for the second one, etc.).	ans..255	E.g.: vads_product_label0 = "tee-shirt" vads_product_label1 = "Biscuit" vads_product_label2 = "Sandwich"
vads_product_amountN	Price of the item incl. VAT. N corresponds to the index of the item (0 for the first one, 1 for the second one, etc.).	n..12	E.g.: vads_product_amount0 = "1200" vads_product_amount1 = "800" vads_product_amount2 = "950"
vads_product_typeN	Item type. N corresponds to the index of the item (0 for the first one, 1 for the second one, etc.).	enum	E.g.: vads_product_type0 = "CLOTHING_AND_ACCESSORIES" vads_product_type1 = "FOOD_AND_GROCERY" vads_product_type2 = "FOOD_AND_GROCERY"
vads_product_refN	Item reference. N corresponds to the index of the item (0 for the first one, 1 for the second one, etc.).	an..64	E.g.: vads_product_ref0 = "CAA-25-006" vads_product_ref1 = "FAG-B5-112" vads_product_ref2 = "FAG-S9-650"
vads_product_qtyN	Item quantity. N corresponds to the index of the item (0 for	n..12	E.g.: vads_product_qty0 = "1" vads_product_qty1 = "2"

Field name	Description	Format	Value
	the first one, 1 for the second one, etc.).		vads_product_qty2 = "2"
vads_product_vatN	Amount or VAT rate applied to the item. N corresponds to the index of the item (0 for the first one, 1 for the second one, etc.). *The decimal separator is mandatory for displaying a rate.* *The decimal separator is represented by the "." symbol.*	n..12	• Populated with an integer, without a decimal separator, for expressing an amount. E.g.: 4525 for EUR 45.25 • Populated with a decimal number less than 100, for expressing a rate. E.g.: 20.0 or 19.6532

2. Value the field vads_nb_products where the number of the articles in the cart.

> We recommend to make populating the field mandatory for taking the cart into account. This implies populating the other fields starting with vads_product_ to see the details of the shopping cart.

3. Value the field **vads_product_amountN** with the amount for the items in the cart, using the smallest currency unit.

   N corresponds to the index of the item (0 for the first one, 1 for the second one, etc.).

4. Value the field **vads_product_typeN** with the value corresponding to the item type.

   N corresponds to the index of the item (0 for the first one, 1 for the second one, etc.).

Value	Description
FOOD_AND_GROCERY	Food and grocery
AUTOMOTIVE	Cars / Moto
ENTERTAINMENT	Entertainment / Culture
HOME_AND_GARDEN	Home / Gardening
HOME_APPLIANCE	Household appliances
AUCTION_AND_GROUP_BUYING	Auctions / Group purchasing
FLOWERS_AND_GIFTS	Flowers / Presents
COMPUTER_AND_SOFTWARE	Computers / Software
HEALTH_AND_BEAUTY	Health / Beauty
SERVICE_FOR_INDIVIDUAL	Services for individuals
SERVICE_FOR_BUSINESS	Services for companies
SPORTS	Sports
CLOTHING_AND_ACCESSORIES	Clothes / Accessories
TRAVEL	Travel
HOME_AUDIO_PHOTO_VIDEO	Audio / Photo / Video
TELEPHONY	Telephony

5. Value the field **vads_product_labelN** with the label of each article in the cart.
   N corresponds to the index of the item (0 for the first one, 1 for the second one, etc.).

6. Value the field **vads_product_qtyN** with the quantity of each article in the cart.
   N corresponds to the index of the item (0 for the first one, 1 for the second one, etc.).

7. Value the field **vads_product_refN** with the reference of each article in the cart.
   N corresponds to the index of the item (0 for the first one, 1 for the second one, etc.).

8. Control the value of the field **vads_amount**. It must correspond to the total amount of the order.

Example of the payment form with cart description "vads_product_xxx" :

```
<form method="POST" action="https://secure.lyra.com/vads-payment/">
<input type="hidden" name="vads_action_mode" value="INTERACTIVE" />
<input type="hidden" name="vads_amount" value="11000" />
<input type="hidden" name="vads_capture_delay" value="0" />
<input type="hidden" name="vads_ctx_mode" value="PRODUCTION" />
<input type="hidden" name="vads_currency" value="978" />
<input type="hidden" name="vads_nb_products" value="2"/>
<input type="hidden" name="vads_product_amount0" value="5000" />
<input type="hidden" name="vads_product_label0" value="produit1" />
<input type="hidden" name="vads_product_qty0" value="2" />
<input type="hidden" name="vads_product_ref0" value="ref1" />
<input type="hidden" name="vads_product_amount1" value="1000" />
<input type="hidden" name="vads_product_label1" value="produit2" />
<input type="hidden" name="vads_product_qty1" value="1" />
<input type="hidden" name="vads_product_ref1" value="ref2" />
<input type="hidden" name="vads_order_id" value="CD100000857" />
<input type="hidden" name="vads_page_action" value="PAYMENT" />
<input type="hidden" name="vads_payment_config" value="SINGLE" />
<input type="hidden" name="vads_site_id" value="12345678" />
<input type="hidden" name="vads_trans_date" value="20190627145218" />
<input type="hidden" name="vads_trans_id" value="571381" />
<input type="hidden" name="vads_version" value="V2" />
<input type="hidden" name="signature" value="xYw1UnU3BACGhf3UEyqbQzpwuvZDEkCAWAE5fgbtfxI="/>
<input type="submit" name="payer" value="Payer"/></form>
```


11.7. Transmitting merchant preferences

Use field **vads_threeds_mpi** to transmit your preference:

Use case	Values	Description
CHALLENGE: with cardholder interaction	**1**	**Deprecated**.
	3	**3DS Requestor Preference:** allows to request strong authentication for the transaction.
	4	**Challenge request mandate:** allows to indicate that, due to regulatory reasons, strong authentication is required for the transaction.
FRICTIONLESS: without cardholder interaction	**2***	Allows to request an exemption from strong authentication: • Low value transactions. • Transaction Risk Analysis(TRA Acquirer). • Safe'R by CB. More information: Table of exemptions, below.
No merchant preference	**0 or absent or empty**	The choice of the preference is transferred to the card issuer. If the issuer decides to perform an authentication without interaction (frictionless), the payment will be guaranteed.
	5	

* **Table of exemptions:**

Exemptions	Description
Low value transactions	In Europe, you can request an exemption from strong authentication, for transactions of less than EUR 30, and within the limit of either 5 successive operations or a cumulative amount of less than EUR 100. If the amount is higher than EUR 30, the value transmitted by the merchant is ignored and the choice of the preference is transferred to the card issuer (No Preference). For payments made in a currency other than euro, a request for frictionless is transmitted to the issuer. If the frictionless request is accepted, the transaction does not benefit from liability shift dispute by the cardholder.. If the store does not have the “Frictionless 3DS2” option, the choice of the preference is transferred to the card issuer (No Preference).
Transaction Risk Analysis(TRA Acquirer)	If your store has the “TRA 3DS2 Acquirer” option, you can ask the issuer for an exemption from strong authentication if the amount is below the threshold set by your financial institution. If the frictionless request is accepted, the transaction does not benefit from liability shift dispute by the cardholder.. The “TRA 3DS2 Acquirer” activation option is subject to the prior agreement of your financial institution.
Safe'R by CB	CB offers theSafe'R by CBprogram. This program is designed to meet the needs of very low-risk, high-volume merchants. You can request an exemption from strong authentication: • If the amount is less than EUR 100, the exemption is systematic for eligible merchants.

Exemptions	Description

	• If the amount is between EUR 100 and EUR 250, an experiment is underway. To qualify, the merchant must:
	  • Have a CB contract.
	  • Be eligible for acquirer TRA.
	  • Transmit the required values in the 3D Secure flow, according to the rules defined by the platform.
	If the frictionless request is accepted, the transaction does not benefit from liability shift dispute by the cardholder..
	To benefit from theSafe'R by CBprogram, contact sales administration to get an explicit agreement.

11.8. Overriding the Instant Payment notification (IPN) URL

You can override the Instant Payment Notification (also called IPN) in the payment form in case you use one shop for various sales channels, payment types, languages, etc.

This feature is not compatible with the execution of the request sent to the IPN via the Expert Back Office. The called URL is the URL that was set up in the notification rule (see chapter **Setting up notifications**).

Use field **vads_url_check** to override the URL of the page to notify.

If the value of the field **vads_url_check** is wrong, the form will be rejected.

Example of payment form with IPN override:

```
<form method="POST" action="https://secure.lyra.com/vads-payment/">
<input type="hidden" name="vads_action_mode" value="INTERACTIVE" />
<input type="hidden" name="vads_amount" value="3000" />
<input type="hidden" name="vads_capture_delay" value="0" />
<input type="hidden" name="vads_ctx_mode" value="PRODUCTION" />
<input type="hidden" name="vads_currency" value="978" />
<input type="hidden" name="vads_page_action" value="PAYMENT" />
<input type="hidden" name="vads_payment_config" value="SINGLE" />
<input type="hidden" name="vads_site_id" value="12345678" />
<input type="hidden" name="vads_trans_date" value="20190626101407" />
<input type="hidden" name="vads_trans_id" value="239848" />
<input type="hidden" name="vads_url_check" value="http://www.myshop.com/check" />
<input type="hidden" name="vads_version" value="V2" />
<input type="hidden" name="signature" value="yXvSZnYvcMRORVGiapWaHT0euKDI0OGlrddYKc4XDZc="/>
<input type="submit" name="pay" value="Pay"/>
</form>
```


11.9. Defining the Merchant ID (MID)

In the payment form, the Merchant must specify the value of the Merchant ID (MID).

This feature is used only if you have several MIDs that accept the same currency within the same acceptance network.

Use the **vads_contracts** optional field to define the Merchant ID to be used.

- To **define a list** of MIDs, separate them with a semi-colon ";".

```
vads_contracts=NETWORK_CODE_A=MID_A1;NETWORK_CODE_B=MID_B2
```

- To **exclude a network**, add **network name**= **NO**.

```
vads_contracts=NETWORK_CODE_A=NO
```

- To **force the TID**, separate the MID number and the TID number by a colon: ":"

```
vads_contracts=NETWORK_CODE_A=MID_A1:TID_1
```

- If the field is submitted empty, the MID used will be the one defined by the priority order in the Expert Back Office (**Settings** > **Shop** > **MID association** tab).

List of available networks:

Network code	Description
ACCORD_SANDBOX	Oney network (private and gift cards) - sandbox mode
ACCORD	Oney network (private and gift cards)
ALIPAY_PLUS	Alipay + network
ALMA	ALMA network
AMEXGLOBAL	American Express network
AURORE	Cetelem Aurore network (Brand cards and universal Aurore card)
BIZUM	Bizum Bancário network
CB	CB network
CONECS	Titre-Restaurant Conecs network
COFIDIS	Cofidis network
CVCONNECT	Chèque-Vacances Connect network
DFS	DFS (Discover Financial Services) network
EDENRED	Edenred network (Tickets Restaurant, Tickets EcoChèque, Tickets Compliments)
FRANFINANCE	Franfinance network
FRANFINANCE_SB	Franfinance network - Sandbox mode)
FULLCB	FULL CB Network (Payment in 3 or 4 times without fees by BNPP PF)
GATECONEX	GATECONEX Network
GICC_DINERS	GICC network (Diners Club cards)
GICC_MAESTRO	GICC network (Maestro cards)

Network code	Description
GICC_MASTERCARD	GICC network (Mastercard cards)
GICC_VISA	GICC network (Visa cards)
GICC	GICC network
IP	Payment initiation (SEPA Credit Transfer and SEPA Instant Credit Transfer) network
JCB	JCB network
LYRA_COLLECT_PPRO	PPRO network
MULTIBANCO	MULTIBANCO network
NPCIUPI	UPI network
ONEY_API	Oney API network
ONEY_API_SB	Oney API network - Sandbox mode
ONEY_SANDBOX	Oney Network (Payment in 3 or 4 installments by FacilyPay) - sandbox mode
ONEY	Oney Network (Payment in 3 or 4 installments by FacilyPay)
PAYCONIQ	Payconiq network
PAYDIREKT_V2	PayDirekt V2 network
PAYPAL	PayPal network
PAYPAL_SB	PayPal network - sandbox mode
PLANET_DCC	Planet network
POSTFINANCEV2	POSTFINANCE network
PRESTO	Presto network
REDSYS_REST	RedSys REST network
SEPA	SEPA network (SDD and SCT)

Examples:

You have:

- Two MIDs within the A network: MID_A1 and MID_A2
- Two MIDs within the B network: MID_B1 and MID_B2

To specify which MID to use for these two networks, **vads_contracts** must be populated as follows:

```
vads_contracts=A=MID_A2;B=MID_B1
```

To offer a payment only for the MID_A1 contract and prevent payments within the B network, populate **vads_contracts** as follows:

```
vads_contracts=A=MID_A1;B=NO
```

In order to force the TID to be used within the A network:

```
vads_contracts=A=MID_A1:TID_A1
```


11.10. Creating specific fields according to your requirements

The merchant can transmit specific information in the payment form. For example, the merchant can add information in the payment confirmation e-mail that he or she will receive.

This information will be visible in the Back Office, in transaction details (**Extras** tab), and will also be returned in the notification URL.

The name must start with **vads_ext_info** to be taken into account.

vads_ext_info_*fieldname*=value

1. Use the fields required for your use case (see chapter **Generating a payment form**) to create your payment form.

2. Use the optional field **vads_ext_info** depending on your needs and respecting the syntax:

   vads_ext_info_*fieldname*=*value*,

   where:

   - ***fieldname***

     Allows to define the name of the field.

   - ***value***

     Allows to define the value of the field.

   There are no restrictions to the number of specific fields that can be created.

   This/these field(s) will be returned with the response and will include the value transmitted in the form.

3. Compute the value of the field signature using all the fields of your form that start with **vads_** (see chapter **Computing the signature**).

```
<form method="POST" action="https://secure.lyra.com/vads-payment/">
<input type="hidden" name="vads_action_mode" value="INTERACTIVE" />
<input type="hidden" name="vads_amount" value="4000" />
<input type="hidden" name="vads_capture_delay" value="0" />
<input type="hidden" name="vads_ctx_mode" value="TEST" />
<input type="hidden" name="vads_currency" value="978" />

[...]
<input type="hidden" name="vads_ext_info_qty_articles" value="2" /> />
[...]
<input type="hidden" name="vads_page_action" value="PAYMENT" />
<input type="hidden" name="vads_payment_config" value="SINGLE" />
<input type="hidden" name="vads_site_id" value="12345678" />
<input type="hidden" name="vads_trans_date" value="20150826133115" />
<input type="hidden" name="vads_trans_id" value="722754" />
<input type="hidden" name="vads_version" value="V2" />
<input type="hidden" name="signature" value="7896adcaf7338930db9715afa123531f42"/>
<input type="submit" name="pay" value="Pay"/>
</form>
```


11.11. Transmitting sub-merchant details

The payment facilitator can transmit the details of the sub-merchant involved in the transaction.

Field name	Description	Format
vads_submerchant_address	Address of the sub-merchant. Transmitted by the payment facilitator.	ans..255
vads_submerchant_address2	Address line 2 of the sub-merchant. Transmitted by the payment facilitator.	ans..255
vads_submerchant_city	City of the sub-merchant. Transmitted by the payment facilitator.	ans..128
vads_submerchant_company_type	Company type of the sub-merchant. Transmitted by the payment facilitator.	ans..60
vads_submerchant_country	Country of the sub-merchant's address (ISO 3166 alpha-2 standard). Transmitted by the payment facilitator.	a2
vads_submerchant_facilitatorId	Payment Facilitator ID. Transmitted by the payment facilitator.	ans..128
vads_submerchant_legal_number	Legal Entity Identifier of the sub-merchant. Transmitted by the payment facilitator.	ans..24
vads_submerchant_mcc	Merchant Category Code of the sub-merchant. Transmitted by the payment facilitator.	n4
vads_submerchant_mid	Merchant ID number of the sub-merchant. Transmitted by the payment facilitator.	n..64
vads_submerchant_name	Legal name of the sub-merchant. Transmitted by the payment facilitator.	ans..255
vads_submerchant_phone	Phone number of the sub-merchant. Transmitted by the payment facilitator.	ans..32
vads_submerchant_soft_descriptor	Soft descriptor of the sub-merchant that appears on the buyer's bank statement. Transmitted by the payment facilitator.	ans..255
vads_submerchant_state	Region of the sub-merchant address. Transmitted by the payment facilitator.	ans..128
vads_submerchant_url	URL of the sub-merchant. Transmitted by the payment facilitator.	ans..128
vads_submerchant_zip	Zip code of the sub-merchant. Transmitted by the payment facilitator.	an..64

12. CUSTOMIZING ELEMENTS ON THE PAYMENT PAGE

Allows to customize certain elements on the payment page:

- the payment methods offered at the moment of payment,
- the language used for displaying the payment pages,
- the languages offered to the buyer on the payment pages,
- the name and the URL of the shop,
- button labels.

Thanks to the **advanced customization** option, you can also:

- create different custom templates of the payment page in order to make it look more similar to your merchant website,
- create different custom templates of e-mails sent to the buyer,
- customize certain labels that appear on the payment pages.

This will result in a better user experience during redirection to proceed to payment.

Consult the Advanced customization user manual for more details or contact sales administration.

12.1. Overriding the custom template

The Expert Back Office allows:

- to create several custom templates of payment pages,
- to define the template that will be used by default for all your transactions.

The payment form allows to dynamically override the template to be used thanks to the field **vads_theme_config**.

For this, you must use the keyword: **RESPONSIVE_MODEL** and indicate the name of the template to be used (Model_1,Model_2, ...).

Example of use:

```
<input type="hidden" name="vads_theme_config" value="RESPONSIVE_MODEL=Model_1" />
```

See the *Back Office user manual - Advanced customization* for more details on template creation.

See the vads_theme_config field description for more details on using this field.

12.2. Managing the payment methods offered to the buyer

It is possible to customize the payment methods offered to the buyer by using the **vads_payment_cards** field.

It is recommended to provide a different payment button for each payment method on the merchant website and to transmit the buyer's choice in the **vads_payment_cards**.

The list of possible values is available in the Data dictionary.

For more information, please consult the documentation dedicated to each payment method that you wish to offer.

To offer payment by CB, Visa, Mastercard, Maestro, Visa Electron and e-CB cards, we recommend to only submit the **CB** value.

To offer payment by card via European acquirers (Elavon, Six, Concardis, VR Pay, etc.), we recommend to submit the "**VISA**" or "**MASTERCARD**" value.

Thus, the buyer is redirected to the card data entry page, and the card type is automatically detected.

It is strongly recommended not to leave this field empty. In case of adding a new payment method to your shop, it will be offered automatically, even if you do not wish to offer it.

Example of a payment form with payment method selection:

```
<form method="POST" action="https://secure.lyra.com/vads-payment/">
<input type="hidden" name="vads_action_mode" value="INTERACTIVE" />
<input type="hidden" name="vads_amount" value="30000" />
<input type="hidden" name="vads_capture_delay" value="0" />
<input type="hidden" name="vads_ctx_mode" value="PRODUCTION" />
<input type="hidden" name="vads_currency" value="978" />
<input type="hidden" name="vads_page_action" value="PAYMENT" />
<input type="hidden" name="vads_payment_cards" value="CB" />
<input type="hidden" name="vads_payment_config" value="SINGLE" />
<input type="hidden" name="vads_site_id" value="12345678" />
<input type="hidden" name="vads_trans_date" value="20190626101407" />
<input type="hidden" name="vads_trans_id" value="239848" />
<input type="hidden" name="vads_version" value="V2" />
<input type="hidden" name="signature" value="qqpxF6z1+Ri5jtkHNVDCCJulxxpJYehrfP1OLwJ4Ysg="/>
<input type="submit" name="pay" value="Pay"/>
</form>
```


12.3. Selecting a different language

You can customize the language of the payment pages.

Set the**vads_language**field to one of the values presented in the table below.

Language	ISO 639-1 standard
German	de
English	en
Chinese	zh
Spanish	es
French	fr
Italian	it
Japanese	ja
Dutch	nl
Polish	pl
Portuguese	pt
Russian	ru
Swedish	sv
Turkish	tr

- If the value of the **vads_language** field is wrong, the form will be rejected.
- If the field has not been sent or is empty, the payment page will be shown in the language of the buyer's browser.
- The buyer will be able to change the language anytime by using the language selector in the top right corner of the payment page.

Example of a payment form with a list of available languages:

```
<form method="POST" action="https://secure.lyra.com/vads-payment/">
<input type="hidden" name="vads_action_mode" value="INTERACTIVE" />
<input type="hidden" name="vads_amount" value="3000" />
<input type="hidden" name="vads_capture_delay" value="0" />
<input type="hidden" name="vads_ctx_mode" value="PRODUCTION" />
<input type="hidden" name="vads_currency" value="978" />
<input type="hidden" name="vads_language" value="fr" />
<input type="hidden" name="vads_page_action" value="PAYMENT" />
<input type="hidden" name="vads_payment_config" value="SINGLE" />
<input type="hidden" name="vads_site_id" value="12345678" />
<input type="hidden" name="vads_trans_date" value="20190626101407" />
<input type="hidden" name="vads_trans_id" value="239848" />
<input type="hidden" name="vads_version" value="V2" />
<input type="hidden" name="signature" value="PAMdHJ8FJc2CqUJLXQLxz+e77K4k1YGJmI5mHqGN74g="/>
<input type="submit" name="pay" value="Pay"/>
</form>
```


12.4. Modifying the languages available to the buyer

You can customize the list of languages offered to the buyer using the language selector at the top right of the payment page.

The last language selected by the buyer will be the default language for the payment confirmation e-mail.

Set the **vads_available_languages** field using the table below:

- with one single value, if you do not wish to show the page of payment method selection,
- **with a list of values separated by a ";"** to show the available languages.

Language	Value	Default available language
German	de	x
English	en	x
Chinese	zh	x
Spanish	es	x
French	fr	x
Italian	it	x
Japanese	ja	x
Dutch	nl	x
Polish	pl	
Portuguese	pt	x
Russian	ru	x
Swedish	sv	x
Turkish	tr	

If the value of the field **vads_available_languages** is wrong, the form will be rejected.

Example of a payment form with a list of available languages:

```
<form method="POST" action="https://secure.lyra.com/vads-payment/">
<input type="hidden" name="vads_action_mode" value="INTERACTIVE" />
<input type="hidden" name="vads_amount" value="3000" />
<input type="hidden" name="vads_available_languages" value="fr;en;nl;de" />
<input type="hidden" name="vads_capture_delay" value="0" />
<input type="hidden" name="vads_ctx_mode" value="PRODUCTION" />
<input type="hidden" name="vads_currency" value="978" />
<input type="hidden" name="vads_page_action" value="PAYMENT" />
<input type="hidden" name="vads_payment_config" value="SINGLE" />
<input type="hidden" name="vads_site_id" value="12345678" />
<input type="hidden" name="vads_trans_date" value="20190626101407" />
<input type="hidden" name="vads_trans_id" value="239848" />
<input type="hidden" name="vads_version" value="V2" />
<input type="hidden" name="signature" value="aEWutqzuHH6Q8ns3a6cj5XitZCuhYsDcsKjlLpL8flA="/>
<input type="submit" name="pay" value="Pay"/>
</form>
```


12.5. Modifying the name and the URL of the shop

If you have two domain names, you can modify the name and the URL of the shop to make the domain name visible.

1. Use field **vads_shop_name** to override the name of the shop that appears on the summary payment page, the receipt and the confirmation e-mail.

2. Use field **vads_shop_url** to modify the shop URL that appears on the payment pages.

   This value will be used for the confirmation e-mail.

   If the value of the field **vads_shop_url** is wrong, the form will not be rejected.

Example of a payment form including the modification of the shop name and URL:

```
<form method="POST" action="https://secure.lyra.com/vads-payment/">
<input type="hidden" name="vads_action_mode" value="INTERACTIVE" />
<input type="hidden" name="vads_amount" value="3000" />
<input type="hidden" name="vads_capture_delay" value="0" />
<input type="hidden" name="vads_ctx_mode" value="PRODUCTION" />
<input type="hidden" name="vads_currency" value="978" />
<input type="hidden" name="vads_page_action" value="PAYMENT" />
<input type="hidden" name="vads_payment_config" value="SINGLE" />
<input type="hidden" name="vads_shop_name" value="My Shop" />
<input type="hidden" name="vads_shop_url" value="http://www.myshop.com" />
<input type="hidden" name="vads_site_id" value="12345678" />
<input type="hidden" name="vads_trans_date" value="20190626101407" />
<input type="hidden" name="vads_trans_id" value="239848" />
<input type="hidden" name="vads_version" value="V2" />
<input type="hidden" name="signature" value="gV0f2HZzQ9BxttHM2W5ZM+AKQsxu0HjDvKy0NAE/G24="/>
<input type="submit" name="pay" value="Pay"/>
</form>
```


12.6. Changing the name of the "Return to shop" button

You can customize the text of the button **"Return to the shop"**.

1. Use field **vads_theme_config** to change the name of the "Return to shop" button.

2. Use the keyword **SUCCESS_FOOTER_MSG_RETURN** to change the name of the "Return to shop" button that appears if the payment has been accepted.

3. Use the keyword **CANCEL_FOOTER_MSG_RETURN** to change the name of the "Cancel and return to shop" button that appears on payment pages.

By subscribing to the **advanced customization** option, you will be able to change the names (e.g.: shop ID) of the buttons on the payment page.

See: Back Office user manual Advanced customization for more details or contact sales administration.

Example of payment form with modification of the name of the "Return to shop" button:

```
<form method="POST" action="https://secure.lyra.com/vads-payment/">
<input type="hidden" name="vads_action_mode" value="INTERACTIVE" />
<input type="hidden" name="vads_amount" value="4000" />
<input type="hidden" name="vads_capture_delay" value="0" />
<input type="hidden" name="vads_ctx_mode" value="PRODUCTION" />
<input type="hidden" name="vads_currency" value="978" />
<input type="hidden" name="vads_order_id" value="CD100000858" />
<input type="hidden" name="vads_page_action" value="PAYMENT" />
<input type="hidden" name="vads_payment_config" value="SINGLE" />
<input type="hidden" name="vads_site_id" value="12345678" />
<input type="hidden" name="vads_theme_config"
 value="CANCEL_FOOTER_MSG_RETURN=Cancel;SUCCESS_FOOTER_MSG_RETURN=Return" />
<input type="hidden" name="vads_trans_date" value="20190631092024" />
<input type="hidden" name="vads_trans_id" value="408248" />
<input type="hidden" name="vads_version" value="V2" />
<input type="hidden" name="signature" value="ge5DHBbUGsq4cFfSIR1QyB/L/9qPNp2vhX9/G3kKJeQ="/>
<input type="submit" name="pay" value="Pay"/>
</form>
```


13. COMPUTING THE SIGNATURE

To be able to compute the signature, you must have:

- all the fields that start with `vads_`
- the signature algorithm chosen in the shop configuration
- the **key**

The key value is available in your Expert Back Office via **Settings** > **Shop** > **Keys** tab.

The signature algorithm is defined in your Expert Back Office via **Settings** > **Shop** > **Configuration** tab.

For maximum security, it is recommended to use HMAC-SHA-256 algorithm and an alphanumeric key.

The use of SHA-1 algorithm is deprecated but maintained for compliance reasons.

To compute the signature:

1. Sort the fields that start with `vads_` alphabetically.
2. Make sure that all the fields are encoded in UTF-8.
3. Concatenate the values of these fields separating them with the "**+**" character.
4. Concatenate the result with the test or production key separating them with a "**+**".
5. According to the signature algorithm defined in your shop configuration:
   a. If your shop is configured to use "SHA-1", apply the **SHA-1** hash function to the chain obtained during the previous step **Deprecated.**
   b. If your shop is configured to use "HMAC-SHA-256", compute and encode in Base64 format the message signature using the **HMAC-SHA-256** algorithm with the following parameters:
      - the SHA-256 hash function,
      - the test or production key (depending on the value of the `vads_ctx_mode` field) as a shared key,
      - the result of the previous step as the message to authenticate.
6. Save the result of the previous step in the `signature` field.

Example of parameters sent to the payment gateway:

```
<form method="POST" action="https://secure.lyra.com/vads-payment/">
<input type="hidden" name="vads_action_mode" value="INTERACTIVE" />
<input type="hidden" name="vads_amount" value="5124" />
<input type="hidden" name="vads_ctx_mode" value="TEST" />
<input type="hidden" name="vads_currency" value="978" />
<input type="hidden" name="vads_page_action" value="PAYMENT" />
<input type="hidden" name="vads_payment_config" value="SINGLE" />
<input type="hidden" name="vads_site_id" value="12345678" />
<input type="hidden" name="vads_trans_date" value="20170129130025" />
<input type="hidden" name="vads_trans_id" value="123456" />
<input type="hidden" name="vads_version" value="V2" />
<input type="hidden" name="signature" value="ycA5Do5tNvsnKdc/eP1bj2xa19z9q3iWPy9/rpesfS0="/>

<input type="submit" name="pay" value="Pay"/>
</form>
```

This sample form is analyzed as follows:

1. The fields whose names start with vads_ are sorted **alphabetically**:
   - vads_action_mode
   - vads_amount
   - vads_ctx_mode
   - vads_currency
   - vads_page_action
   - vads_payment_config
   - vads_site_id
   - vads_trans_date
   - vads_trans_id
   - vads_version
2. The values of these fields are concatenated using the "**+**" character:

   ```
   INTERACTIVE+5124+TEST+978+PAYMENT+SINGLE+12345678+20170129130025+123456+V2
   ```

3. The value of the test key is added at the end of the chain and separated with the "**+**" character. In this example, the test key is **1122334455667788**

   ```
   INTERACTIVE+5124+TEST+978+PAYMENT+SINGLE+12345678+20170129130025+123456+V2+1122334455667788
   ```

4. If you use the SHA-1 algorithm, apply it to the obtained chain.

   The result that must be transmitted in the signature field is: **59c96b34c74b9375c332b0b6a32e6deeec87de2b**

5. If your shop is configured to use "HMAC-SHA-256", compute and encode in Base64 format the message signature using the **HMAC-SHA-256** algorithm with the following parameters:
   - the SHA-256 hash function,
   - the test or production key (depending on the value of the vads_ctx_mode field) as a shared key,
   - the result of the previous step as the message to authenticate.

   The result that must be transmitted in the signature field is:

   ycA5Do5tNvsnKdc/eP1bj2xa19z9q3iWPy9/rpesfS0=

13.1. Example of implementation with JAVA

Definition of the utility class SHA that will include the elements required for processing the HMAC-SHA-256 algorithm:

```
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
import java.io.UnsupportedEncodingException;
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import java.util.Base64;
import java.util.TreeMap;

public class VadsSignatureExample {
 /**
  * Build signature (HMAC SHA-256 version) from provided parameters and secret key.
  * Parameters are provided as a TreeMap (with sorted keys).
  */
 public static String buildSignature(TreeMap<String, String> formParameters, String
  secretKey) throws NoSuchAlgorithmException, InvalidKeyException, UnsupportedEncodingException
 {
   // Build message from parameters
   String message = String.join("+", formParameters.values());
   message += "+" + secretKey;
   // Sign
   return hmacSha256Base64(message, secretKey);
  }
    /**
    * Actual signing operation.
    */
 public static String hmacSha256Base64(String message, String secretKey) throws
  NoSuchAlgorithmException, InvalidKeyException, UnsupportedEncodingException {
   // Prepare hmac sha256 cipher algorithm with provided secretKey
   Mac hmacSha256;
   try {
    hmacSha256 = Mac.getInstance("HmacSHA256");
   } catch (NoSuchAlgorithmException nsae) {
    hmacSha256 = Mac.getInstance("HMAC-SHA-256");
   }
   SecretKeySpec secretKeySpec = new SecretKeySpec(secretKey.getBytes("UTF-8"), "HmacSHA256");
   hmacSha256.init(secretKeySpec);
   // Build and return signature
   return Base64.getEncoder().encodeToString(hmacSha256.doFinal(message.getBytes("UTF-8")));
  }
}
```

Definition of the utility class SHA that will include the elements required for processing the SHA-1 algorithm:

```
import java.security.MessageDigest;
import java.security.SecureRandom;

public class Sha {
    static public final String SEPARATOR = "+" ;
    public static String encode(String src) {
        try {
            MessageDigest md;
            md = MessageDigest.getInstance( "SHA-1" );
            byte bytes[] = src.getBytes( "UTF-8" );
            md.update(bytes, 0, bytes. length );
            byte[] sha1hash = md.digest();
            return convertToHex(sha1hash);
        }
        catch(Exception e){
            throw new RuntimeException(e);
        }
    }
    private static String convertToHex(byte[] sha1hash) {
        StringBuilder builder = new StringBuilder();
            for (int i = 0; i < sha1hash. length ; i++) {
               byte c = sha1hash[i];
               addHex(builder, (c >> 4) & 0xf);
               addHex(builder, c & 0xf);
        }
        return builder.toString();
    }
    private static void addHex(StringBuilder builder, int c) {
        if (c < 10)
            builder.append((char) (c + '0' ));
        else
            builder.append((char) (c + 'a' - 10));
    }
```



```
}
```


Function that computes the signature:

```
public ActionForward performCheck(ActionMapping actionMapping, Basivoirorm form,
    HttpServletRequest request, HttpServletResponse response){
        SortedSet<String> vadsFields = new TreeSet<String>();
        Enumeration<String> paramNames = request.getParameterNames();

        // retrieve and sort the fields starting with vads_* alphabetically
        while (paramNames.hasMoreElements()) {
            String paramName = paramNames.nextElement();
            if (paramName.startsWith( "vads_" )) {
                vadsFields.add(paramName);
            }
        }
// Compute the signature
String sep = Sha.SEPARATOR;
StringBuilder sb = new StringBuilder();
for (String vadsParamName : vadsFields) {
    String vadsParamValue = request.getParameter(vadsParamName);
    if (vadsParamValue != null) {
        sb.append(vadsParamValue);
    }
    sb.append(sep);
}
sb.append( shaKey );
String c_sign = Sha.encode(sb.toString());
return c_sign;}
```


13.2. Example of implementation with PHP

Example of signature computation using the HMAC-SHA-256 algorithm:

```
function getSignature ($params,$key)
{
    /**
     *Function that computes the signature.
     * $params : table containing the fields to send in the payment form.
     * $key : TEST or PRODUCTION key
     */
    //Initialization of the variable that will contain the string to encrypt
    $signature_content = "";

    //sorting fields alphabetically
    ksort($params);
    foreach($params as $name=>$value){

        //Recovery of vads_ fields
        if (substr($name,0,5)=='vads_'){

            //Concatenation with "+"
            $signature_content .= $value."+";
         }
    }
    //Adding the key at the end
    $signature_content .= $key;

    //Encoding base64 encoded chain with SHA-256 algorithm
    $signature = base64_encode(hash_hmac('sha256',$signature_content, $key, true));
    return $signature;
 }
```

Example of signature computation using the SHA-1 algorithm:

```
function getSignature($params, $key)
{
    /**
     * Function that computes the signature.
     * $params : table containing the fields to send in the payment form.
     * $key : TEST or  PRODUCTION key
     */
    //Initialization of the variable that will contain the string to encrypt
    $signature_content = "" ;

    // Sorting fields alphabetically
    ksort($params);
        foreach ($params as $name =>$value)
    {
    // Recovery of vads_ fields
        if (substr($name,0,5)=='vads_') {
            // Concatenation with "+"
            $signature_content .= $value."+";
        }
    }
    // Adding the key at the end
    $signature_content .= $key;

    // Applying SHA-1 algorithm
    $signature = sha1($signature_content);
    return $signature ;
}
```


14. SENDING THE PAYMENT REQUEST

To finalize a purchase, the buyer must be redirected to the payment page.

His browser must transmit the payment form data.

14.1. Redirecting the buyer to the payment page

The URL of the payment gateway is:

https://secure.lyra.com/vads-payment/

Example of parameters sent to the payment gateway:

```
<form method="POST" action="https://secure.lyra.com/vads-payment/">
<input type="hidden" name="vads_action_mode" value="INTERACTIVE" />
<input type="hidden" name="vads_amount" value="1315" />
<input type="hidden" name="vads_currency" value="978" />
<input type="hidden" name="vads_cust_id" value="1234" />
<input type="hidden" name="vads_cust_email" value="jg@sample.com" />
<input type="hidden" name="vads_ctx_mode" value="TEST" />
<input type="hidden" name="vads_order_id" value="CMD012859" />
<input type="hidden" name="vads_page_action" value="PAYMENT" />
<input type="hidden" name="vads_payment_cards" value="VISA;MASTERCARD" />
<input type="hidden" name="vads_payment_config" value="SINGLE" />
<input type="hidden" name="vads_site_id" value="12345678" />
<input type="hidden" name="vads_trans_date" value="20200326101407" />
<input type="hidden" name="vads_trans_id" value="362812" />
<input type="hidden" name="vads_version" value="V2" />
<input type="hidden" name="signature" value="NM25DPLKEbtGEHCDHn8MBT4ki6aJI/ODaWhCzCnAfvY="/>
<input type="submit" name="pay" value="Pay"/>
</form>
```

14.2. Processing errors

If the payment gateway detects an error while receiving the form, an error message is displayed and the buyer can not be to proceed to the payment.

In test mode:

The message indicates the source of the error and provides a link to the error code description to help you fix it.

In production mode:

The message indicates to the buyer that a technical problem has occurred.

In both cases, the merchant receives a warning e-mail containing the information:

- the source of the error;
- a link to possible causes to facilitate its analysis;
- all the fields of the form.

The e-mail is sent to the company administrator.

To change this address or add an address, contact sales administration.

You can also create a personalized notification rule to receive this e-mail at another address.

To do so:

1. Sign in to your Expert Back Office:

https://secure.lyra.com/portal/

2. Open the **Settings** > **Notification rules** menu.
3. Select **Advanced notification**.
4. Select the type of **E-mail sent to the merchant** notification.
5. Click **Next**.
6. Select the trigger event for **Invalid payment form**.
7. In the **General settings**, fill in the fields:
   - **Rule reference**
   - **E-mail address to notify**
8. Click **Create**.

A description of the error codes with their possible causes is available on our website:

https://docs.lyra.com/fr/collect/error-code/error-00.html

Other messages may appear during the payment process.

Here is a list of the most frequent messages:

Message	Description
Your payment request has been declined by your financial institution.	• The buyer's bank has rejected the authorization or information request. • The risk assessment rules have triggered the rejection of the transaction.
Your registration request has been declined by your financial institution.	• The buyer's bank has rejected the authorization or information request. • The risk assessment rules have triggered the rejection of the transaction.
This payment order is expired. Please contact your shop.	The buyer clicked on the payment link after the payment order expiration date.
This payment order has already been paid.	The buyer clicked on the payment link one more time after having already made the payment.
An error occurred during the payment request, the merchant website has been informed of the impossibility to finalize the transaction.	The payment form has been rejected. The shop administrator has received an e-mail with the details about the origin of the error.
The transaction has already been made.	The merchant website sends a transaction identifier that has already been used for another transaction (accepted or rejected). The transaction identifier must be unique within the same day (00:00:00 at 23:59:59 UTC).
Sorry, you have been disconnected due to a long period of inactivity.	• The buyer attempts to validate the card number while the payment session is expired. The session is open for about 10 minutes. • The merchant website sends a transaction identifier that has already been used but that did not result in a transaction (e.g. abandoned payment). The transaction identifier must be unique within the same day (00:00:00 at 23:59:59 UTC).

Message	Description
Cookies are blocked by your browser. Make sure you authorize them before retrying the operation.	The buyer has disabled cookies in his or her browser. Cookies are necessary for the payment to be processed correctly.

14.3. Managing timeouts

Payment session

A "payment session" is the time spent by a buyer on the payment page.

The payment session begins as soon as the payment gateway receives the payment form.

The delay of payment session is 10 minutes (except for certain payment methods).

This delay is:

- **sufficient** to enable each buyer to make his or her payment
- **fixed in time**: it is not reset after every action of the user
- **non-modifiable**: it is fixed by the payment gateway due to technical constraints

After this delay, the payment session times out and the session data is purged.

Expiration of the payment session

In some cases the payment session will expire while the buyer has not completed the payment.

Most frequent cases:

1. For example, once redirected to the payment page, the buyer realizes that it is time to go to lunch.

   An hour later, the buyer decides to continue his or her payment and clicks on the logo corresponding to his or her payment method.

   The buyer's payment session has already expired, the payment gateway displays an error message indicating that the buyer was disconnected due to an extended period of inactivity.

   The buyer then has the opportunity to click a button to return to the merchant website.

   The return to the shop is done via the URL specified by the merchant:

   - in the *vads_url_return* field transmitted in the payment form,
   - in the "Return URL to the merchant website" field in the buyer's Expert Back Office, no *vads_url_return* field is transmitted in his or her payment form.

2. Once redirected to the payment page, the buyer closes the browser (by mistake or because he or she no longer wants to make the payment).

Notification in case of session expiration

It is possible to notify the merchant website in case of expiration of the payment session.

To do this, the merchant must set up and activate the ***notification on cancellation*** rule (see chapter Setting up notifications).

15. IMPLEMENTING THE IPN

To process the payment result, the merchant website must have a separate page with a script.

This page will be automatically called after each payment (whether it has been accepted or declined): the parameters linked to the payment result are sent in POST mode by the payment gateway.

This server to server call, made at the same time as the payment, must be as short as possible and its length depends only on your processing time.

The script must include at least the following steps:

- Retrieve the field list sent with the POST response
- Compute the signature taking into account the received data
- Compare the computed signature with the received signature
- Analyzing the nature of the notification
- Retrieve the payment result

The script may check the order status (or any information of your choice) to see if it has not already been updated.

Once these steps are completed, the script can update the database (new order status, stock update, registration of payment information, etc.).

In order to facilitate support and diagnosis by the merchant in the event of a notification error, we recommend to write messages that will allow you to know at which stage of processing the error occurred.

The gateway reads and stores the first 256 bytes of the HTTP response.

You can write messages throughout the processing. Here are some examples of messages that you can use:

Message	Use case
Data received	Message to display when retrieving data. Allows to confirm that the notification has been received by the merchant website.
POST is empty	Message to display when retrieving data. Allows to bring out a possible redirection that would have caused the parameters posted by the payment gateway to be lost.
An error occurred while computing the signature.	Message to be displayed when the verification of the response signature has failed.
Order successfully updated.	Message to be displayed at the end of the file once your processing has been successfully completed.
An error occurred while updating the order.	Message to be displayed at the end of the file if an error occurred during your processing.

15.1. Preparing your environment

The notifications of Instant Payment Notification URL call type are very important as they represent the only reliable way for the merchant website to obtain the payment result.

It is therefore necessary to make sure the notifications function properly.

Here are some guidelines:

- Notifications are sent from an IP address in the **194.50.38.0/24**, default port 443 (HTTPS) range in Test and Production mode. This range of IP addresses must be authorized in the event of a restriction on the part of the merchant site.
- The merchant has to make sure that this URL is available via the payment gateway without redirection. Using redirection leads to losing data presented in POST.

  This is the case if there is a configuration on your devices or on the side of your host that redirects the URLs of "https://**www.**example.com" type to "https://example.com" or "**http**://example.com" to "**https**://example.com".

- HTML must not be visible on the page. Access to images or CSS slows down the exchange between the payment gateway and the merchant website.
- Avoid integrating time-consuming tasks, such as PDF invoice generation or sending e-mails in your script.

  The processing time has a direct influence on the time it takes to display the payment summary page.

  The longer the processing of the notification, the greater the delay for displaying the ticket to the buyer.

  The buyer may be tempted to close the browser and place another order.

  After 10 seconds, the payment gateway consider that the call has failed (timeout).

- Make sure the IPN processing time is as short as possible (10s maximum). This will allow you to:
  - provide a seamless user experience at checkout and increase the chances of payment conversion;
  - ensure reliable synchronization of the transaction status in your IS, so that it matches the payment result.
- If your page is only accessible in https, test your URL on the website ofQualys SSL Labs(https://www.ssllabs.com/ssltest/ and change your configuration if necessary in order to obtain an A score.

  Your SSL certificate must be signed by a certification authority known and recognized on the market.

- Make sure that you use the latest version of the TLS protocol in order to maintain a high level of security.

15.2. Retrieving data returned in the response

The data returned in the response depends on the parameters sent in the payment request, the payment type, the settings of your shop and the notification format.

The data is always sent by the payment gateway using the **POST** method.

The first step consists in retrieving the contents received via the POST method.

Examples:

- In PHP, data is stored in the super global variable **$_POST**,
- In ASP.NET (C#), you must use the **Form** property of the **HttpRequest** class,
- In Java, you must use the **getParameter** method of the **HttpServletRequest** interface.

The response consists of a field list. Each field contains a response value. The field list can be updated.

The script will have to create a loop to retrieve all the transmitted fields.

It is recommended to test the presence of the **vads_hash** field, which is only present during a notification.

```
if (empty ($_POST)){
 echo 'POST is empty';

}else{
 echo 'Data Received ';
 if (isset($_POST['vads_hash'])){

  echo 'Form API notification detected';
  //Signature computation
  //Signature verification
  //Order Update
 }
}
```


15.3. Computing the IPN signature

The signature is computed by following the same logic as for creating the payment request .

The data submitted by the payment gateway is encoded in UTF-8. Any alteration of received data will result in signature computation error.

You must compute the signature with the fields received in the notification and not the ones that you transmitted in the payment request.

1. Take all the fields whose name starts with **vads_**.
2. Sort these fields alphabetically.
3. Concatenate the values of these fields separating them with the "**+**" character.
4. Concatenate the result with the test or production key separating them with a "**+**".
5. According to the signature algorithm defined in your shop configuration:
   a. If your shop is configured to use "SHA-1", apply the **SHA-1** hash function to the chain obtained during the previous step **Deprecated.**
   b. If your shop is configured to use "HMAC-SHA-256", compute and encode in Base64 format the message signature using the **HMAC-SHA-256** algorithm with the following parameters:
      - the SHA-256 hash function,
      - the test or production key (depending on the value of the vads_ctx_mode field) as a shared key,
      - the result of the previous step as the message to authenticate.

Examples in PHP:

```
function getSignature ($params,$key)
{
    /**
     *Function that computes the signature.
     * $params: table containing the fields received in the IPN.
     * $key: TEST or PRODUCTION key
     */
    //Initialization of the variable that will contain the string to encrypt
    $signature_content = "";

    //Sorting fields alphabetically
    ksort($params);
    foreach($params as $name=>$value){

        //Recovery of vads_ fields
        if (substr($name,0,5)=='vads_'){

            //Concatenation with "+"
            $signature_content .= $value."+";
         }
    }
    //Adding the key at the end
    $signature_content .= $key;

    //Encoding base64 encoded chain with HMAC-SHA-256 algorithm
    $sign = base64_encode(hash_hmac('sha256',$signature_content, $key, true));
    return $sign;
 }
```


15.4. Comparing signatures

To ensure the integrity of the response, you must compare the signature contained in the IPN with the value computed in the previous step.

You should not compare the signature of the IPN with the signature that you transmitted in your payment request.

If the signatures match

- You may consider the response as safe and proceed with the analysis.
- Otherwise, the script will have to raise an exception and notify the merchant about the anomaly.

Example in PHP:

```
if ($_POST['signature'] == $sign){

    //Processing data

}else{
    throw new Exception('An error occurred while computing the signature');
}
```

The signatures may not match in case of:

- an implementation error (error in your calculation, problem with UTF-8 encoding, etc.),
- an error in the key value or in the **vads_ctx_mode** field (frequent issue when shifting to production mode),
- a data corruption attempt.

15.5. Analyzing the nature of the notification

In a notification, field **vads_url_check_src** allows to differentiate the notifications based on their triggering event:

- creation of a transaction
- new notification sent by the merchant via the Expert Back Office.

It specifies the applied notification rule.

Values associated with the vads_url_check_src field:

Value	Description
PAY	Payment creation by form. Value is sent in the following cases: • registration request for a mandate or a token (REGISTER) • registration request for a mandate or a token when subscribing to a recurring payment (REGISTER_SUBSCRIBE) • immediate payment (or first installment payment of a recurring payment) • payment deferred for less than 7 days • payment abandoned or canceled by the buyer Only if the merchant has configured the “Instant Payment Notification URL on cancellation” rule.
BO	Execution of the notification URL from the Expert Back Office. (Right-click a transaction > Send the Instant Payment Notification).
BATCH_AUTO	Value sent as part of an authorization request for a payment that was awaiting authorization. Not applicable for one-off direct debit on SEPA.
BATCH	Value sent in case of an update of a transaction status after its synchronization on the acquirer side. Only if the merchant has configured the “Instant Payment Notification URL on batch change” rule.
DCF	Value sent following a transaction originating from the data collection form.
MERCH_BO	Value sent following an operation made via the Expert Back Office if the merchant has configured the following notification rule: “Instant Payment Notification URL on an operation coming from the Back Office”.
PAYMENT_ORDER	Value sent following a transaction originating from a payment order (e-mail, payment URL or SMS).
REC	Value is sent only for recurring payments if the merchant has configured the Instant Payment Notification “URL when creating recurring payments rule”. Not applicable for SEPA one-off direct debit.
RETRY	Automatic retry of the IPN.

After checking its value, the script can process differently depending on the nature of the notification.

For example:

If **vads_url_check_src** is set to **PAY** or **BATCH_AUTO** the script updates the order status, ...

If **vads_url_check_src** is valued at **REC** the script will retrieve the recurring payment reference and will increment the number of the expired installment payments in case the payment has been accepted, etc.

15.6. Processing the response data

Here is an example of analysis to guide you through processing the response data.

1. Identify the mode (TEST or PRODUCTION) that was used for creating the transaction by analyzing the value of the **vads_ctx_mode** field.

2. Identify the order by retrieving the value of **vads_order_id** if you have included it in the payment form.
   Make sure that the order status has not been updated yet.

3. Retrieve the payment result transmitted in the field **vads_trans_status**.
   Its value allows you to define the order status.

Value	Description
ABANDONED	**Abandoned** Payment abandoned by the buyer The transaction has not been created, and **therefore cannot be viewed in the Expert Back Office**.
ACCEPTED	**Accepted**. Status of a VERIFICATION type transaction for which the authorization request or information request has been successfully completed. This status cannot evolve. Transactions with the **Accepted** status will never be captured.
AUTHORISED	**Waiting for capture** The transaction has been accepted and will be automatically captured at the bank on the expected date.
AUTHORISED_TO_VALIDATE	**To be validated** The transaction, created with manual validation, is authorized. The merchant must manually validate the transaction in order for it to be captured. The transaction can be validated as long as the expiration date of the authorization request has not passed. If the authorization validity period has been passed, the payment takes **Expired** status. This status is final.
CANCELLED	**Cancelled** The transaction has been canceled by the Merchant.
CAPTURED	**Captured** The transaction has been captured by the bank.
CAPTURE_FAILED	**Capture failed** Contact the technical support.
EXPIRED	**Expired** This status appears in the lifecycle of a payment with deferred capture. The expiry date of the authorization request has passed and the merchant has not validated the transaction. The account of the cardholder will therefore not be debited.
REFUSED	**Refused**

Value	Description
	The transaction is refused.
SUSPENDED	**Suspended** The capture of the transaction is temporarily blocked by the acquirer (AMEX GLOBAL or SECURE TRADING). Once the transaction has been correctly captured, its status changes to **CAPTURED**.
UNDER_VERIFICATION	**Control in progress** Waiting for the response from the acquirer. This status is temporary. For CB or PPRO transactions, this status indicates that a refund has been requested. Verification is being made in order to validate the refund. A notification will be sent to the merchant website to inform the Merchant of the status change. Requires the activation of the Instant Payment Notification URL on batch change notification rule.
WAITING_AUTHORISATION	**Waiting for authorization** The capture delay in the bank exceeds the authorization validity period.
WAITING_AUTHORISATION_TO_VALIDATE	**To be validated and authorized** The capture delay in the bank exceeds the authorization validity period. A EUR 1 (or information request about the CB network if the acquirer supports it) authorization has been accepted. The merchant must manually validate the transaction for the authorization request and the capture to occur.

4. Analyze the field**vads_occurrence_type**to determine if it is a single payment or a payment that is part of a series (recurring payment or payment in installments).

Value	Description
UNITAIRE	Single payment (immediate payment).
RECURRENT_INITIAL	First payment of a series.
RECURRENT_INTERMEDIAIRE	Nth payment of a series.
RECURRENT_FINAL	Last payment of a series.

5. Analyze the field **vads_payment_config** to determine whether it is a **payment in installments**.

Field name	Value for an immediate payment	Value for a payment in installments
vads_payment_config	SINGLE	MULTI (the exact syntax isMULTI:first=X;count=Y;period=Z)

For a payment in installments, identify the installment number by retrieving the value of the field **vads_sequence_number**.

Warning: with the application os Soft Decline, the **vads_sequence_number** field no longer allows to easily identify the first installment of a payment in installments. Since the sequence number of the first installment can be different from 1, the sequence number of the second installment will not necessarily be 2.

6. Retrieve the value of the field **vads_trans_date** to identify the payment date.

7. Analyze the **vads_payment_option_code** field to determine whether it is an installment payment:

Value	Description
1	Payment in 1 installment
2	Payment in 2 installments
3	Payment in 3 installments
n	Payment in n installments

8. Retrieve the value of the field **vads_capture_delay** Delay (in days) before the payment is captured.
It will allow you to identify whether the payment is an immediate or a deferred payment.

9. Retrieve the used amount and currency. To do this, retrieve the values of the following fields:

Field name	Description
vads_amount	Payment amount in the smallest currency unit.
vads_currency	Code of the currency used for the payment.
vads_change_rate	Exchange rate used for calculating the effective payment amount (see vads_effective_amount).
vads_effective_amount	Payment amount in the currency used for the capture in the bank.
vads_effective_currency	Currency used for the capture in the bank.

10. Retrieve the value of the **vads_auth_result** field to know the result of the authorization request.
The complete list of returned codes can be viewed in the data dictionary.

Here is a list of frequently returned codes that can help you understand the reason of the rejection:

Value	Description
03	**Invalid acceptor** This code is sent by the card issuer. It refers to a configuration problem on authorization servers. (e.g. closed contract, incorrect MCC declared, etc.). **To find out the specific reason of the rejection, the buyer must contact his or her bank.**
05	**Do not honor** This code is sent by the card issuer. This code is used in the following cases: • Invalid expiry date • Invalid CVV • Exceeded credit limit • Insufficient funds (etc.) **To find out the specific reason of the rejection, the buyer must contact his or her bank.**
51	**Insufficient balance or exceeded credit limit** This code is sent by the card issuer. This code appears if the funds on the buyer's account are insufficient for making the purchase. **To find out the specific reason of the rejection, the buyer must contact his or her bank.**
56	**Card absent from the file** This code is sent by the card issuer. The entered card number is incorrect or the card number + expiration date combination does not exist.
57	**Transaction not allowed for this cardholder**

Value	Description
	This code is sent by the card issuer. This code is used in the following cases: • The buyer attempts to make an online payment with a cash withdrawal card. • The authorized payment limit is exceeded. **To find out the specific reason of the rejection, the buyer must contact his or her bank.**
59	**Suspected fraud** This code is sent by the card issuer. This code appears when an incorrect CVV code or expiration date has been entered several times. **To find out the specific reason of the rejection, the buyer must contact his or her bank.**
60	**The acceptor of the card must contact the acquirer** This code is sent by the card issuer. It refers to a configuration problem on authorization servers. It is used when the merchant ID does not correspond to the used sales channel. (e.g.: an e-commerce transaction with a distant sale contract with manual entry of contract data). **Contact the customer service to resolve the problem.**
81	**Unsecured payment is not accepted by the issuer** This code is sent by the card issuer. After receiving this code, the payment gateway automatically makes a new payment attempt with 3D Secure authentication, when possible.

11. Retrieve the cardholder authentication result. To do this:

a. Retrieve the value of the field **vads_threeds_enrolled** to determine the status of the card enrollment.

Value	Description
Empty	Incomplete 3DS authentication process (3DS disabled in the request, the merchant is not enrolled or the payment method is not eligible for 3DS).
Y	Authentication available, cardholder enrolled.
N	Cardholder not enrolled.
U	Impossible to identify the cardholder or authentication is not available for the card (e.g. commercial or prepaid cards).

b. Retrieve the result of cardholder authentication by retrieving the value of the field **vads_threeds_status**.

Value	Description
Empty	Incomplete 3DS authentication (3DS disabled in the request, the cardholder is not enrolled or the payment method is not eligible for 3DS).
Y	Cardholder authentication success.
N	Cardholder authentication error.
U	Authentication impossible.
A	Authentication attempted but not completed.

12. Retrieve the result of fraud checks by identifying the value of the field **vads_risk_control**. This field is sent only if the merchant has:

- Subscribed to the "**Risk management**" service.
- Enabled at least one verification process in the Expert Back Office (**Settings** > **Risk management** menu).

It is populated with the list of values separated by ";" with the following syntax: **vads_risk_control = control1=result1;control2=result2**

The possible values for **control** are:

Value	Description
CARD_FRAUD	Verifies whether the cardholder's card number is on the card greylist.
SUSPECT_COUNTRY	Checks whether the issuing country of the buyer's card is on the list of forbidden countries.
IP_FRAUD	Verifies whether the cardholder's IP address is on the IP greylist.
CREDIT_LIMIT	Checks the purchase frequency and amounts for the same card number, or the maximum amount of an order.
BIN_FRAUD	Checks whether the BIN code of the card is on the BIN code greylist.
ECB	Checks whether the buyer's card is of "e-carte bleue" type.
COMMERCIAL_CARD	Checks whether the buyer's card is a commercial card.
SYSTEMATIC_AUTO	Checks whether the buyer's card is a MAESTRO or VISA ELECTRON card.
INCONSISTENT_COUNTRIES	Checks whether the country of the IP address, the country of the payment card and the buyer's country of residence match.
NON_WARRANTY_PAYMENT	Liability shift.
SUSPECT_IP_COUNTRY	Checks whether the buyer's country, identified by their IP address, is on the list of forbidden countries.

The possible values for **result** are:

Value	Description
OK	OK.
WARNING	Informational control failed.
ERROR	Blocking control failed.

13. Retrieve the card type used for the payment.

Two scenarios are possible:

- For a payment processed with **only one card**. The fields to process are:

Field name	Description
vads_acquirer_network	Acquirer network code
vads_card_brand	Brand of the card used for the payment, e.g.: CB, VISA, VISA_ELECTRON, MASTERCARD, MAESTRO, VPAY
vads_card_number	Card number used for the payment.
vads_expiry_month	Expiry month between 1 and 12 (e.g.: 3 for March, 10 for October).
vads_expiry_year	Expiry year in 4 digits (e.g.: 2023).
vads_bank_code	Code of the issuing bank
vads_bank_label	Name of the issuing bank

Field name	Description
vads_bank_product	Product code of the card
vads_card_country	Code of the issuing country (Alpha ISO 3166-2, e.g.:"FR"for France,"PF"for French Polynesia,"NC"for New Caledonia, "US"for the United States).

- For a **split payment** (i.e. a transaction using several payment methods), the following fields must be processed:

Field name	Value	Description
vads_card_brand	MULTI	Several types of payment cards are used for the payment.
vads_payment_seq	In Json format, see details below.	Details of performed transactions.

The field **vads_payment_seq** (json format) describes the split payment sequence. It contains the following elements:

1. "trans_id" : transaction identifier used for the entire payment sequence.
2. "transaction" : transaction table of the sequence. It contains the following elements:

Field name	Description
amount	Amount of the payment sequence.
operation_type	Debit transaction.
auth_number	Authorization number. Will not be returned if not applicable to the used payment method. Example: 949478
auth_result	Return code of the authorization request.
capture_delay	Delay before the capture (in days). • For a payment by card, this parameter is the requested capture date (ISO 8601 format). If not sent in the payment form, the value defined in the Expert Back Office will be used.
card_brand	Used payment method. For a payment by card (e.g. CB or Visa or MasterCard co-branded CB cards), this parameter is set to **"CB"**. See the Payment Gateway Implementation Guide available in our online documentation archive to see the complete list of card types.
card_number	Payment method number.
expiry_month	Expiry month of the payment method.
expiry_year	Expiry year of the payment method.
payment_certificate	Payment certificate.
contract_used	Contract used for the payment.
identifier	Unique identifier (token) associated with a payment method.
identifier_status	Only present if the requested action is token creation or update. Possible values:

<table>
<tr><th>Field name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>CREATED</td><td>The authorization request has been accepted.
The token (or UMR for SEPA payment) has been successfully created.</td></tr>
<tr><td>NOT_CREATED</td><td>The authorization request has been declined.
The token (or UMR for SEPA payment) has not been created, and therefore cannot be viewed in the Expert Back Office.</td></tr>
<tr><td>UPDATED</td><td>The token (or UMR for SEPA payment) has been successfully updated.</td></tr>
<tr><td>NOT_UPDATED</td><td>The token (or UMR for SEPA payment) has not been updated.</td></tr>
<tr><td>ABANDONED</td><td>The action has been abandoned by the buyer (debtor).
The token (or UMR for SEPA payment) has not been created, and therefore cannot be viewed in the Expert Back Office.</td></tr>
</table>
</td></tr>
<tr><td>presentation_date</td><td>For a payments by card, this parameter is the requested capture date (ISO 8601 format).</td></tr>
<tr><td>trans_id</td><td>Transaction number.</td></tr>
<tr><td>ext_trans_id</td><td>This field is not sent for credit card payments.</td></tr>
<tr><td>trans_uuid</td><td>Unique reference generated by the payment gateway after the creation of a payment transaction.
Guarantees that each transaction is unique.</td></tr>
<tr><td>extra_result</td><td>Numeric code of the risk assessment result.
<table>
<tr><th>Code</th><th>Description</th></tr>
<tr><td>Empty</td><td>No verification completed.</td></tr>
<tr><td>00</td><td>All the verification processes have been successfully completed.</td></tr>
<tr><td>02</td><td>Credit card velocity exceeded.</td></tr>
<tr><td>03</td><td>The card is on the Merchant’s greylist.</td></tr>
<tr><td>04</td><td>The country of origin of the card is on the Merchant’s greylist.</td></tr>
<tr><td>05</td><td>The IP address is on the Merchant’s greylist.</td></tr>
<tr><td>06</td><td>The BIN code is on the Merchant’s greylist.</td></tr>
<tr><td>07</td><td>Detection of an e-carte bleue.</td></tr>
<tr><td>08</td><td>Detection of a national commercial card.</td></tr>
<tr><td>09</td><td>Detection of a foreign commercial card.</td></tr>
<tr><td>14</td><td>Detection of a card that requires systematic authorization.</td></tr>
<tr><td>20</td><td>Relevance verification: countries do not match (country IP address, card country, buyer's country).</td></tr>
<tr><td>30</td><td>The country of the this IP address is on the greylist.</td></tr>
<tr><td>99</td><td>Technical issue encountered by the server during a local verification process.</td></tr>
</table>
</td></tr>
</table>

Field name	Description
sequence_number	Sequence number.
trans_status	Status of the transaction.

Canceled transactions are also displayed in the table.

14. Record the used payment wallet type by retrieving the value of the field **vads_wallet**.

    The field **vads_wallet** Is present only when a wallet was used for the payment.

Field values	Wallet type
APPLE_PAY	Apple Pay
GOOGLEPAY	Google pay

15. Store the value of the **vads_trans_uuid** field. It will allow you to assign unique identification to the transaction if you use the Web Service APIs.

16. Retrieve all the order, buyer and shipping details.

    These details will be provided in the response only of they have been transmitted in the payment form.

    Their values are identical to the ones submitted in the form.

17. Proceed to order update.

15.7. Running tests and troubleshooting

In order to test the notifications, follow the steps below:

1. Make a payment (in TEST mode or in PRODUCTION mode).
2. Once the payment is complete, look for the transaction in your Back Office (**Management** > **Transactions** or **TEST Transactions** menu if you made the payment in TEST mode).
3. Double-click the transaction to view the **transaction details**.
4. In the transaction details, search for the section entitled **Technical data**.
5. Check the status of the Instant Payment Notification URL:

The list of possible statuses is provided below:

Status	Description
N/A	The transaction did not result in a notification or no notification rules have been enabled.
Undefined URL	An event has triggered the end of payment notification rule but the URL is not configured.
Call in progress	The notification is in progress. This status is temporary.
Sent	The notification has been successfully sent and a remote device returned an HTTP 200, 201, 202, 203, 204, 205 or 206 response status code.
Sent (permanent redirection)	The merchant website has returned an HTTP 301 or 308 response status code with a new URL to contact. A new call in POST mode has been made to the new URL.
Sent (temporary redirection)	The merchant website has returned an HTTP 302 or 307 response status code with a new URL to contact. A new call in POST mode has been made to the new URL.
Sent (redirection to another page)	The merchant website has returned an HTTP 303 response status code with a new URL to contact. A new call in GET mode has been made to the new URL.
Failed	Generic error different from the codes described below.
Server unavailable	The notification has lasted more than 35s.
SSL handshake failure	Your server is incorrectly configured. Run a test on the Qualys website (https://www.ssllabs.com/ssltest/) and correct the errors.
Connection interrupted	Communication error.
Connection refused	Communication error.
Server error 300	Case of redirection not supported by the gateway.
Server error 304	Case of redirection not supported by the gateway.
Server error 305	Case of redirection not supported by the gateway.
Server error 400	The merchant website returned a HTTP 400 Bad Request code.

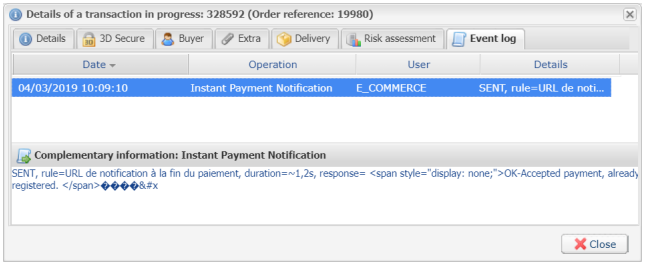
Status	Description
Server error 401	The merchant website returned a HTTP 401 Unauthorized code. Make sure that the resource is not protected by an .htaccess file.
Server error 402	The merchant website returned a HTTP 402 Payment Required code.
Server error 403	The merchant website returned a HTTP 403 Forbidden code. Make sure that the resource is not protected by an .htaccess file.
Server error 404	The merchant website returned a HTTP 404 Not Found code. Make sure that the URL is correctly specified in the rule configuration. Make sure that the file is present on your server.
Server error 405	The merchant website returned a HTTP 405 Method Not allowed code.
Server error 406	The merchant website returned a HTTP 406 Not Acceptable code.
Server error 407	The merchant website returned a HTTP 407 Proxy Authentication Required code.
Server error 408	The merchant website returned a HTTP 408 Request Time-out code.
Server error 409	The merchant website returned a HTTP 409 Conflict code.
Server error 410	The merchant website returned a HTTP 410 Gone code.
Server error 411	The merchant website returned a HTTP 411 Length Required code.
Server error 412	The merchant website returned a HTTP 412 Precondition Failed code.
Server error 413	The merchant website returned a HTTP 413 Request Entity Too Large code.
Server error 414	The merchant website returned a HTTP 414 Request-URI Too long code.
Server error 415	The merchant website returned a HTTP 415 Unsupported Media Type code.
Server error 416	The merchant website returned a HTTP 416 Requested range unsatisfiable code.
Server error 417	The merchant website returned a HTTP 417 Expectation failed code.
Server error 419	The merchant website returned a HTTP 419 Authentication Timeout code.
Server error 421	The merchant website returned a HTTP 421 Misdirected Request code.
Server error 422	The merchant website returned a HTTP 422 Unprocessable Entity code.
Server error 423	The merchant website returned a HTTP 423 Locked code.
Server error 424	The merchant website returned a HTTP 424 Failed Dependency code.
Server error 425	The merchant website returned a HTTP 425 Too Early code.
Server error 426	The merchant website returned a HTTP 426 Upgrade Required code.
Server error 429	The merchant website returned a HTTP 431 Request Header Fields Too Large code.
Server error 431	The merchant website returned a HTTP 415 Unsupported Media Type code.
Server error 451	The merchant website returned a HTTP 451 Unavailable For Legal Reasons code.
Server error 500	The merchant website returned a HTTP 500 Internal Server Error code.

Status	Description
	An application error has occurred on the level of the server hosting your shop. See the logs of your HTTP server (usually apache). The issue can only be corrected by performing an action on your server.
Server error 501	The merchant website returned a HTTP 501 Not Implemented code.
Server error 502	The merchant website returned a HTTP 502 Bad Gateway / Proxy Error code.
Server error 503	The merchant website returned a HTTP 503 Service Unavailable code.
Server error 504	The merchant website returned a HTTP 504 Gateway Time-out code. The merchant server has not accepted the call within the time limit of 10s.
Server error 505	The merchant website returned a HTTP 505 HTTP Version not supported code.

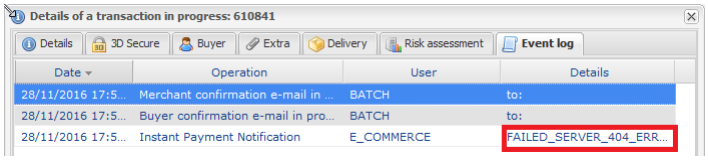
For more information on a notification, click the link **Display the details** or click the **Event log** tab and search for the line **Notification URL call**.

In order to help the merchant identify the source of the error, the gateway systematically analyses the 512 first characters returned by the merchant website and displays them in the **Details** column.

- Example of a successfully processed notification:



- Example of a failed notification:



If the payment gateway is unable to access the URL of your page, an e-mail alert will be sent to the shop administrator.

It contains:

- The HTTP code of the encountered error
- Parts of error analysis
- Its consequences
- Instructions to follow via the Expert Back Office for resending the request to the URL specified in step 4

16. RETURNING TO THE SHOP

By default, when the buyer returns to the merchant website, no parameters will be transmitted by their browser.

However, if the **vads_return_mode** field has been transmitted in the payment form (see chapter **Managing the return to the merchant website**) it will be possible to retrieve the data:

- either via GET, the data is presented in the URL as follows: ?field1=value1&field2=value2
- or via POST: the data is sent in a POST form

The data transmitted to the browser is the same as for notifications (IPN).

Only the **vads_url_check_src** and **vads_hash** fields will be sent in the instant notification.

To analyze this data, see chapter **Analyzing the payment result**.

The return to the shop should only allow you to display visual context to the buyer.

17. OBTAINING HELP

Looking for help? See our FAQ:

https://support.lyra.com/hc/fr

For any technical inquiries or if you need any help, contact technical support.

To help us process your demands, please have your customer code ready (e.g.: **CLXXXXX**, **MKXXXXX** or **AGXXXXX**).

This information is available in the Merchant Back Office top of menu.

