Lyra Documentation

Generated on: 2025-01-06 03:05:39,441

Table of Contents

24. # Presentation of methods
25. # Themes
26. # Use cases
27. # Example file : formToken.php
28. # Example files: ipn.php and paid.php
29. # Step 1: Set the notification URL at the end of the payment
30. # Sample file: config.php
31. # KR.setFormConfig()
32. # KR.field.focus()
33. # KR.getPaymentMethods()
34. # KR.openPaymentMethod()
35. # KR.setBrand()
36. # KR.userPaymentMethodsOrder()
37. # KR.validateForm()
38. # Step 4: Display the payment form
39. # Show / Hide
40. # Personalizing the "Pay" button
41. # Customization of the layout for the embedded form
42. # Themes
43. # Step 5: Analyze the payment result
44. # Sample files: embedded.php and popin.php
45. # Presentation of methods
46. # Embedded mode
47. # Pop-in mode

48. # JavaScript client reference

49. # Step 6: Shift in Production

50. # Adding custom fields to your form

51. # Managing errors (JS client)

52. # New payment attempt (retry)

53. # Code samples

54. # Getting started: single payment

55. ## 404 Error...

56. # require.js

SmartForm presentation

Source: https://docs.lyra.com/en/rest/V4.0/javascript/redirection/presentation.html

Le formulaire embarqué (smartForm) vous permet d'intégrer un parcours de paiement fluide et de présenter plusieurs moyens de paiement.

The buyer will complete the payment process without leaving the merchant site.

Major evolution of the embedded form (cards)

The smartForm offers the advantage of enhancing the embedded form (cards) by providing **new features**. If you would like to migrate, click [here](/en/rest/V4.0/javascript/redirection/migration.html)

List of compatible payment methods

See [the list of compatible payment methods](/en/rest/V4.0/javascript/redirection/compatible_payment_method.html).

Use cases

| Description | List mode (default) | |---|---| | This mode displays the list of the card payment and the compatible payment method buttons. |

Check this link to get

additional information::[list mode](/en/rest/V4.0/javascript/redirection/use_case_list.html)| Description | Pop-in mode | |---|---| | This mode displays a single button that opens a pop-in containing the card payment button and the compatible payment methods. |

Check this link to get

additional information::[pop-in mode](/en/rest/V4.0/javascript/redirection/use_case_popin.html)| Description | List mode with embedded card | |---|---| | This mode displays the embedded fields for card payments and compatible payment methods. |

Check this link to get

additional information::[List mode with embedded card](/en/rest/V4.0/javascript/redirection/use_case_embedded.html)| Description | List mode (default) | |---|---| | Here is an illustration to highlight Paypal. It displays the card payment fields and a button for the compatible payment method of your choice. |

Check this link to get

additional information::[highlight a payment method.](/en/rest/V4.0/javascript/redirection/use_case_one_payment_method.html)| Description | Pop-in mode | |---|---| | Here is an illustration to highlight Paypal It displays a button for card payment and a button for the compatible payment method of your choice. |

Check this link to get

additional information::[highlight a payment method.](/en/rest/V4.0/javascript/redirection/use_case_one_payment_method.html)| Description | List mode with embedded card | |---|---| | Here is an illustration to highlight Paypal. It displays the card payment fields and a button for the compatible payment method of your choice. |

Check this link to get

additional information::[highlight a payment method.](/en/rest/V4.0/javascript/redirection/use_case_one_payment_method.html)| Description | List mode with embedded card | |---|---| | Here is an illustration to highlight several payment methods: Paypal and Apple Pay. It displays the card payment fields, with a button for each compatible payment method highlighted. |

Check this link to get

additional information::[highlighting several payment methods:](/en/rest/V4.0/javascript/redirection/use_case_many_payment_method.html)| Description | Pop-in mode | |---|---| | Here is an illustration to highlight several payment methods: Paypal and Apple Pay. It displays a button for card payments and a button for each of the highlighted compatible payment methods. |

Check this link to get

additional information::[highlighting several payment methods:](/en/rest/V4.0/javascript/redirection/use_case_many_payment_method.html)| Description | List mode with embedded card | |---|---| | Here is an illustration to highlight several payment methods It displays the card payment fields, with a button for each compatible payment method highlighted. |

Check this link to get

additional information::[highlighting several payment methods:](/en/rest/V4.0/javascript/redirection/use_case_many_payment_method.html)| Hide card payment logos | | |---|---| <body> <div class="kr-smart-form" kr-card-form-expanded kr-no-card-logo-header kr-form-token="[GENERATED FORMTOKEN]"> (...) </body> |

Check this link to get

additional information::[Hide card payment logos](/en/rest/V4.0/javascript/redirection/custom_without_logo.html)| Personalize the page layout | | |---|---| <style type="text/css"> (...) /*to use the CSS Flexbox (Flexible Box)*/ .kr-smart-form

.kr-embedded .flex-container { flex-direction: row !important; display: flex; } </style> <body> <div class="kr-smart-form" kr-card-form-expanded kr-form-token="[GENERATED FORMTOKEN]"> (...)   </body> |

Check this link to get

additional information::[Advanced personalization](/en/rest/V4.0/javascript/redirection/custom.html)You also have the possibility to:

Technical implementation

Si vous avez déjà implémenté le formulaire embarqué (cartes), vous trouverez les informations pour migrer vers le formulaire embarqué (smartForm) :

The different stages of integration are :

Prerequisites**Authenticate****Create**the`formToken`

.**Display**the payment form.**Analyze**the payment result.- Switch to **Production**.

To get more information, go to [Payment Form Integration](/en/rest/V4.0/javascript/guide/presentation.html).

Integration of the payment form

Source: https://docs.lyra.com/en/rest/V4.0/javascript/guide/presentation.html

6 steps for integrating the payment form

1. Prerequisites

Set the Instant Payment Notification URL at the end of the payment.

2. Authenticate

Secure data during the payment.

3. Create the formToken

Generate the formToken to define the payment context (amount, currency, order number, buyer details, etc.),

4. Display the payment form

Discover the different display modes and the customization of the payment form.

5. Analyzing the payment result

Analyze the payment result via the IPN or via the return to the shop.

6. Go into production

Carry out the tests to go to Production mode.

List of compatible payment methods

Source: https://docs.lyra.com/en/rest/V4.0/javascript/redirection/compatible_payment_method.html

Payment by card is compatible with the smartForm.

| CB | Visa | mastercard | maestro | Visa Electron | American Express |

| Bimpli (ex Apetiz) | Ticket restaurant | Pass Restaurant | Chèque Déjeuner |

.

Apple Pay | Bizum | SEPA Credit Transfer | Alma in several installments | Payconiq | PayPal | PayPal Pay Later | UPI |

The list will be updated regularly.

For more information:

[Select payment methods](/en/rest/V4.0/javascript/redirection/custom_filter.html).- View [the table of payment methods](/en/rest/V4.0/javascript/redirection/reference_smartform.html#tableau-de-lattribut-kr-payment-method)with the attribute`kr-payment-method`

.

Advanced personalization

Source: https://docs.lyra.com/en/rest/V4.0/javascript/redirection/custom.html

Welcome to the customization area.

Personalize the payment buttons

Customize your payment buttons (label, color, icon, ...).

Customize the font

Customize the font of the embedded fields.

Hide card payment logos

Integrate this feature to customize the logos in the payment form.

Show/Hide CVV and PAN help icon

Integrate these functions to show or hide the CVV and the PAN help icon.

Change the display order of the payment methods

Set the order of display for payment methods via the

Group payment methods

Create a button grouping several payment methods.

Select payment methods

Source: https://docs.lyra.com/en/rest/V4.0/javascript/redirection/custom_filter.html

Par défaut, l'acheteur a le choix parmi tous les moyens de paiement éligibles (en fonction de la devise, du montant minimum ou maximum, des contraintes techniques) associés à la boutique.

Pour sélectionner les moyens de paiement proposés à l'acheteur, utilisez le champ `paymentMethods`

, lors la création du formToken (voir : [Etape 3 : Créer un formToken](/en/rest/V4.0/javascript/guide/formToken/presentation.html)).

Consultez [le tableau de l'attribut kr-payment-method](/en/rest/V4.0/javascript/redirection/reference_smartform.html#tableau-de-lattribut-kr-payment-method) pour connaître les valeurs des moyens de paiement.

Query examples

Offer **only** the following payment methods:

payment by cards and PayPal

/en/rest/V4.0/api/kb/authentication.html

https://api.lyra.com/api-payment/V4/Charge/CreatePayment

{ "amount": 10000, "currency": "EUR", "paymentMethods": ["PAYPAL", "PAYPAL_BNPL", "CARDS"], "customer": { "email": "sample@example.com" }, "orderId": "myOrderId-1234" }

payment by cards and Alma

/en/rest/V4.0/api/kb/authentication.html

https://api.lyra.com/api-payment/V4/Charge/CreatePayment

{ "amount": 10000, "currency": "EUR", "paymentMethods": ["ALMA_2X", "ALMA_3X", "ALMA_4X", "ALMA_10X", "ALMA_12X", "CARDS"], "customer": { "email": "sample@example.com" }, "orderId": "myOrderId-1234" }

payment by cards and Apple Pay

/en/rest/V4.0/api/kb/authentication.html

https://api.lyra.com/api-payment/V4/Charge/CreatePayment

{ "amount": 10000, "currency": "EUR", "paymentMethods": ["APPLE_PAY", "CARDS"], "customer": { "email": "sample@example.com" }, "orderId": "myOrderId-1234" }

To view the fields and their description, go to the playground: [Charge/CreatePayment](/en/rest/V4.0/api/playground/?ws=Charge%2FCreatePayment) (menu on the left).

Une fois le formToken généré, affichez le formulaire de paiement :

Changing the display order

Source: https://docs.lyra.com/en/rest/V4.0/javascript/redirection/custom_order.html

Cette action s'effectue depuis le

You need:

[Sign in to the keyword>bom](https://secure.lyra.com/portal/)- Go to **Settings > Shop**then select the shop for which the configuration must be changed. - Click on the **Association contrats**tab to display the MIDs available for the shop. - click on the contract to select it from the **Associated contracts**section. - use the arrows (at the bottom of the page) to determine the order priority. - Click Save to save the changes.

Also use the `KR.userPaymentMethodsOrder`

function to manage the display order. See: undefined.

assistance

Source: https://docs.lyra.com/en/rest/V4.0/javascript/redirection/custom_without_logo.html

Search

Categories

Tags

Europe (English)

. Code sample:

```
Code Example:
```

(...)

By default, the display of the card payment logos is automatic according to the contracts associated with the shop. By integrating this feature, the merchant can integrate their logos to customize the payment form.

Example without the logos | Example with the logos | |---|---|

Migrating from the embedded form (cards) to the smartForm

Source: https://docs.lyra.com/en/rest/V4.0/javascript/redirection/migration.html

Objective

- Migrating from the embedded form (cards) to the smartForm

I. Choose the display mode

This step allows you to define the display mode of the smartForm, depending on the implementation of your current payment form. Choose your implementation mode:

Embedded mode

In your code, replace:

`<div class="kr-embedded" kr-form-token="GENERATED TOKEN">`

By

`<div class="kr-smart-form" kr-form-token="GENERATED TOKEN">`

The form is displayed using the **list mode** , if the shop has compatible payment methods (link to [List of compatible payment
methods](/en/rest/V4.0/javascript/redirection/compatible_payment_method.html)).

If the store has **only** card payment, then the form is displayed directly with the embedded fields.

Pop-in mode

In your code, replace:

`<div class="kr-embedded" kr-popin kr-form-token="[GENERATED FORMTOKEN]">`

By

`<div class="kr-smart-form" kr-popin kr-form-token="[GENERATED FORMTOKEN]">`

When you click on the payment button, a pop-up window appears with all the compatible payment methods.

III. Payment Analysis - Instant Notification

The **material theme** is not supported by the smartForm, make sure to not use it.

If you use it, replace it in your code:

```
Code Example: src="https://static.payzen.eu/static/js/krypton-client/V4
.0/ext/material.js">
```

By

```
Code Example: src="https://static.payzen.eu/static/js/krypton-client/V4
.0/ext/neon.js">
```

If you want to know more about the theme, click on the "Themes" link.

Advanced personalization

If you have customized the embedded form, click on this link "[Advanced customization](/en/rest/V4.0/javascript/redirection/custom.html) ". This page describes how to customize your new payment form.

- For example: replace on your CSS the class name : `class="kr-embedded"`

By`class="kr-smart-form"`

III. Payment Analysis - Instant Notification

An additional object is returned at the end of the payment. This new object describes the payment method used by the buyer:

- The `paymentMethodDetails`

You can ignore it if you wish. It does not have to be taken into account. You can keep your current implementation of instant notification at the end of the payment (IPN).

For example,

si vous utilisez la carte de TEST "VISA" avec ce**pan**(card number) "4970 1000 0000 0014", the object will contain`paymentMethodDetails`

this result:

"transactions":[{ (...) "transactionDetails":{ (...) "paymentMethodDetails":{ "id":"497010XXXXXX0014", (...) }, }, },],

List mode with embedded card

Source: https://docs.lyra.com/en/rest/V4.0/javascript/redirection/use_case_embedded.html

- Display order:

- Compatible payment methods are displayed according to the order of priority of the contracts associated with the store. - This order can be configured from the Back Office (link:[Changing the display order](/en/rest/V4.0/javascript/redirection/custom_order.html)).

- Integration:

- Add the **kr-smart-form**class and the**kr-card-form-expanded**attribute, - Set the parameter `kr-form-token`

with the`formToken`

generated.

. Code sample:

```
Code Example:
```

(...)

List mode

Source: https://docs.lyra.com/en/rest/V4.0/javascript/redirection/use_case_list.html

- Display order:

- Compatible payment methods are displayed according to the order of priority of the contracts associated with the store. - This order can be configured from the Back Office (link:[Changing the display order](/en/rest/V4.0/javascript/redirection/custom_order.html)).

- Integration:

- Add the **kr-smart-form**class, - Set the parameter `kr-form-token`

with the`formToken`

generated.

. Code sample:

```
Code Example:
```

(...)

- Specific use cases:

If the only compatible payment method is card payment, the payment form will be displayed with the fields embedded.

highlighting several payment methods

Source: https://docs.lyra.com/en/rest/V4.0/javascript/redirection/use_case_many_payment_method.html

The highlighting of several payments is possible whatever the chosen display mode:

This function allows you to position several buttons of a payment method where you want on your page.

Integration

- Use the **kr-smart-button**class; - add **kr-payment-method**attribute to choose a payment method.

See the [table of values for the kr-payment-method](/en/rest/V4.0/javascript/redirection/reference_smartform.html#tableau-de-lattribut-kr-payment-method) attribute.

Case for 2 payment buttons

Implement 2`DIV`

, **une par bouton**.

These 2`DIV`

sont ensuite imbriquées dans une`DIV PARENT`

to apply a CSS style.

Example: .

```
Code Example:
```

(...)

(...)

Use Case: Highlight Paypal and Apple Pay

Below is an example for the Paypal and Apple Pay payment method, in **list mode with embedded card**.

For information, you can use**CSS Flexbox**to create your layout and correctly position the payment method buttons.

For more information, click here:[CSS Flexbox](https://www.w3schools.com/css/css3_flexbox.asp).

. Code sample:

<head> (...) <!-- link Bootstrap --> </head> <style type="text/css"> /*Flexbox direction row*/ .container { display: flex; flex-direction: row; } /*Change the button : width, border*/ .container .kr-smart-button { width : 75% !important; border-style: solid; border-color: orange; } /*Gap between the container and the div*/ .container>div { margin: 0 100px; flex: 1; } </style> <body> <!-- payment form --> <div class="kr-smart-form" kr-card-form-expanded kr-form-token="[GENERATED FORMTOKEN]">   </body>

Highlight a payment method

Source: https://docs.lyra.com/en/rest/V4.0/javascript/redirection/use_case_one_payment_method.html

The payment method can be highlighted regardless of the display mode:

This function allows you to position the button of a payment method where you want it on your page.

Integration

- Use the **kr-smart-button**class; - add **kr-payment-method**attribute to choose a payment method.

See the [table of values for the kr-payment-method](/en/rest/V4.0/javascript/redirection/reference_smartform.html#tableau-de-lattribut-kr-payment-method) attribute.

Code sample

```
Code Example:
```

Use case: highlight Paypal

Below is an example for the Paypal payment method, in **list mode with embedded card**.

For information, you can use**CSS Flexbox**to create your layout and correctly position the payment method buttons.

For more information, click here:[CSS Flexbox](https://www.w3schools.com/css/css3_flexbox.asp).

. Code sample:

<head> (...) <!-- link Bootstrap --> </head> <style type="text/css"> /*Flexbox direction row*/ .container { display: flex; flex-direction: row !important; } /*Change the button : width, border*/ .container .kr-smart-button { width : 75% !important; border-style: solid; border-color: orange; } /*Gap between the container and the div*/ .container>div { margin: 0 100px; flex: 1; } </style> <body> <!-- payment form --> <div class="kr-smart-form" kr-card-form-expanded kr-form-token="[GENERATED FORMTOKEN]">   </body>

Pop-in mode

Source: https://docs.lyra.com/en/rest/V4.0/javascript/redirection/use_case_popin.html

- Display order:

- Compatible payment methods are displayed according to the order of priority of the contracts associated with the store. - This order can be configured from the Back Office (link:[Changing the display order](/en/rest/V4.0/javascript/redirection/custom_order.html)).

- Integration:

- Add the **kr-smart-form**class and the**kr-popin**attribute, - Set the parameter `kr-form-token`

with the`formToken`

generated.

. Code sample:

```
Code Example:
```

(...)

SmartForm parameters

Source: https://docs.lyra.com/en/rest/V4.0/javascript/redirection/reference_smartform.html

The general parameters available for the smartForm are:

| Parameter | Description | |---|---| | kr-form-token | Encrypted object allowing to initialize the smartForm with the transaction details. | | paymentMethods | When creating the`formToken` , this field is used for filtering the compatible payment methods offered by the merchant. | | kr-smart-form | Class that must be integrated for implementing the smartForm. | | kr-popin | It is associated with the kr-smart-form class and defines the display pop-in mode | | kr-card-form-expanded | It is associated with the kr-smart-form class and defines the display list mode with embedded card | | kr-no-card-logo-header | Attribute to hide card payment logos (|

kr-payment-methodattribute for selecting the desired payment method. (Links for[highlighting a payment
method](/en/rest/V4.0/javascript/redirection/use_case_many_payment_method.html)or[highlighting several payment
methods](/en/rest/V4.0/javascript/redirection/use_case_many_payment_method.html))[table of the kr-payment-method attribute](/en/rest/V4.0/javascript/redirection/reference_smartform.html#tableau-de-lattribut-kr-payment-method)to see the possible values.[table of the kr-brands attribute](/en/rest/V4.0/javascript/redirection/reference_smartform.html#tableau-de-lattribut-kr-brands)to check the available values.### Table of **kr-payment-method** attribute

| value | Description | |---|---| | CARDS | Selection of all card payment methods. | | Titre-Restaurant(*) cards | | | APETIZ | Selection of payment method Bimpli (ex Apetiz). | | EDENRED | Selection of payment method Ticket restaurant. | | SODEXO | Selection of payment method Sodexo. | | CHQ_DEJ | Selection of payment method Chèque déjeuner. | | Other payment methods | | | IP_WIRE | Selection of the SEPA Credit Transfer payment method. | | BIZUM | Selection of the Bizum payment method. | | APPLE_PAY | Selection of the Apple Pay payment method. | | ALMA_2X | Select the Alma 2-installment payment method. | | ALMA_3X | Select the Alma 3-installment payment method. | | ALMA_4X | Select the Alma 4-installment payment method. | | ALMA_10X | Select the Alma 10-installment payment method. | | ALMA_12X | Selection of the Alma payment method in 12 installments. | | PAYCONIQ | Sélection du moyen de paiement Payconiq. | | PAYPAL_SB | Selection of the payment methodPayPalin TEST mode. | | PAYPAL | Selection of the payment methodPayPalin PRODUCTION mode. | | PAYPAL_BNPL_SB | Selection of the payment methodPayPal Pay Laterin TEST mode. | | PAYPAL_BNPL | Selection of the payment methodPayPal Pay Laterin PRODUCTION mode. | | UPI | Sélection du moyen de paiement BHIM-UPI. |

Table of **kr-brands** attribute

Value the `kr-brands`

attribute to create a payment button among the card payment methods.

| value | Description | |---|---| | CB | Selection of the CB network. | | VISA | Selection of the Visa brand. | | MASTERCARD | Selection of the Mastercard brand. | | MAESTRO | Selection of the Maestro payment method. | | VISA_ELECTRON | Selection of the Visa Electron payment method. | | AMEX | Selection of the American Express payment method. |

Then, add the `kr-payment-method`

attribute set to **CARDS**.

For example, generate a payment button for **AMEX** :

Step 4: Display the payment form

Source: https://docs.lyra.com/en/rest/V4.0/javascript/guide/display/presentation.html

Objective

- Display all payment fields (card number, expiration date, CVV, etc.) on your website.

I. Initialize the payment form

In the `HEAD`

,,

- Load **our JavaScript library**in a tag called`<script>`

.

src="https://static.lyra.com/static/js/krypton-client/V4.0/stable/kr-payment-form.min.js"

- Implement **the public key**.

Set the **public key** in the parameter `kr-public-key`

<head> (...) <script type="text/javascript"
src="https://static.lyra.com/static/js/krypton-client/V4.0/stable/kr-payment-form.min.js"
kr-public-key="69876357:testpublickey_DEMOPUBLICKEY95me92597fd28tGD4r5" (...) > </script> (...)
</head>

As a reminder, it corresponds to the [3 rd key of the
table](/en/rest/V4.0/javascript/guide/keys/presentation.html).

3. Add other initialization parameters

- Add **2 initialization parameters**, for example:

| name | Description | |---|---| | kr-post-url-success | URL to which the buyer's browser is redirected in case of a successful payment. (recommended) | | kr-language | Defines the language in which the form will be displayed. Accepts ISO 639-1 (en or en-US). If the parameter is not defined, the form uses the language of the browser. |

. Code sample:

<head> (...) <script type="text/javascript"
src="https://static.lyra.com/static/js/krypton-client/V4.0/stable/kr-payment-form.min.js"
kr-public-key="69876357:testpublickey_DEMOPUBLICKEY95me92597fd28tGD4r5"
kr-post-url-success="paid.php" kr-language="en-EN"> </script> (...) </head>

Référez-vous à ces 2 rubriques [Paramètres
généraux](/en/rest/V4.0/javascript/features/general_parameter.html) et [Paramètres de
personnalisation des "placeholders"](/en/rest/V4.0/javascript/features/custom_parameter.html) pour
les autres paramètres. Après la balise `<script>`

, choisissez un **thème**.

4. Choose a theme

Neon is the new default theme. Here are the links to load this theme.

<head> (...) <link rel="stylesheet"
href="https://static.lyra.com/static/js/krypton-client/V4.0/ext/neon-reset.min.css"> <script
src="https://static.lyra.com/static/js/krypton-client/V4.0/ext/neon.js"> </script> (...) </head>

More info: [themes](/en/rest/V4.0/javascript/redirection/themes.html).

5. Adding JS functions (optional)

In another

`<script>`

tag, you can integrate events or JS methods.For more details,

[Event presentation](/en/rest/V4.0/javascript/redirection/presentation_methods.html)and[Method
presentation](/en/rest/V4.0/javascript/redirection/presentation_methods.html).

Example of code for the `HEAD`

without JS function

<head> <!-- STEP : 1 : load the JS librairy 2 : required public key 3 : the JS parameters url sucess and
langage EN --> <script type="text/javascript"
src="https://static.lyra.com/static/js/krypton-client/V4.0/stable/kr-payment-form.min.js"
kr-public-key="69876357:testpublickey_DEMOPUBLICKEY95me92597fd28tGD4r5"
kr-post-url-success="paid.php" kr-language="en-EN"> </script> <!-- theme NEON should be loaded in
the HEAD section --> <link rel="stylesheet"
href="https://static.lyra.com/static/js/krypton-client/V4.0/ext/neon-reset.min.css"> <script
src="https://static.lyra.com/static/js/krypton-client/V4.0/ext/neon.js"> </script> </head>

II. Define the display mode

Dans le `BODY`

, choisissez le mode d'affichage et valorisez le paramètre `kr-form-token`

avec le `formToken`

généré ([Etape 3 : Créer un formToken](/en/rest/V4.0/javascript/guide/formToken/presentation.html)).

| Mode | Description | |---|---| list | Displays a list of available and compatible payment methods Default display. | pop-in | Displays a payment button that opens a pop-in containing available and compatible payment methods. | list with embedded card | Displays the embedded fields for card payment and the available and compatible payment methods. |

List mode (default) | Pop-in mode | List mode with embedded card | |---|---|---| <body> (...) </body> | <body> <div class="kr-smart-form" kr-popin kr-form-token="[GENERATED FORMTOKEN]"> (...) </body> | <body> <div class="kr-smart-form" kr-card-form-expanded kr-form-token="[GENERATED FORMTOKEN]"> (...) </body> | | More info: |

[Pop-in mode](/en/rest/V4.0/javascript/redirection/use_case_popin.html)[List mode with embedded card](/en/rest/V4.0/javascript/redirection/use_case_embedded.html)You also have the choice :

III. Advanced customization

This step is optional.

Example of customization :

[Hide card payment logos](/en/rest/V4.0/javascript/redirection/custom_without_logo.html)[Changing the display order](/en/rest/V4.0/javascript/redirection/custom_order.html)[Personalization of payment buttons.](/en/rest/V4.0/javascript/redirection/custom_button.html)[Personalize the page layout](/en/rest/V4.0/javascript/redirection/custom_row.html)

IV. Simplified example in PHP

In the `sample`

folder, the file for this step is **smartForm.php**.

For more details, [Example file: smartForm.php](/en/rest/V4.0/javascript/guide/display/file.html)

Step 3: Create a formToken

Source: https://docs.lyra.com/en/rest/V4.0/javascript/guide/formToken/presentation.html

Objective

- Generate a formToken.

The `formToken`

is a key generated by the gateway. It defines the payment context (amount, currency, order number, buyer contact details, etc.).

The `formToken`

is the **result of the response** of the call to the **Charge/CreatePayment** REST Web Service. It is valid for 15 minutes.

I. Request for a simple payment

Example: .

- amount : 9.90 €. - order: "myOrderId-999999", - buyer's email: "sample@example.com".

/en/rest/V4.0/api/kb/authentication.html

https://github.com/lyra/rest-php-examples/blob/master/www/minimalEmbeddedForm.php#L9-L44

https://api.lyra.com/api-payment/V4/Charge/CreatePayment

{ "amount": 990, "currency": "EUR", "orderId": "myOrderId-999999", "customer": { "email": "sample@example.com" } }

{ "amount": 1500, "currency": "PEN", "orderId": "myOrderId-999999", "more": "parameters", "customer": { "email": "sample@example.com" } }

{ "amount": 20000, "currency": "ARS", "orderId": "myOrderId-999999", "more": "parameters", "customer": { "email": "sample@example.com" } }

{ "amount": 100000, "currency": "COP", "orderId": "myOrderId-999999", "more": "parameters", "customer": { "email": "sample@example.com" } }

{ "amount": 2500, "currency": "BRL", "orderId": "myOrderId-999999", "more": "parameters", "customer": { "email": "sample@example.com" } }

II. Response for a simple payment

{ "status": "SUCCESS", "_type": "V4/WebService/Response", "webService": "Charge/CreatePayment", "applicationProvider": "PAYZEN", "version": "V4", "applicationVersion": "4.1.0", "answer": { "formToken": "DEMO-TOKEN-TO-BE-REPLACED", "_type": "V4/Charge/PaymentForm" } }

III. Other use cases

[Select payment methods](/en/rest/V4.0/javascript/redirection/custom_filter.html)[Offering payment method registration.](/en/rest/V4.0/javascript/guide/features.html#proposer-lenregistrement-du-moyen-de-paiement)[Increasing the chances of frictionless in 3DS2](/en/rest/V4.0/javascript/guide/features.html#augmenter-les-chances-de-frictionless-en-3ds2)[Transmitting custom data](/en/rest/V4.0/javascript/guide/features.html#transmettre-des-donnees-personnalisees)

Other examples are available on this page: [Usage cases](/en/rest/V4.0/javascript/guide/features.html).

IV. Using the Playground

Use the [Charge/CreatePayment](/en/rest/V4.0/api/playground/?ws=Charge%2FCreatePayment) Playground to see **all available fields**.

Click the "Test me" button, create your request and execute the REST call in order to generate the formToken.

V. Simplified PHP example

In the `sample`

folder, the file for this step is **formToken.php**.

For more details, [Example file: formToken.php](/en/rest/V4.0/javascript/guide/formToken/file.html)

Personalization parameters of "placeholders"

Source: https://docs.lyra.com/en/rest/V4.0/javascript/features/custom_parameter.html

Different parameters can be customized when the JavaScript client is being loaded.

Classes à utiliser pour la personnalisation des champs embarqués sont les suivantes :

| PARAMETER | Description | |---|---| | kr-pan | Card number. | | kr-expiry | Expiry date | | kr-security-code | Security code (CVV) | | kr-label-do-register | the label of the checkbox on the payment form to register the card |

The classes to be used for the personalization of "placeholders" are:

| PARAMETER | required | Description | LATAM case | |---|---|---|---| | kr-placeholder-expiry | Placeholder of the kr-expiry field (expiry date). | || | kr-placeholder-pan | Placeholder of the kr-pan field (card number). | || | kr-placeholder-security-code | Placeholder of the kr-security-code field (CVV). |

Code sample

Personalize the placeholders, using the following code:

<script src="https://static.lyra.com/static/js/krypton-client/V4.0/stable/kr-payment-form.min.js" kr-public-key="69876357:testpublickey_DEMOPUBLICKEY95me92597fd28tGD4r5" kr-placeholder-pan="XXXX XXXX XXXX 1234" kr-placeholder-expiry="ex : 01/25" kr-placeholder-security-code="XXX"> </script>

General settings

Source: https://docs.lyra.com/en/rest/V4.0/javascript/features/general_parameter.html

Supported platforms

We make every effort to support all recent versions of the major browsers available on the market.

For security reasons and to deliver the best user experience to the majority of our merchants, we do not support browsers and operating systems that no longer receive security patches.

These browsers represent a minor part of the traffic making payments on the Internet.

We support:

- Edgefrom version 17 - Chromefrom version 70 onwards - Firefoxfrom version 64 - Safari(desktop and mobile) from version 11 onwards - Android native browserfromAndroid 5.0 - All the latest versions of IOS starting from iPhone 4S.

TLS 1.2 must be supported by the browser.

We proactively test most browsers, both mobile and desktop. But it is possible that a mobile + browser combination escapes our vigilance. In this case, please contact our support.

If you want to support a combination that is not taken into account by our JavaScript client, you can implement our redirect form.

Also note that some antivirus or ad-blocker software may block our solution.Please contact support if you notice any improper detection.

Initialization parameters

Various parameters can be defined while loading the JavaScript client. For example, in order to define **kr-public-key** and **kr-post-url-success** :

The available general parameters are:

| PARAMETER | required | Description | |---|---|---| | kr-public-key | YES | Public key for making a payment. | | kr-language | Language used for displaying the form in Culture format (en-US). | | | kr-post-url-success | URL to which the form is submitted (POST method) if successful. | | | kr-get-url-success | URL to which the form is submitted (POST method) if successful. | | | kr-post-url-refused | URL called when all attempts have failed (POST method). | | | kr-get-url-refused | URL called when all attempts have failed (GET method). | | | kr-clear-on-error | Disables the removal of CVV in case of rejected transaction (true or false). | | | kr-hide-debug-toolbar | Hides the debug sidebar

in test mode (true or false). | | | kr-spa-mode | If the value is true, the form is not automatically initialized (false is the default value). |

Step 5: Analyze the payment result

Source: https://docs.lyra.com/en/rest/V4.0/javascript/guide/analyse/presentation.html

Objective

Analyzing the payment result:

Via the IPN (Instant Payment Notification), during a server-to-server call.

Via the return to the shop, during the browser return.

I. Response structure

| PARAMETER | Description | |---|---| | kr-hash-key | Type of key to sign the`kr-answer` . The possible values are:`password` for the IPN /`sha256_hmac` for the return to shop. | | kr-hash-algorithm | Algorithm used to calculate the hash. Its value is sha256_hmac. | | kr-answer | Object containing the payment result, encoded in JSON. | | kr-answer-type | Type the JSON object stored in kr-answer. | | kr-hash | Hash of the JSON object stored in kr-answer. It allows to verify the authenticity of the response. |

- Go to the **REST API Keys**tab, from the**Settings > Shop**to retrieve your keys.

II. Analyze the IPN (Instant Payment Notification)

It is **imperative** to retrieve and analyze the IPN payment data.

- Retrieve the JSON from the IPN ([IPN settings](/en/rest/V4.0/javascript/guide/notification/ipn.html)) - Check the authenticity of the notification with the value of kr-hash (**2nd**key[of the REST API key table](/en/rest/V4.0/javascript/guide/keys/presentation.html)) - Check the payment status

More info: [NPI Analysis (notification URL)](/en/rest/V4.0/api/kb/ipn_usage.html).

III. Analyze the response when returning to the shop

In the `HEAD`

, implement the **kr-post-url-success** initialization parameter to receive the payment result in case of successful payment (Step 4: Display the payment form).

- Retrieve the JSON posted in the browser - Check the authenticity of the notification with the value of kr-hash (**4th**key[of the REST API key table](/en/rest/V4.0/javascript/guide/keys/presentation.html)) - Check the payment status

.

VI. Simplified PHP example

In the `sample`

folder, the sample files are:

- ipn.php. - paid.php.

For more information, click on: [Example files: ipn.php and paid.php](/en/rest/V4.0/javascript/guide/analyse/file.html)

Example file: smartForm.php

Source: https://docs.lyra.com/en/rest/V4.0/javascript/guide/display/file.html

This file is used for displaying the payment `smartForm`

in List mode.

In the `HEAD`

,,

load

our JavaScript libraryintegrate

the public key, mandatory initialization parameterintegrate

other initialization parameters, like a URL in case of accepted paymentchoose

a theme(neon theme)

In the `BODY`

,,

define

the display mode(List mode)use ,

the formToken, created in step 3, in parameter`kr-form-token`

<?php include_once 'config.php'; ?> <?php include_once 'formToken.php'; ?> <head> <!-- STEP : 1 : load the JS librairy 2 : required public key 3 : the JS parameters url sucess and langage EN --> <meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1.0, user-scalable=no" /> <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" /> <meta http-equiv="X-UA-Compatible" content="IE=edge" /> <script type="text/javascript" src="<?php echo DOMAIN_URL; ?>/static/js/krypton-client/V4.0/stable/kr-payment-form.min.js" kr-public-key="<?php echo PUBLIC_KEY; ?>" kr-post-url-success="paid.php"; kr-language="en-EN"> </script> <!-- theme NEON should be loaded in the HEAD section --> <link rel="stylesheet" href= "<?php echo DOMAIN_URL; ?>/static/js/krypton-client/V4.0/ext/neon-reset.css"> <script src= "<?php echo DOMAIN_URL; ?>/static/js/krypton-client/V4.0/ext/neon.js"> </script> </head> <body> <div class="kr-smart-form" kr-form-token="<?php echo $formToken; ?>" > </body> </html>

Step 2: Authenticate him/herself

Source: https://docs.lyra.com/en/rest/V4.0/javascript/guide/keys/presentation.html

Objective

- Secure data exchanges.

I. Retrieving the keys

Click on

Other Actions, from theMerchant Back Office . The following window will appear:

- In the **Settings > Shop**menu, select your shop and go to the**REST API keys**tab.

Table of REST API keys :

For each use, there is a key for the TEST mode and a key for the PRODUCTION mode.

| NUMBER | KEY NAME | Description | |---|---|---| | 1 | User | For calls to REST Web Services | | 2 | password | For calls to REST Web Services and to verify the authenticity of data returned in the IPN | | 3 | Public key. | For creating a payment form in the Buyer's browser. | | 4 | HMAC-SHA-256 Key | In order to verify the authenticity of the data returned during the payment form response in the browser |

This action is performed by a user who is authorized to manage keys and signing algorithms. Additionally, access to the **REST API Keys** tab requires specific functionality to be enabled. Please contact

More info: [REST API keys](/en/rest/V4.0/api/get_my_keys.html).

II. Authentication phase

- Create a string with the **user**and the**password**separated by a colon (:)

- the **username**is the shop identifier:**N° 1: user of the REST API key table** - the **Password**, for example, for the test storetestpassword_DEMOPRIVATEKEY23G4475zXZQ2UA5x7M :**2: REST API key table password**

Encode the obtained chain in base64

Add the “Authorization” header to your request containing the word “Basic” followed by the string encoded in base64:

`Authorization: Basic Njk4NzYzNTc6dGVzdHBhc3N3b3JkX0RFTU9QUklWQVRFS0VZMjNHNDQ3NXpYWlEyVUE1eDdN`

For example, for

PHP, the**Authorization**header will be calculated as follows:

`$header = "Authorization: Basic " . base64_encode($username . ':' . $password);`

Here is an example of a request. You can use [the TEST SDK](/en/rest/V4.0/api/playground/?ws=Charge%2FSDKTest) provided in the Playground to test your integration.

The complete HTTP request will resemble the image below. You can use the [TEST SDK](/en/rest/V4.0/api/playground/?ws=Charge%2FSDKTest) available via the Playground.

{ "value": "my testing value" }

III. Code samples

Code samples, in **different languages** , [are available in GitHub](https://gist.github.com/brandonmwest/a2632d0a65088a20c00a).

IV. Simplified example in PHP

In the `sample`

folder, the file of this step is **config.php**. It allows you to define the authentication keys.

For more details, [Example file: config.php](/en/rest/V4.0/javascript/guide/keys/file.html).

Personalization of payment buttons.

Source: https://docs.lyra.com/en/rest/V4.0/javascript/redirection/custom_button.html

Le bouton : "Payer"

The payment button is automatically added to your form using the following code:

There are numerous benefits to using our payment button:

- The labels are translated automatically. - The amount is automatically formatted and updated. - We handle the pending transition animation for you - The button automatically changes to read-only mode if necessary.

It is possible to override the payment button by proceeding to the:

Name personalization

<button class="kr-payment-button">Custom label</button>

You can also insert a variable that represents the amount and the currency.

<button class="kr-payment-button">this will cost %amount-and-currency% !!</button>

Color customization

The style of the "Pay" button is defined by the **kr-payment-button** class.

To override the color of the "Pay" button, it is recommended to set the new style in the `<HEAD>`

tag of your payment page, right after the theme and Javascript code are loaded.

Here is an example using the css rule `!important`

.

<head> <!-- Javascript library. Should be loaded in head section --> <script type="text/javascript" src="https://static.lyra.com/static/js/krypton-client/V4.0/stable/kr-payment-form.min.js" kr-public-key="69876357:testpublickey_DEMOPUBLICKEY95me92597fd28tGD4r5:testpublickey_DEMOPUBLICKEY95me92597fd28tGD4r5"> </script> <!-- theme and plugins. should be loaded in the HEAD section --> <link rel="stylesheet" href="https://static.lyra.com/static/js/krypton-client/V4.0/ext/neon-reset.min.css"> <script type="text/javascript" src="https://static.lyra.com/static/js/krypton-client/V4.0/ext/neon.js"></script> <!-- Overrride payment button background color --> <style type="text/css"> .kr-smart-form .kr-payment-button {

background-color: #00D152 !important; } </style> </head> <body> ... </body>

Buttons for compatible payment methods

Customize the name and the icon of the buttons for compatible payment methods, using the code below:

```
Code Example:
```

KR.onFormReady(function (){ KR.setFormConfig({ smartForm: { paymentMethods: { "[NAME]": { label: "[CUSTOM_NAME]", icon : "[SRC_PICTURE]", } } }, }); });

Utilisez les méthodes `KR.onFormReady`

et `KR.setFormConfig`

, dans le **header** de votre page de paiement (pour plus de détails, consultez [les fonctions JS](/en/rest/V4.0/javascript/redirection/presentation_methods.html)).

Integrate the following data:

| Attribute | Description | FORMAT | |---|---|---| | "[NAME]" | Sélectionne le moyen de paiement (lien vers le |

Illustration for personalizing the SEPA Credit Transfer payment method

- Use the value `IP_WIRE`

with the attribute`paymentMethods`

to choose the payment method.**SEPA Credit Transfer**. - Change the name to **Payment by bank transfer**. - Change the icon with an external CDN link.

. Code sample:

```
Code Example:
```

KR.onFormReady(function (){ KR.setFormConfig({ language:'en', smartForm: { paymentMethods: { "IP_WIRE": { label: "Payment by bank transfer", icon : "https://img.icons8.com/external-wanicon-lineal-wanicon/64/null/external-bank-money-exchange-wanicon-lineal-wanicon.png", }, } }, }); });

Personalize the page layout

Source: https://docs.lyra.com/en/rest/V4.0/javascript/redirection/custom_row.html

Horizontal alignment

To align , **horizontally** , the fields of the embedded form, use the , **Flexbox in CSS** , with the ,`flex-direction`

property, set to `row`

.

The fields of the embedded form are enclosed in one or more containers:

...

Inside a parent DIV:

...

For more information [CSS Flexbox (Flexible Box)](https://www.w3schools.com/css/css3_flexbox.asp).

Example

Code sample

<head> <!-- Javascript library. Should be loaded in head section --> <script type="text/javascript" src="https://static.lyra.com/static/js/krypton-client/V4.0/stable/kr-payment-form.min.js" kr-public-key="69876357:testpublickey_DEMOPUBLICKEY95me92597fd28tGD4r5:testpublickey_DEMOPUBLICKEY95me92597fd28tGD4r5"> </script> <!-- theme and plugins. should be loaded in the HEAD section --> <link rel="stylesheet" href="https://static.lyra.com/static/js/krypton-client/V4.0/ext/neon-reset.min.css"> <script type="text/javascript" src="https://static.lyra.com/static/js/krypton-client/V4.0/ext/neon.js"></script> <style type="text/css"> /*choice the size*/ .kr-smart-form { width: 33% !important; } /*to choice the size of the embedded form*/ .kr-smart-form .kr-embedded { width: 100% !important; } /*to use the CSS Flexbox (Flexible Box)*/ .kr-smart-form .kr-embedded .flex-container { flex-direction: row !important; display: flex; } /*tocenter the button*/ .kr-smart-form .kr-embedded .kr-payment-button { margin-left: auto; margin-right: auto; width: 100% !important; } </style> </head> <body> <div class="kr-smart-form" kr-card-form-expanded kr-form-token="[GENERATED FORMTOKEN]">   <button class="kr-payment-button"> </button>   </body>

Compact mode

The compact mode is used for customizing the display of the **list mode** and of the **list mode with embedded card**.

In **list mode** , the payment method buttons are displayed in the same row (in pairs).

Intégez l'attribut `smartForm: { layout: 'compact' }`

en utilisant les fonctions `KR.onFormReady()`

et [KR.setFormConfig()](/en/rest/V4.0/javascript/features/kr_setFormConfig.html), dans le **header** de votre page de paiement (plus de détails, consultez [les fonctions JS](/en/rest/V4.0/javascript/redirection/presentation_methods.html)).

. Code sample:

```
Code Example:
```

KR.onFormReady(function (){ KR.setFormConfig({ smartForm: { layout: 'compact' }, }); });

In **list mode with embedded card** , the expiration date and CVV are displayed on the same line.

Intégrez l'attribut `cardForm: { layout: 'compact' }`

en utilisant les fonctions `KR.onFormReady()`

et [KR.setFormConfig()](/en/rest/V4.0/javascript/features/kr_setFormConfig.html), dans le **header** de votre page de paiement (plus de détails, consultez [les fonctions JS](/en/rest/V4.0/javascript/redirection/presentation_methods.html)).

. Code sample:

```
Code Example:
```

KR.onFormReady(function (){ KR.setFormConfig({ cardForm: { layout: 'compact' }, }); });

Combine the attributes ,`cardForm`

, and `smartForm`

You can combine the attributes `cardForm`

and `smartForm`

if your shop allows payments by card and with compatible payment methods.

. Code sample:

Code Example:

```
KR.onFormReady(function (){ KR.setFormConfig( { cardForm: { layout: 'compact' }, smartForm: { layout: 'compact' }, } ); });
```

Presentation of methods

Source: https://docs.lyra.com/en/rest/V4.0/javascript/redirection/presentation_methods.html

The JavaScript client supports **events** during integration: [Initialize payment form](/en/rest/V4.0/javascript/guide/embedded/presentation.html).

| Method | Description | SmartForm case | |---|---|---| |

a pop-in (or pop-up)without giving a choice to the buyer.[KR.getPaymentMethods()](/en/rest/V4.0/javascript/features/kr_getPaymentMethods.html)[KR.userPaymentMethodsOrder()](/en/rest/V4.0/javascript/features/kr_userPaymentMethodsOrder.html) |

[KR.setFormConfig()](/en/rest/V4.0/javascript/features/kr_setFormConfig.html)[initialization parameters](/en/rest/V4.0/javascript/features/general_parameter.html)from`formToken`

.[KR.setBrand()](/en/rest/V4.0/javascript/features/kr_setBrand.html)**Obsolete**, to use`KR.validateForm()`

[KR.validateForm()](/en/rest/V4.0/javascript/features/kr_validateForm.html)The following methods are obsolete and are no longer supported. **They should not be used** :.

- KR.validate(): use KR.validateForm() - KR.registerPlugin()

Display management in on-board mode.

| Method | Description | |---|---| |

[KR.fields.pan.help.button.show()](/en/rest/V4.0/javascript/redirection/custom_cvv.html#icone-daide-du-pan)[KR.fields.cvv.hide()](/en/rest/V4.0/javascript/redirection/custom_cvv.html##cvv)[KR.fields.cvv.show()](/en/rest/V4.0/javascript/redirection/custom_cvv.html##cvv)Pop-in display management.

| Method | Description | |---|---| | KR.closePopin() | Closes the pop-in (if open). | | KR.openPopin() | Opens the Pop-in (if closed) | | KR.setShopName() | Change the name of the shop defined in the header of the Pop-in. |

Dynamic form management (add, remove DOM):

| Method | Description | |---|---| | KR.addForm(CSS class or id) | Adds a form to a DOM element. Returns a formId. | | KR.attachForm(CSS class or id) | Obsolete, to use`KR.renderElements()` | | KR.renderElements(CSS class or id) | Enable the form on an existing DOM. Returns a formId. | | KR.hideForm(formId) | Hide the form | | KR.removeEventCallbacks() | Deletes all previously attached callbacks using KR.on[*] functions. | | KR.removeForms() | Removes all forms from the DOM (automatically calls KR.removeEventCallbacks()) | | KR.showForm(formId) | Displaying a form |

See: [embedded-form-glue](https://github.com/lyra/embedded-form-glue).

Managing the payment form submission button:

| PARAMETER | Description | |---|---| | KR.button.setLabel('MON LABEL %amount-and-currency% ') | Defines a label where %amount-and-currency% will be replaced by the amount and currency | | KR.button.showSpinner() | Displays the waiting animation | | KR.button.hideSpinner() | Hides the waiting animation | | KR.button.disable() | Disables the button (not clickable) | | KR.button.enable() | Activate the button |

Themes

Source: https://docs.lyra.com/en/rest/V4.0/javascript/redirection/themes.html

The embedded form includes 2 ready-to-use themes. Each theme requires for corresponding CSS and JS files to be loaded.

The CSS theme file allows to apply a basic theme while waiting for the payment form to finish loading. This is particularly important on devices with slow connection. It must always be placed in the header of the page.

The JS theme file contains the active part of the theme (animations, styles, elements, etc.). It must be loaded before the main JavaScript library.

Each of these themes can be used in list mode with embedded card, or in pop-in mode.

Neon theme

neon is the default theme. The associated files are:

| files | description | |---|---| | neon-reset.min.css | Applies the neon theme by forcing the styles (!important). | | neon.css | Applies the neon theme while taking into account the styles of the page. | | neon.js | Active part of the neon theme |

Example of neon theme:

Code sample for displaying the neon theme:

<!DOCTYPE html> <html> <head> <meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1.0, user-scalable=no" /> <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" /> <meta http-equiv="X-UA-Compatible" content="IE=edge" /> <!-- STEP : 1 : load the JS librairy 2 : required public key 3 : the JS parameters url sucess --> <script type="text/javascript" src="https://static.lyra.com/static/js/krypton-client/V4.0/stable/kr-payment-form.min.js" kr-public-key="69876357:testpublickey_DEMOPUBLICKEY95me92597fd28tGD4r5" kr-post-url-success="paid.html"> </script> <!-- theme NEON should be loaded in the HEAD section --> <link rel="stylesheet" href="https://static.lyra.com/static/js/krypton-client/V4.0/ext/neon-reset.min.css"> <script src="https://static.lyra.com/static/js/krypton-client/V4.0/ext/neon.js"> </script> </head> <body> <!-- payment form --> <div class="kr-smart-form" kr-card-form-expanded kr-form-token="DEMO-TOKEN-TO-BE-REPLACED"> <!-- error zone -->   </body> </html>

Classic theme

The associated files are:

| files | description | |---|---| | classic-reset.min.css | Applies the classic theme by forcing the styles (!important). | | classic.css | Applies the classic theme while taking into account the styles of the page. | | classic.js | Active part of the classic theme. |

Example of classic theme:

Example of code for displaying the classic theme:

<!DOCTYPE html> <html> <head> <meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1.0, user-scalable=no" /> <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" /> <meta http-equiv="X-UA-Compatible" content="IE=edge" /> <!-- STEP : 1 : load the JS librairy 2 : required public key 3 : the JS parameters url sucess --> <script type="text/javascript" src="https://static.lyra.com/static/js/krypton-client/V4.0/stable/kr-payment-form.min.js" kr-public-key="69876357:testpublickey_DEMOPUBLICKEY95me92597fd28tGD4r5" kr-post-url-success="paid.html"> </script> <!-- theme NEON should be loaded in the HEAD section --> <link rel="stylesheet" href="https://static.lyra.com/static/js/krypton-client/V4.0/ext/classic-reset.min.css"> <script src="https://static.lyra.com/static/js/krypton-client/V4.0/ext/classic.js"> </script> </head> <body> <!-- payment form --> <div class="kr-smart-form" kr-card-form-expanded kr-form-token="DEMO-TOKEN-TO-BE-REPLACED"> <!-- error zone -->   </body> </html>

The result will be:

Classic theme (pop-in)

You can also display the **classic** theme in a pop-in by adding the **kr-popin** attribute:

Material theme (not compatible)

Form without a theme

If you want to create a custom theme, it is recommended to include the **no-theme-css** CSS. It ensures minimum compatibility with all browsers (desktop and mobile) on the market :

| files | description | |---|---| | no-theme.min.css | Applies minimal CSS to guarantee that the form works properly. |

Customizing a theme

The embedded form (as well as the pop-in) applies the styles in 2 steps:

- by loading a CSS file (e.g. classic-reset.min.css) in the header of the page. - Afterwards, the theme is refined thanks to a configuration object (included in classic.js).

The initial CSS file

This file allows to reserve the space and apply a minimal style to the form before JavaScript is loaded and executed.

We recommend always loading this CSS file in the page header. **classic-reset.min.css** , or **no-theme.min.css** are two examples of the initial CSS files provided.

The configuration object

JavaScript theme files (such as **classic.js** or **material.js**) contain a configuration object that defines the whole theme: animations, styles, HTML elements.

The only difference between the classic, material, embedded or pop-in forms is in this configuration object.

The configuration object reference

| PARAMETER | Type | Description | |---|---|---| | form.fields.order | string list | Default field order (if not included), such as ["pan", "securityCode", "expiry"]. | | form.controls.order | string list | Default controls order (if not included) such as ["formButton", "error"]. | | form.layout | string | Payment form layout: default or compact. | | merchant.header.image.src | string | Image url or data:image (type supported by CSS). | | merchant.header.image.type | string | Background (occupies the entire header) or logo (round centered logo). | | merchant.header.image.visibility | boolean | True/false: if false, the image is hidden. | | merchant.header.shopName.color | string | Color of shop name. Example: ‘red’ (CSS attribute). | | merchant.header.shopName.gradient | boolean | True/false: applies (or not) a gradient in the header. | | merchant.header.backgroundColor | string | Background color of the header. Example: ‘red’ (CSS attribute). |

Examples of configuration of the pop-in header

The configuration object must be passed as follows:

let config = { "merchant": { "header": { "image": { "visibility": false } } } }; KR.setFormConfig(config);

Here are some examples of configuration of the pop-in header.

Change the logo:

{ "merchant": { "header": { "image": { "type": "logo", "visibility": true, "src": "<https://www.logomoose.com/wp-content/uploads/2018/02/logomoosedogandowl-011.jpg>" } } } }

Pass an image in string format:

{ "merchant": { "header": { "image": { "type": "background", "visibility": true, "src": "

pjYz0iaHR0cDovL2NyZWF0aXZlY29tbW9ucy5vcmcvbnMjIgogICB4bWxuczpyZGY9Imh0dHA6Ly93d3
cudzMub3JnLzE5OTkvMDIvMjItcmRmLXN5bnRheC1ucyMiCiAgIHhtbG5zOnN2Zz0iaHR0cDovL3d3dy
53My5vcmcvMjAwMC9zdmciCiAgIHhtbG5zPSJodHRwOi8vd3d3LnczLm9yZy8yMDAwL3N2ZyIKICAg
aWQ9InN2ZzQiCiAgIHZlcnNpb249IjEuMSIKICAgdmlld0JveD0iMCAwIDU3NiA1MTIiPgogIDxtZXRhZG
F0YQogICAgIGlkPSJtZXRhZGF0YTEwIj4KICAgIDxyZGY6UkRGPgogICAgICA8Y2M6V29yawogICAgI
CAgICByZGY6YWJvdXQ9IiI+CiAgICAgICAgPGRjOmZvcm1hdD5pbWFnZS9zdmcreG1sPC9kYzpmb3
JtYXQ+CiAgICAgICAgPGRjOnR5cGUKICAgICAgICAgICByZGY6cmVzb3VyY2U9Imh0dHA6Ly9wdXJ
sLm9yZy9kYy9kY21pdHlwZS9TdGlsbEltYWdlIiAvPgogICAgICAgIDxkYzp0aXRsZT48L2RjOnRpdGxlP
gogICAgICA8L2NjOldvcms+CiAgICA8L3JkZjpSREY+CiAgPC9tZXRhZGF0YT4KICA8ZGVmcwogICAg
IGlkPSJkZWZzOCIgLz4KICA8cGF0aAogICAgIHN0eWxlPSJmaWxsOiNmZmZmZmY7ZmlsbC1vcGFja
XR5OjE7c3Ryb2tlLXdpZHRoOjAuMzkzNDMyNjgiCiAgICAgaWQ9InBhdGgyIgogICAgIGQ9Im0gMzczL
jI2NjY0LDI3My44Mjk5NyAxOC41OTg3NSwtODEuODMzOTkgYyAxLjM0Mjc4LC01LjkwODU3IC0zLjE
0ODI1LC0xMS41MzUwNCAtOS4yMDc1MSwtMTEuNTM1MDQgSCAyMjguMTI0NiBsIC0zLjYwNjIsLT
E3LjYyOTcyIGMgLTAuODk4NjEsLTQuMzk0MjYgLTQuNzY1MjUsLTcuNTQ5OTkgLTkuMjUwNzgsLTc
uNTQ5OTkgaCAtNDAuMzM4MjcgYyAtNS4yMTQ5NCwwIC05LjQ0MjM4LDQuMjI3NDQgLTkuNDQyM
zgsOS40NDIzOCB2IDYuMjk0OTQgYyAwLDUuMjE0OTQgNC4yMjc0NCw5LjQ0MjM5IDkuNDQyMzgs
OS40NDIzOSBoIDI3LjQ5NDI3IGwgMjcuNjM3ODcsMTM1LjExODU0IGMgLTYuNjEyMDUsMy44MDI1
MyAtMTEuMDY3NjcsMTAuOTMyMzEgLTExLjA2NzY3LDE5LjEwNzA2IDAsMTIuMTY4MDggOS44NjQ
xNCwyMi4wMzIyMyAyMi4wMzIyMywyMi4wMzIyMyAxMi4xNjgwOSwwIDIyLjAzMjIzLC05Ljg2NDE1IDIy
LjAzMjIzLC0yMi4wMzIyMyAwLC02LjE2NjY3IC0yLjUzNjQ2LC0xMS43MzgwNyAtNi42MTkxMSwtMTU
uNzM3MzEgaCA4Mi40ODE5NyBjIC00LjA4MjI2LDMuOTk5MjQgLTYuNjE4NzIsOS41NzA2NCAtNi42M
Tg3MiwxNS43MzczMSAwLDEyLjE2ODA4IDkuODY0MTYsMjIuMDMyMjMgMjIuMDMyMjQsMjIuMDMy
MjMgMTIuMTY4MDgsMCAyMi4wMzIyMywtOS44NjQxNSAyMi4wMzIyMywtMjIuMDMyMjMgMCwtOC4
3MjMyIC01LjA3MDU2LC0xNi4yNjEzNyAtMTIuNDI0MjEsLTE5LjgzMDk3IGwgMi4xNzA1NywtOS41NT
A5NyBjIDEuMzQyNzcsLTUuOTA4NTkgLTMuMTQ4MjUsLTExLjUzNTA2IC05LjIwNzUsLTExLjUzNTA2
IEggMjUxLjMwMTMzIGwgLTIuNTc1MDIsLTEyLjU4OTg2IGggMTE1LjMzMjgyIGMgNC40MDg4MSwwI
DguMjMwNjEsLTMuMDUwNjcgOS4yMDc1MSwtNy4zNDk3MSB6IiAvPgo8L3N2Zz4K" } } } }

Creating you own themes

The payment form fields are customizable via standard CSS directives. All you need to do is apply them and they will be automatically transferred, even for the elements contained in the iframes of sensitive fields.

Thanks to a system of hidden fields, the JavaScript client will automatically retrieve the styles of your page and include them in iframes.

Use cases

Source: https://docs.lyra.com/en/rest/V4.0/javascript/guide/features.html

Here is a non-exhaustive list of use cases with the `formToken`

creation (when calling the **Charge/createPayment** Web Service).

Transmitting the order number

To transmit the order number, use the **orderId** field:

{ "orderId" : "CX-1254" }

Transmitting buyer details

The Merchant may transmit the Buyer's billing address and contact information (e-mail address, title, phone number etc.).

These details will be:

- Visible in the Merchant Back Office , in the transaction details (**Buyer**tab), - Returned in the IPN, without modification.

To do this, use the **customer** field:

Example

{ "customer": { "reference":"C2383333540", "email" : " <sample@example.net>", "billingDetails" : { "category" : "PRIVATE", "title" : "M", "firstName" : "Laurent", "lastName" : "DURANT", "streetNumber" : "109", "address" : "rue de l'innovation", "zipCode" : "31670", "city" : "LABEGE", "country" : "FR", "phoneNumber" : "0123456789", "cellPhoneNumber" :"0623456789" } } }

Transmitting delivery details

The Merchant can transmit the Buyer's delivery details (address, title, phone number etc.).

These details will be:

- Visible in the Merchant Back Office , in the transaction details (**Buyer**tab), - Returned in the IPN, without modification.

Example for a delivery of "In-store pickup" type.

The delivery address is the same as the shop address.

The billing address is different from the delivery address.

The name of the delivery recipient is the same one as in the billing address.

{ "customer": { "shippingDetails": { "shippingMethod": "RECLAIM_IN_SHOP", "shippingSpeed": "STANDARD", "streetNumber": "230", "address": "avenue des Champs Elysées", "zipCode": "59170", "city": "CROIX", "country": "FR", "firstName": "Marie-Charlotte", "lastName": "GRIMALDI", "phoneNumber": "0328386789" } } }

Example for a "Pickup point" type delivery

The delivery address is the same as the pickup point address.

The name of the pickup point is transmitted in the second line of the delivery address.

The address of the pickup point is transmitted in the first line of the delivery address.

The recipient's name is the one mentioned in the billing address.

The billing address is different from the delivery address.

{ "customer": { "shippingDetails": { "shippingMethod": "RELAY_POINT", "shippingSpeed": "STANDARD", "streetNumber": "100", "address": "avenue du parc Barbieux", "address2": "Pressing du Parc", "zipCode": "59170", "city": "CROIX", "country": "FR", "firstName": "Martine", "lastName": "DURAND", "phoneNumber": "0328386789", "deliveryCompanyName":"Chronospost" } } }

Transmitting the contents of the shopping cart

When creating a payment, the Merchant can transmit the contents of the Buyer's cart.

These details will be visible in the **Shopping cart** tab).

For this, use the **cartItemInfo** (json object array) field when calling the **Charge/CreatePayment** web service.

Example for defining 2 items in the shopping cart

{ "customer": { "shoppingCart": { "cartItemInfo": [{ "productRef": "myRef1", "productAmount": "1200", "productLabel": "myLabel1", "productQty" : "1" }, { "productRef": "myRef2", "productAmount": "2400", "productLabel": "myLabel2", "productQty" : "1" }] } } }

Notes:

- The **cartItemInfo**field is always returned to**empty**in the response. - In order for the **Shopping cart**tab to be displayed correctly in theMerchant Back Office , you must pass at least the**productAmount**field of each product.

Send the sub-merchant data

The Payment Facilitator can transmit the data of the sub-merchant involved in the transaction.

This data will be :

- Visible in the Merchant Back Office , in the transaction details (**Buyer**tab), - Returned in the IPN, without modification.

The table describes the common fields available with sub-merchant information.

| Field name | FORMAT | Description | |---|---|---| | ans..255 | Address of the sub-merchant.Transmitted by the payment facilitator. | | ans..255 | Addition of the sub-merchant's address. Transmitted by the payment facilitator. | | an..128 | City of the sub-merchant.Transmitted by the payment facilitator. | | ans..60 | Company type of the sub-merchant.Transmitted by the payment facilitator. | | a2 | Country code of the sub-merchant address (ISO 3166 alpha-2 standard). Transmitted by the payment facilitator. | | ans..128 | Payment Facilitator ID. Transmitted by the payment facilitator. | | ans..24 | Legal identifier of the sub-merchant.Transmitted by the payment facilitator. | | n4 | MCC code of the sub-merchant.Transmitted by the payment facilitator. | | n..64 | Contract number (MID) of the sub-merchant. Transmitted by the payment facilitator. | | ans..255 | Business name of the sub-merchant.Transmitted by the payment facilitator. | | an..32 | Telephone number of the sub-merchant.Transmitted by the payment facilitator. | | ans..255 | Label (Soft descriptor) of the sub-merchant that appears on the buyer's bank statement. Transmitted by the payment facilitator. | | ans..128 | Region of the sub-merchant's address. Transmitted by the payment facilitator. | | ans..128 | URL of the sub-merchant.Transmitted by the payment facilitator. | | an..64 | Zip code of the sub-merchant.Transmitted by the payment facilitator. |

Offering payment method registration.

The Merchant may offer the Buyer to facilitate their purchases by requesting to register banking details via the payment gateway.

With this operation, the payment gateway allocates a unique token to the payment method and returns it to the Merchant via the **paymentMethodToken** field.

Thus, the Buyer will no longer have to enter their payment method details during subsequent purchases.

This solution brings an additional level of security to payments since only the token that can only be used by the payment gateway is transmitted.

To offer the Buyer to register their payment method during the payment, use the **formAction** field with the **ASK_REGISTER_PAY** value.

{ "formAction" : "ASK_REGISTER_PAY", "customer": { "email": " <sample@example.net>" } }

Note:

The **e-mail** field becomes mandatory when recording the payment method.

When the form is displayed, a checkbox will appear.

By default, this box is unchecked.

If the Buyer accepts to record their payment method, they must tick the box.

If the payment is refused, the payment method will not be recorded.

Force payment method registration

If the Merchant has notified the buyer earlier in the purchase process, they can "force" payment method registration without displaying an additional checkbox.

For this, use the **formAction** field with the **REGISTER_PAY** value:

{ "formAction" : "REGISTER_PAY", "customer": { "email": " <sample@example.net>" } }

Using a registered payment method

The Merchant can transmit the token to be debited when initiating the payment.

When the form is displayed, the **kr-pan** and **kr-expiry** fields will be automatically filled in. The buyer will only have to enter their security code (CVV) to finalize their purchase.

To do this, simply transmit the token to be debited in the **paymentMethodToken** field and set the **formAction** field to **PAYMENT**.

Since this is the default value, the **formAction** field is no longer needed.

{ "formAction" : "PAYMENT", "paymentMethodToken":"d3d9cc0d24a4400e9d123e9e1b8f1793", }

Using a registered payment method without displaying the embedded form

This mode allows you to create a transaction without displaying the payment form and without authentication (server-to-server call).

{ "formAction" : "SILENT", "paymentMethodToken":"d3d9cc0d24a4400e9d123e9e1b8f1793", }

Note:

- This use case does not use the JavaScript client, but only a server-to-server call. There is no redirection to the return url specified in **kr-post-url-success**. - The answer does not contain a **formToken**but directly a**Transaction**object. - Use the chaining reference in the `initialIssuerTransactionIdentifier`

field, otherwise the issuers may refuse the transaction in case the chaining is absent("Soft Decline"). - See: [0-click payment](/en/rest/V4.0/api/kb/zero_click_payment.html)

Transmitting merchant preferences

Utilisez le champ **strongAuthentication** (lien vers le playground : ** strongAuthentication**).

Ce champ indique la préférence du marchand au sujet de l'authentification de l'acheteur.

- Without cardholder interaction (**frictionless**). - With cardholder interaction (strong authentication or **challenge**). - No merchant preference.

| Use cases | Possible values | |---|---| CHALLENGE : With cardholder interaction | | | | | FRICTIONLESSWithout cardholder interaction“Frictionless 3DS2” option mandatory | If you do not have the “Frictionless 3DS” option, the choice of preference is delegated to the card issuer (No Preference). | | No merchant preference | | |

Table of exemptions (DISABLED value)

| exemption | Description | |---|---| | Low value transaction | For payments in EUR, you can requestan exemptionTo strong authentication: If the request for frictionless is accepted, the transaction does not benefit from the liability shift in case of a dispute by the cardholder. | | LRM (Low Risk Merchant) | CB's LRM (Low Risk Merchant) program aims to meet the expectations of merchants with very low risk and high volume (120,000 CB transactions / year). Vous pouvez demander If the request for frictionless is accepted, the transaction does not benefit from the liability shift in case of a dispute by the cardholder. |

Increasing the chances of frictionless in 3DS2

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

List of fields to be transmitted

| PARAMETER | Description | |---|---| | customer.email | Buyer's e-mail address. | | customer.billingDetails.identityCode | National identifier. Allows to identify each citizen within a country with a unique code. CPF or CNPJ in Brazil. | | customer.billingDetails.streetNumber | Street number of the billing address | | customer.billingDetails.address | Postal address | | customer.billingDetails.address2 | Address line 2 | | customer.billingDetails.zipCode | Zip code | | customer.billingDetails.city | City | | customer.billingDetails.state | State/Region | | customer.billingDetails.country | Country code according to the ISO 3166 alpha-2 standard | | customer.billingDetails.phoneNumber | Phone number | | customer.billingDetails.cellPhoneNumber | Cell phone number | | customer.shippingDetails.address | Postal address | | customer.shippingDetails.address2 | Address line 2 | | customer.shippingDetails.zipCode | Zip code | | customer.shippingDetails.city | City | | customer.shippingDetails.state | State/Region | | customer.shippingDetails.country | Country code according to ISO 3166 | | customer.shippingDetails.shippingMethod | Shipping mode. | | customer.shippingDetails.shippingSpeed | Shipping delay. |

Transmitting custom data

When creating a payment, the Merchant can transmit specific information to the payment gateway in order to meet their business needs.

For example: a file number, contract number, etc.

These details will be:

- Visible in the Merchant Back Office , in the transaction details (**Buyer**tab), - Visible in the payment confirmation e-mail sent to the Merchant. - Returned in the notification URL, without change.

For this, use the **metadata** (in JSON format) fields when calling the **Charge/CreatePayment** web service:

Example of transmitting a piece of data entitled "contract" and its value:

{ "metadata": { "contract": "1245KL-78/ZE" } }

Override the instant notification URL (not recommended)

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 incompatible with the execution of the request sent to the IPN URL via the

The URL configured in the notification rule is the one that will be called (see the "Setting up notifications" chapter).

Use the **ipnTargetUrl** field when initiating the payment to override the URL of the page to notify.

If the value in the **ipnTargetUrl** field is incorrect, the form will not be rejected.

{ "ipnTargetUrl" : "<https://my.site/my-ipn>", }

Example file : formToken.php

Source: https://docs.lyra.com/en/rest/V4.0/javascript/guide/formToken/file.html

This file is used to create the `formToken`

.

Request data

sending data with the CURL command

the call to the endpoint of the REST Web Service

`V4/Charge/CreatePayment`

the response containing the

`formToken`

```
Code Example:
```

<?php // STEP 1 : the data request $data = array('orderId' => uniqid('order_'), 'amount' => 250, 'currency' => 'EUR', 'customer' => array('email' => 'sample@example.com', 'reference' => uniqid('customer_'), 'billingDetails' => array('language' => 'fr', 'title' => 'M.', 'firstName' => 'Test', 'lastName' => 'Krypton', 'category' => 'PRIVATE', 'address' => '25 rue de l\'Innovation', 'zipCode' => '31000', 'city' => 'Testville', 'phoneNumber' => '0615665555', 'country' => 'FR')), 'transactionOptions' => array('cardOptions' => array('retry' => 1))); // STEP 3 : call the endpoint V4/Charge/CreatePayment with the json data. $response = post(SERVER."/api-payment/V4/Charge/CreatePayment", json_encode($data)); // Check if there is errors. if ($response['status'] != 'SUCCESS') { // An error occurs, throw exception $error = $response['answer']; throw new Exception('error ' . $error['errorCode'] . ': ' . $error['errorMessage']); } // Everything is fine, extract the formToken // STEP 4 : the answer with the creation of the formToken $formToken = $response['answer']['formToken']; function post($url, $data){ // STEP 2 : send data with curl command $curl = curl_init($url); curl_setopt($curl, CURLOPT_HEADER, false); curl_setopt($curl, CURLOPT_RETURNTRANSFER, true); curl_setopt($curl, CURLOPT_HTTPHEADER, array('Content-type: application/json')); curl_setopt($curl, CURLOPT_POST, true); curl_setopt($curl, CURLOPT_USERAGENT, 'test'); curl_setopt($curl, CURLOPT_USERPWD, USERNAME . ':' . PASSWORD); curl_setopt($curl, CURLOPT_HTTPAUTH, CURLAUTH_BASIC); curl_setopt($curl, CURLOPT_POSTFIELDS, $data); curl_setopt($curl, CURLOPT_CONNECTTIMEOUT, 45); curl_setopt($curl, CURLOPT_TIMEOUT, 45); curl_setopt($curl, CURLOPT_SSL_VERIFYHOST, 0); curl_setopt($curl, CURLOPT_SSL_VERIFYPEER, 0); $raw_response = curl_exec($curl); $status = curl_getinfo($curl, CURLINFO_HTTP_CODE); if (!in_array($status, array(200, 401))) {

```
curl_close($curl); throw new \Exception("Error: call to URL $url failed with unexpected status $status, response $raw_response."); } $response = json_decode($raw_response, true); if (!is_array($response)) { $error = curl_error($curl); $errno = curl_errno($curl); curl_close($curl); throw new \Exception("Error: call to URL $url failed, response $raw_response, curl_error $error, curl_errno $errno."); } curl_close($curl); return $response; } ?>
```

Example files: ipn.php and paid.php

Source: https://docs.lyra.com/en/rest/V4.0/javascript/guide/analyse/file.html

1. ipn.php

This file allows you to analyze the payment result using **the IPN**.

It is used for:

vérifier la signature

avec le mot de passe(qui commence par testpassword**ou prodpassword**) pour comparer avec la valeur du`kr-hash`

. ().**2 ème clé**du tableau des clés API RESTretrieve information from the

`kr-answer`

response (status, order number, unique UUID number ...)

https://github.com/lyra/rest-php-examples/blob/master/www/sample/ipn.php

2. paid.php

Ce fichier permet d'analyser le résultat du paiement lors du **retour à la boutique**.

It is used for:

vérifier de la signature avec la

Clé HMAC-SHA-256pour comparer avec la valeur du`kr-hash`

. ().**4 ème clé**du tableau des clés API RESTretrieve information from the

`kr-answer`

response (status, order number, unique UUID number ...)

https://github.com/lyra/rest-php-examples/blob/master/www/sample/paid.php

Step 1: Set the notification URL at the end of the payment

Source: https://docs.lyra.com/en/rest/V4.0/javascript/guide/notification/ipn.html

Objective

- Set the notification URL at the end of the payment in the Merchant Back Office .

I. Concept

IPNmeans**Instant Payment Notification**.

The IPN is a server-to-server notification of the payment result.

Notification is also sent in the following special cases:

- Internet connection disconnected during payment - The buyer closed their browser during the payment - The buyer did not complete their payment before the expiration of the payment session

II. Define the notification URL at the end of the payment

- Click on **Other Actions**, from theMerchant Back Office . The following window will appear:

- Go to the following menu: **Settings**>**Notification rules**. - Right-click **Instant Payment Notification**. - Select **Manage the rule**. - In the **General Setup**section, fill in the**Email address(es) to notify if IPN fails**to send. - Check the **Automatic retry in case of failure**checkbox if you want to allow the gateway to automatically resend the notification in case of failure (can be resent up to 4 times). - In the **REST API notification URL**section, fill in the URL of your page in the**URL of the IPN to call in TEST mode**and**URL of the IPN to call in PRODUCTION mode**fields. - Save the changes.

Sample file: config.php

Source: https://docs.lyra.com/en/rest/V4.0/javascript/guide/keys/file.html

See the chapter: **Retrieving the keys** :

Depuis le

For the DEMO store:

| NUMBER | VARIABLE | value | Description | |---|---|---|---| | 1 | USERNAME | 69876357 | For calls to REST Web Services | | 2 | PASSWORD | testpassword_DEMOPRIVATEKEY23G4475zXZQ2UA5x7M | For calls to REST Web Services | | 3 | PUBLIC_KEY | 69876357:testpublickey_DEMOPUBLICKEY95me92597fd28tGD4r5 | For creating a payment form in the Buyer's browser. | | 4 | SHA_KEY | 38453613e7f44dc58732bad3dca2bca3 | In order to verify the authenticity of the data returned during the payment form response in the browser |

Le cas échéant, à remplacer par les données de votre boutique.

Ajoutez les données suivantes :

| NUMBER | VARIABLE | value | Description | |---|---|---|---| | 5 | SERVER | https://api.lyra.com | To create the`formToken` | | 6 | URL_JS | https://static.lyra.com/static/js/krypton-client/V4.0/stable/kr-payment-form.min.js | To load our JS library |

```
Code Example: <?php /** * Define configuration * Configuration
initialisation, using Lyra account informations. * provided in your
Back Office (Menu: Settings > Shop > API REST Keys). **/ // DEMO SHOP
define('USERNAME', '69876357'); define('PASSWORD',
'testpassword_DEMOPRIVATEKEY23G4475zXZQ2UA5x7M'); define('PUBLIC_KEY',
'69876357:testpublickey_DEMOPUBLICKEY95me92597fd28tGD4r5');
define('SHA_KEY', '38453613e7f44dc58732bad3dca2bca3'); define('SERVER',
'https://api.lyra.com'); $URL_JS = 'https://static.lyra.com/static/js/k
rypton-client/V4.0/stable/kr-payment-form.min.js'; // SUBSTITUTE BY
MERCHANT SHOP (Menu: Settings > Shop > API REST Keys) //
define('USERNAME', 'KEY Number 1'); // define('PASSWORD', 'KEY Number
2'); // define('PUBLIC_KEY', 'KEY Number 3'); // define('SHA_KEY', 'KEY
Number 4'); // define('SERVER', 'KEY Number 5'); // $URL_JS = 'KEY
Number 6'; /*DOMAIN_URL : racine domaine URL_JS */ define('DOMAIN_URL',
strstr($URL_JS,'/static/',true)); ?>
```

KR.setFormConfig()

Source: https://docs.lyra.com/en/rest/V4.0/javascript/features/kr_setFormConfig.html

Description

This method allows to override [the initialization parameters](/en/rest/V4.0/javascript/features/general_parameter.html) , as well as the following elements:

| Use | Description | |---|---| | KR.setFormConfig({formToken: "NEW_FORM_TOKEN"}); | Changes the current formToken. | | KR.setFormConfig({language: "fr-FR"}); | Changes the language of the payment form and error messages. | | KR.setFormConfig({'kr-label-do-register': 'custom'}) | Sets the label of the "Save my card" checkbox |

This method returns a promise.

Example of integration

Here is an example for setting the language of the payment form to **English** :

```
Code Example:
```

$(document).ready(function(){ KR.setFormConfig({language:"en-EN"}); });

KR.field.focus()

Source: https://docs.lyra.com/en/rest/V4.0/javascript/features/kr_field_focus.html

Description

To give focus to a form field, you can use the **KR.field.focus(FIELD_CLASS)** method. You must pass the class of the embedded form field as a parameter.

Example of integration

For example, to add a button that will set focus on the expiry date field:

<button type="button" onclick="KR.fields.focus('kr-expiry')">Focus expiry field</button>

KR.getPaymentMethods()

Source: https://docs.lyra.com/en/rest/V4.0/javascript/features/kr_getPaymentMethods.html

Description

Cette méthode liste les moyens de paiement disponibles et associés à la boutique. Le marchand connaît en temps réel les moyens de paiement à sa disposition.

This method returns a promise.

Example of integration

```
Code Example: const result = await KR.getPaymentMethods();
console.log(result); // result { "paymentMethods": ["PAYPAL", "CARDS",
"APPLE_PAY"], "cardBrands": ["VISA", "MASTERCARD"] }
```

KR.openPaymentMethod()

Source: https://docs.lyra.com/en/rest/V4.0/javascript/features/kr_openPaymentMethod.html

Description

This function is used for directly offering a payment method by opening **a pop-in (or a pop-up)**. The merchant decides on the payment method without giving a choice to the buyer.

Example of integration (with smartForm)

To implement this method, here is an example of code, as an information:

```
Code Example:
```

document.addEventListener('DOMContentLoaded', function (){ KR.onFormReady(function (){ KR.openPaymentMethod('SELECTED PAYMENT METHOD').then().catch() }) });

KR.setBrand()

Source: https://docs.lyra.com/en/rest/V4.0/javascript/features/kr_setBrand.html

Description

Forces the detection of the payment method.The method takes as parameter the name of the payment method. The supported values are (not exhaustive):.

- VISA - VISA_DEBIT - VISA_ELECTRON - MASTERCARD - MASTERCARD_DEBIT - MAESTRO - AMEX - CB - (...)

Caution: If you set another value, it will be sent unchanged to the payment platform. If the value is not supported by your store configuration, no transaction will be created.

To re-enable automatic detection, call **KR.setBrand()** without any parameters.

Example of integration

To force the AMEX payment method, here is an example:

```
Code Example:
```

$(document).ready(function(){ KR.setBrand("AMEX"); });

KR.userPaymentMethodsOrder()

Source: https://docs.lyra.com/en/rest/V4.0/javascript/features/kr_userPaymentMethodsOrder.html

Description

This function allows you to manage the display order of the payment methods available in the shop.

Prerequisite: availability of the payment method

Pour gérer l'ordre d'affichage, vérifiez la disponibilité du moyen de paiement (en fonction de la devise, du montant minimum ou maximum, des contraintes techniques...) :

- If it is on the list of compatible payment methods (see [List of compatible payment methods](/en/rest/V4.0/javascript/redirection/compatible_payment_method.html)). - If its contract is associated with th shop via the Merchant Back Office . - It is present in the `paymentMethods`

field, optional field (see[Select payment methods](/en/rest/V4.0/javascript/redirection/custom_filter.html)).

If you use a payment method that is not available, a warning message appears in the browser console.

Display rules

By default, the order is defined in the

__Payment by cards__

If payment by card is available, it always appears first.

Example

- Payment by card is absent from the function.

```
Code Example: // function smartForm.userPaymentMethodsOrder =
['PAYPAL', 'APPLE_PAY'] // result Result = ['CARDS', 'PAYPAL',
'APPLE_PAY']
```

- Payment by card is not first on the list.

```
Code Example: // function smartForm.userPaymentMethodsOrder =
['PAYPAL', 'APPLE_PAY'] // result Result = ['CARDS', 'PAYPAL',
'APPLE_PAY']
```

__Other payment methods__

If the other payment methods are available and not listed in the function, they are displayed last.

Example

- For Apple Pay

```
Code Example: // your shop SHOP = ['CARDS', 'PAYPAL', 'APPLE_PAY'] //
function without APPLE_PAY smartForm.userPaymentMethodsOrder =
['CARDS', 'PAYPAL'] // result Result = ['CARDS', 'PAYPAL', 'APPLE_PAY']
```

- For PayPal

```
Code Example: // your shop SHOP = ['CARDS', 'PAYPAL', 'APPLE_PAY' ] //
function without APPLE_PAY smartForm.userPaymentMethodsOrder =
['CARDS', 'APPLE_PAY'] // result Result = ['CARDS', 'APPLE_PAY',
'PAYPAL']
```

Example of integration

```
Code Example: KR.setFormConfig({ smartForm: { userPaymentMethodsOrder:
['PAYPAL', 'APPLE_PAY', 'CARDS'] } } );
```

KR.validateForm()

Source: https://docs.lyra.com/en/rest/V4.0/javascript/features/kr_validateForm.html

Description

This method checks locally if the form is valid. It returns a promise:.

then()is called when the form is valid.**result**will be set to null.**catch()**is called when the form is invalid.**result**contains the details of the error.

Example of integration

KR.validateForm().then(({KR, result}) => { /* there is no error */ /* result == null */ }) .catch(({KR, result}) => { /* Get the error message */ var code = result.errorCode; var message = result.errorMessage; var myMessage = code + ": " + message; console.log(myMessage); /* if you have defined a callback using */ /* result.onError(), you can trigger it calling: */ return result.doOnError(); });

Once you have intercepted the errors, you can trigger the KR.onError() event manually by calling `result.doOnError();`

.

Step 4: Display the payment form

Source: https://docs.lyra.com/en/rest/V4.0/javascript/guide/embedded/presentation.html

Objective

- Display all payment fields (card number, expiration date, CVV, etc.) on your website.

I. Initialize the payment form

In the `HEAD`

of the page, you must include the following data in a `<script>`

tag.

- Load **our JavaScript library**

src="https://static.lyra.com/static/js/krypton-client/V4.0/stable/kr-payment-form.min.js"

- Implement **the public key**.

Set the **public key** in the parameter `kr-public-key`

<head> (...) <script type="text/javascript" (...)
kr-public-key="69876357:testpublickey_DEMOPUBLICKEY95me92597fd28tGD4r5" (...) </script> (...)
</head>

As a reminder, it corresponds to the [3 rd key of the
table](/en/rest/V4.0/javascript/guide/keys/presentation.html).

- Add more **initialization parameters**.

- Add **2 initialization parameters**, for example:

| name | Description | |---|---| | kr-post-url-success | URL to which the buyer's browser is redirected in case of a successful payment. (recommended) | | kr-language | Defines the language in which the form will be displayed. Accepts ISO 639-1 (en or en-US). If the parameter is not defined, the form uses the language of the browser. |

. Code sample:

<head> (...) <script type="text/javascript"
src="https://static.lyra.com/static/js/krypton-client/V4.0/stable/kr-payment-form.min.js"

kr-public-key="69876357:testpublickey_DEMOPUBLICKEY95me92597fd28tGD4r5" kr-post-url-success="paid.php"; kr-language="en-EN"> </script> (...) </head>

Référez vous à ces 2 rubriques [Paramètres généraux](/en/rest/V4.0/javascript/features/general_parameter.html) et [Paramètres de personnalisation des "placeholders"](/en/rest/V4.0/javascript/features/custom_parameter.html) pour connaître **les autres paramètres**. Après la balise `<script>`

, choisissez un **thème**.

- Choose **a theme**.

Classic is the default theme. Here are the links to load this theme.

<head> (...) <link rel="stylesheet" href="https://static.lyra.com/static/js/krypton-client/V4.0/ext/classic.css"> <script src="https://static.lyra.com/static/js/krypton-client/V4.0/ext/classic.js"> </script> (...) </head>

More info on [themes](/en/rest/V4.0/javascript/features/themes.html).

- Adding **JS functions**(**optional**):

In another

`<script>`

tag, you can integrate events or JS methods.For more details,

[Event presentation](/en/rest/V4.0/javascript/guide/embedded/presentation_methods.html)and[Method presentation](/en/rest/V4.0/javascript/guide/embedded/presentation_methods.html).

Example code for the `HEAD`

without JS function:

<head> <!-- STEP : 1 : load the JS librairy 2 : required public key with file config.php 3 : the JS parameters url sucess and langage EN --> <script type="text/javascript" src="https://static.lyra.com/static/js/krypton-client/V4.0/stable/kr-payment-form.min.js" kr-public-key="69876357:testpublickey_DEMOPUBLICKEY95me92597fd28tGD4r5" kr-post-url-success="paid.php"; kr-language="en-EN"> </script> <!-- theme CLASSIC should be loaded in the HEAD section --> <link rel="stylesheet" href="https://static.lyra.com/static/js/krypton-client/V4.0/ext/classic-reset.min.css"> <script src="https://static.lyra.com/static/js/krypton-client/V4.0/ext/classic.js"> </script> </head>

II. Define the display mode

In the `BODY`

, choose the display mode that suits you.

| Mode | Description | |---|---| embedded | Displays the fields of card payment (card number, expiry date, CVV, etc.) | pop-in | Displays a payment button that opens a pop-up window containing the payment fields (card number, expiry date, CVV, etc.). |

Embedded mode | Pop-in mode | |---|---| <body> (...) </body> | <body> <div class="kr-embedded" kr-popin kr-form-token="[GENERATED FORMTOKEN]"> (...) </body> | | More info: |

[Pop-in mode](/en/rest/V4.0/javascript/guide/embedded/use_case_popin.html)## III. Advanced customization

This step is optional.

Example of customization :

IV. Simplified example in PHP

In the `sample`

folder, the sample files are:

- embedded.php. - popin.php.

To get more details, [Example files: embedded.php and popin.php](/en/rest/V4.0/javascript/guide/embedded/file.html)

Show / Hide

Source: https://docs.lyra.com/en/rest/V4.0/javascript/redirection/custom_cvv.html

CVV

.

Example of integration

`KR.fields.cvv.hide();`

: function to hide the CVV.`KR.fields.cvv.show();`

: function to display the CVV.

<!-- code Jquery --> <script type="text/javascript"> $(document).ready(function() { // Show the CVV KR.fields.cvv.show(); }); </script>

PAN help icon

The buyer enters the PAN of his card, a help icon is displayed. It is possible to hide this icon.

Example of integration

`KR.fields.pan.help.button.hide();`

: function to hide the PAN help icon.`KR.fields.pan.help.button.show();`

: function to display the PAN help icon.

<!-- code Jquery --> <script type="text/javascript"> $(document).ready(function() { // Hide the pan help button KR.fields.pan.help.button.hide(); }); </script>

Personalizing the "Pay" button

Source: https://docs.lyra.com/en/rest/V4.0/javascript/features/custom_button.html

The payment button is automatically added to your form using the following code:

There are numerous benefits to using our payment button:

- The labels are translated automatically. - The amount is automatically formatted and updated. - We handle the pending transition animation for you - The button automatically changes to read-only mode if necessary.

It is possible to override the payment button by proceeding to the:

Personalize the name

If you want to manage the button label yourself, all you need to do is add it as follows:

<button class="kr-payment-button">Custom label</button>

You can also insert a variable that represents the amount and currency. The JavaScript client will automatically replace them:

<button class="kr-payment-button">this will cost %amount-and-currency% !!</button>

Personalize the color

The "Pay" button is placed in the `<div class="kr-embedded">`

container. The style of the "Pay" button is defined by the **kr-payment-button** class.

To override the color of the "Pay" button, it is recommended to set the new style in the `<HEAD>`

tag of your payment page, right after the theme and Javascript code are loaded.

Here is an example using the css rule `!important`

.

<head> <!-- Javascript library. Should be loaded in head section --> <script type="text/javascript" src="https://static.lyra.com/static/js/krypton-client/V4.0/stable/kr-payment-form.min.js" kr-public-key="69876357:testpublickey_DEMOPUBLICKEY95me92597fd28tGD4r5:testpublickey_DEMOPUBLICKEY95me92597fd28tGD4r5"> </script> <!-- theme and plugins. should be loaded in the HEAD section --> <link rel="stylesheet"

href="https://static.lyra.com/static/js/krypton-client/V4.0/ext/classic-reset.min.css"> <script type="text/javascript" src="https://static.lyra.com/static/js/krypton-client/V4.0/ext/classic.js"></script> <!-- Overrride payment button background color --> <style type="text/css"> .kr-embedded .kr-payment-button { background-color: #00D152 !important; } </style> </head> <body> ... </body>

Customization of the layout for the embedded form

Source: https://docs.lyra.com/en/rest/V4.0/javascript/features/custom_flexbox.html

To align , **horizontally** , the fields of the embedded form, you can use the , **Flexbox in CSS** , with the property ,`flex-direction`

, set to `row`

.

The fields of the embedded form are enclosed in one or more containers:

...

Inside a parent DIV:

...

For more information, check out this link: [CSS Flexbox (Flexible Box)](https://www.w3schools.com/css/css3_flexbox.asp).

A presentation of the **horizontal** display of embedded fields:

You will find a code sample:

<head> <!-- Javascript library. Should be loaded in head section --> <script type="text/javascript" src="https://static.lyra.com/static/js/krypton-client/V4.0/stable/kr-payment-form.min.js" kr-public-key="69876357:testpublickey_DEMOPUBLICKEY95me92597fd28tGD4r5:testpublickey_DEMOPUBLICKEY95me92597fd28tGD4r5"> </script> <!-- theme and plugins. should be loaded in the HEAD section --> <link rel="stylesheet" href="https://static.lyra.com/static/js/krypton-client/V4.0/ext/classic-reset.min.css"> <script type="text/javascript" src="https://static.lyra.com/static/js/krypton-client/V4.0/ext/classic.js"></script> <style type="text/css"> /* choice the embedded size */ .kr-embedded { width: 33% !important; } /* CSS Flexbox (Flexible Box) */ .kr-embedded .flex-container { display: flex; flex-direction: row !important; justify-content: center; align-items: center; } /* to center the button with the class kr-payment-button */ .kr-embedded .kr-payment-button { display: block; margin-left: auto; margin-right: auto; } </style> </head> <body> <!-- payment form --> <!--new flex-container class to indicate a flex-direction: row --> <!-- payment form fields -->   <!-- payment form submit button --> <button class="kr-payment-button"></button> </body>

Customization with an additional field: email

You will find an code sample by adding **the e-mail field** , as mandatory data.

<head> <!-- Javascript library. Should be loaded in head section --> <script type="text/javascript" src="https://static.lyra.com/static/js/krypton-client/V4.0/stable/kr-payment-form.min.js" kr-public-key="69876357:testpublickey_DEMOPUBLICKEY95me92597fd28tGD4r5:testpublickey_DEMOPUBLICKEY95me92597fd28tGD4r5"> </script> <!-- theme and plugins. should be loaded in the HEAD section --> <link rel="stylesheet" href="https://static.lyra.com/static/js/krypton-client/V4.0/ext/classic-reset.min.css"> <script type="text/javascript" src="https://static.lyra.com/static/js/krypton-client/V4.0/ext/classic.js"></script> <link rel="stylesheet" id="extraStyleCDN" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.1.1/css/all.min.css"> <style type="text/css"> /* to choice the embedded size */ .kr-embedded { width: 33% !important; } /* to use the CSS Flexbox (Flexible Box) */ .kr-embedded .flex-container { flex-direction: row !important; justify-content: space-between; width: 100%; display: flex; gap: 5px; } /* to have the email field the same width as the KR fields */ .kr-embedded .flex-container .kr-email { width: 100%; } /* to center the button with the class kr-payment-button */ .kr-embedded .kr-payment-button { margin-left: auto; margin-right: auto; display: block; width: 100%; } </style> </head> <body> <!-- payment form --> <!--new flex-container class to indicate a flex-direction: row -->   <input type="text" name="acme-email" placeholder="email" class="kr-theme" kr-icon="fas fa-envelope" required/>   <!--new flex-container class to indicate a flex-direction: row -->   <!-- payment form submit button --> <button class="kr-payment-button"></button> </body>

Customization in case of invalid data entry

You can customize the embedded form to warn the buyer in case of invalid data entry.

See the code sample provided on **GitHub** by clicking the link:

The code allows to display a warning message **in red** , below the erroneous embedded field.

Themes

Source: https://docs.lyra.com/en/rest/V4.0/javascript/features/themes.html

The embedded form includes 2 ready-to-use themes. Each theme requires for corresponding CSS and JS files to be loaded.

The CSS theme file allows to apply a basic theme while waiting for the payment form to finish loading. This is particularly important on devices with slow connection. It must always be placed in the header of the page.

The JS theme file contains the active part of the theme (animations, styles, elements, etc.). It must be loaded before the main JavaScript library.

Each of these themes can be used in embedded form mode or in pop-in mode.

Classic theme

Classic is the default theme. The associated files are:

| files | description | |---|---| | classic-reset.min.css | Applies the classic theme by forcing the styles (!important). | | classic.css | Applies the classic theme while taking into account the styles of the page. | | classic.js | Active part of the classic theme. |

Example of classic theme:

Example of code for displaying the classic theme:

Classic theme (pop-in)

You can also display the **classic** theme in a pop-in by adding the **kr-popin** attribute:

The result will be:

It is possible to personalize the image and the name of the shop appearing in the pop-in. For more information, go to: [JavaScript client reference](/en/rest/V4.0/javascript/features/reference.html).

Material Design theme

Material Theme applies the [guidelines defined by Google](https://material.io/design/). The associated files are:

| files | description | |---|---| | material-reset.css | Applies Material Theme by forcing styles (!important). | | material.css | Applies Material Theme while respecting the styles of the page. | | material.js | Active part

of the classic theme. |

Example of Material Theme:

Example of code for displaying Material Theme:

Material Theme (pop-in)

You can also display **Material** Theme in a pop-in by adding the **kr-popin** attribute:

Form without a theme

If you want to create a custom theme, it is recommended to include the **no-theme-css** CSS. It ensures minimum compatibility with all browsers (desktop and mobile) on the market :

| files | description | |---|---| | no-theme.min.css | Applies minimal CSS to guarantee that the form works properly. |

Customizing a theme

The embedded form (as well as the pop-in) applies the styles in 2 steps:

- by loading a CSS file (e.g. classic-reset.min.css) in the header of the page. - Afterwards, the theme is refined thanks to a configuration object (included in classic.js).

The initial CSS file

This file allows to reserve the space and apply a minimal style to the form before JavaScript is loaded and executed.

We recommend always loading this CSS file in the page header. **classic-reset.min.css** , or **no-theme.min.css** are two examples of the initial CSS files provided.

The configuration object

JavaScript theme files (such as **classic.js** or **material.js**) contain a configuration object that defines the whole theme: animations, styles, HTML elements.

The only difference between the classic, material, embedded or pop-in forms is in this configuration object.

The configuration object reference

| PARAMETER | Type | Description | |---|---|---| | form.fields.order | string list | Default field order (if not included), such as ["pan", "securityCode", "expiry"]. | | form.controls.order | string list | Default controls order (if not included) such as ["formButton", "error"]. | | form.layout | string | Payment form layout: default or compact. | | merchant.header.image.src | string | Image url or data:image (type supported by CSS). | | merchant.header.image.type | string | Background (occupies the entire header) or logo (round

centered logo). | | merchant.header.image.visibility | boolean | True/false: if false, the image is hidden. | | merchant.header.shopName.color | string | Color of shop name. Example: ‘red’ (CSS attribute). | | merchant.header.shopName.gradient | boolean | True/false: applies (or not) a gradient in the header. | | merchant.header.backgroundColor | string | Background color of the header. Example: ‘red’ (CSS attribute). |

Examples of configuration of the pop-in header

The configuration object must be passed as follows:

let config = { "merchant": { "header": { "image": { "visibility": false } } } }; KR.setFormConfig(config);

Here are some examples of configuration of the pop-in header.

Change the logo:

{ "merchant": { "header": { "image": { "type": "logo", "visibility": true, "src": "<https://www.logomoose.com/wp-content/uploads/2018/02/logomoosedogandowl-011.jpg>" } } } }

Pass an image in string format:

{ "merchant": { "header": { "image": { "type": "background", "visibility": true, "src": ""}

MjMgMTIuMTY4MDgsMCAyMi4wMzIyMywtOS44NjQxNSAyMi4wMzIyMywtMjIuMDMyMjMgMCwtOC4
3MjMyIC01LjA3MDU2LC0xNi4yNjEzNyAtMTIuNDI0MjEsLTE5LjgzMDk3IGwgMi4xNzA1NywtOS41NT
A5NyBjIDEuMzQyNzcsLTUuOTA4NTkgLTMuMTQ4MjUsLTExLjUzNTA2IC05LjIwNzUsLTExLjUzNTA2
IEggMjUxLjMwMTMzIGwgLTIuNTc1MDIsLTEyLjU4OTg2IGggMTE1LjMzMjgyIGMgNC40MDg4MSwwI
DguMjMwNjEsLTMuMDUwNjcgOS4yMDc1MSwtNy4zNDk3MSB6IiAvPgo8L3N2Zz4K" } } } }

Creating you own themes

The payment form fields are customizable via standard CSS directives. All you need to do is apply them and they will be automatically transferred, even for the elements contained in the iframes of sensitive fields.

Thanks to a system of hidden fields, the JavaScript client will automatically retrieve the styles of your page and include them in iframes.

Step 5: Analyze the payment result

Source: https://docs.lyra.com/en/rest/V4.0/javascript/guide/embedded/analyse.html

Objective

Analyzing the payment result:

Via the IPN (Instant Payment Notification), during a server-to-server call.

Via the return to the shop, during the browser return.

I. Response structure

| PARAMETER | Description | |---|---| | kr-hash-key | Type of key to sign the`kr-answer` . The possible values are:`password` for the IPN /`sha256_hmac` for the return to shop. | | kr-hash-algorithm | Algorithm used to calculate the hash. Its value is sha256_hmac. | | kr-answer | Object containing the payment result, encoded in JSON. | | kr-answer-type | Type the JSON object stored in kr-answer. | | kr-hash | Hash of the JSON object stored in kr-answer. It allows to verify the authenticity of the response. |

- Go to the **REST API Keys**tab, from the**Settings > Shop**to retrieve your keys.

II. Analyze the IPN (Instant Payment Notification)

It is **imperative** to retrieve and analyze the IPN payment data.

- Retrieve the JSON from the IPN ([IPN settings](/en/rest/V4.0/javascript/guide/notification/ipn.html)) - Check the authenticity of the notification with the value of kr-hash (**2nd**key[of the REST API key table](/en/rest/V4.0/javascript/guide/keys/presentation.html)) - Check the payment status

More info: [NPI Analysis (notification URL)](/en/rest/V4.0/api/kb/ipn_usage.html).

III. Analyze the response when returning to the shop

In the `HEAD`

, implement the **kr-post-url-success** initialization parameter to receive the payment result in case of successful payment (Step 4: Display the payment form).

- Retrieve the JSON posted in the browser - Check the authenticity of the notification with the value of kr-hash (**4th**key[of the REST API key table](/en/rest/V4.0/javascript/guide/keys/presentation.html)) - Check the payment status

.

VI. Simplified PHP example

In the `sample`

folder, the sample files are:

- ipn.php. - paid.php.

For more information, click on: [Example files: ipn.php and paid.php](/en/rest/V4.0/javascript/guide/analyse/file.html)

Sample files: embedded.php and popin.php

Source: https://docs.lyra.com/en/rest/V4.0/javascript/guide/embedded/file.html

1. embedded.php

This file is used to display the payment form, in **embedded mode**.

In the `HEAD`

,,

load

our JavaScript libraryintegrate

the public key, mandatory initialization parameterintegrate

other initialization parameters, like a URL in case of accepted paymentchoose

a theme(classic theme)

In the `BODY`

,,

define

the display modeuse ,

the formToken, created in step 3, in parameter`kr-form-token`

<?php include_once 'config.php'; ?> <?php include_once 'formToken.php'; ?> <head> <!-- STEP : 1 : load the JS librairy 2 : required public key 3 : the JS parameters url sucess and langage EN --> <meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1.0, user-scalable=no" /> <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" /> <meta http-equiv="X-UA-Compatible" content="IE=edge" /> <script type="text/javascript" src="<?php echo DOMAIN_URL; ?>/static/js/krypton-client/V4.0/stable/kr-payment-form.min.js" kr-public-key="<?php echo PUBLIC_KEY; ?>" kr-post-url-success="paid.php"; kr-language="en-EN"> </script> <!-- theme NEON should be loaded in the HEAD section --> <link rel="stylesheet" href= "<?php echo DOMAIN_URL; ?>/static/js/krypton-client/V4.0/ext/neon-reset.css"> <script src= "<?php echo DOMAIN_URL; ?>/static/js/krypton-client/V4.0/ext/neon.js"> </script> </head> <body> <div class="kr-embedded" kr-form-token="<?php echo $formToken; ?>" > </body> </html>

2. popin.php

This file is used to display the payment form, in **pop-in mode**.

In the `HEAD`

,,

load

our JavaScript libraryintegrate

the public key, mandatory initialization parameterintegrate

other initialization parameters, like a URL in case of accepted paymentchoose

a theme(classic theme)

In the `BODY`

,,

define ,

the display mode, (Pop-in mode by adding the attribute`kr-popin`

,).use ,

the formToken, created in step 3, in parameter`kr-form-token`

<head> <?php include_once 'config.php'; ?> <?php include_once 'formToken.php'; ?> <!-- STEP : 1 : load the JS librairy 2 : required public key 3 : the JS parameters url sucess and langage EN --> <meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1.0, user-scalable=no" /> <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" /> <meta http-equiv="X-UA-Compatible" content="IE=edge" /> <script type="text/javascript" src="<?php echo DOMAIN_URL; ?>/static/js/krypton-client/V4.0/stable/kr-payment-form.min.js" kr-public-key="<?php echo PUBLIC_KEY; ?>" kr-post-url-success="paid.php"; kr-language="en-EN"> </script> <!-- theme NEON should be loaded in the HEAD section --> <link rel="stylesheet" href= "<?php echo DOMAIN_URL; ?>/static/js/krypton-client/V4.0/ext/neon-reset.css"> <script src= "<?php echo DOMAIN_URL; ?>/static/js/krypton-client/V4.0/ext/neon.js"> </script> </head> <body> <div class="kr-embedded" kr-popin kr-form-token="<?php echo $formToken; ?>" > </body> </html>

Presentation of methods

Source: https://docs.lyra.com/en/rest/V4.0/javascript/guide/embedded/presentation_methods.html

The JavaScript client supports **events** during integration: [Initialize payment form](/en/rest/V4.0/javascript/guide/embedded/presentation.html).

| Method | Description | |---|---| |

[KR.setFormConfig()](/en/rest/V4.0/javascript/features/kr_setFormConfig.html)[initialization parameters](/en/rest/V4.0/javascript/features/general_parameter.html)from`formToken`

.[KR.setBrand()](/en/rest/V4.0/javascript/features/kr_setBrand.html)**Obsolete**, to use`KR.validateForm()`

[KR.validateForm()](/en/rest/V4.0/javascript/features/kr_validateForm.html)The following methods are obsolete and are no longer supported. **They should not be used** :.

- KR.validate(): use KR.validateForm() - KR.registerPlugin()

Display management in on-board mode.

| Method | Description | |---|---| |

[KR.fields.pan.help.button.show()](/en/rest/V4.0/javascript/redirection/custom_cvv.html#icone-daide-du-pan)[KR.fields.cvv.hide()](/en/rest/V4.0/javascript/redirection/custom_cvv.html##cvv)[KR.fields.cvv.show()](/en/rest/V4.0/javascript/redirection/custom_cvv.html##cvv)Pop-in display management.

| Method | Description | |---|---| | KR.closePopin() | Closes the pop-in (if open). | | KR.openPopin() | Opens the Pop-in (if closed) | | KR.setShopName() | Change the name of the shop defined in the header of the Pop-in. |

Dynamic form management (add, remove DOM):

| Method | Description | |---|---| | KR.addForm(CSS class or id) | Adds a form to a DOM element. Returns a formId. | | KR.attachForm(CSS class or id) | Obsolete, to use`KR.renderElements()` | | KR.renderElements(CSS class or id) | Enable the form on an existing DOM. Returns a formId. | | KR.hideForm(formId) | Hide the form | | KR.removeEventCallbacks() | Deletes all previously attached callbacks using KR.on[*] functions. | | KR.removeForms() | Removes all forms from the DOM (automatically calls KR.removeEventCallbacks()) | | KR.showForm(formId) | Displaying a form |

See: [embedded-form-glue](https://github.com/lyra/embedded-form-glue).

Managing the payment form submission button:

| PARAMETER | Description | |---|---| | KR.button.setLabel('MON LABEL %amount-and-currency% ') | Defines a label where %amount-and-currency% will be replaced by the amount and currency | | KR.button.showSpinner() | Displays the waiting animation | | KR.button.hideSpinner() | Hides the waiting animation | | KR.button.disable() | Disables the button (not clickable) | | KR.button.enable() | Activate the button |

Embedded mode

Source: https://docs.lyra.com/en/rest/V4.0/javascript/guide/embedded/use_case_embedded.html

Once you have the formToken, display the payment form.

- Add the **kr-embedded**class, - Set the parameter `kr-form-token`

with the`formToken`

generated.

. Code sample:

```
Code Example:
```

(...)

Example of a payment form:

Pop-in mode

Source: https://docs.lyra.com/en/rest/V4.0/javascript/guide/embedded/use_case_popin.html

Once you have the formToken, display the payment form.

- Add the **kr-embedded**class and the**kr-popin**attribute, - Set the parameter `kr-form-token`

with the`formToken`

generated.

. Code sample:

```
Code Example:
```

(...)

Example of a fonctional payment form:

JavaScript client reference

Source: https://docs.lyra.com/en/rest/V4.0/javascript/features/reference.html

The JavaScript client allows to create a payment form on your merchant website, using the following code:

For more detailed documentation, go to [Getting started: Single payment](/en/rest/V4.0/javascript/guide/start.html).

Supported platforms

We make every effort to support all recent versions of the major browsers available on the market.

For security reasons and to deliver the best user experience to the majority of our merchants, we do not support browsers and operating systems that no longer receive security patches.

These browsers represent a minor part of the traffic making payments on the Internet.

We support:

- Internet Explorer from version 10 onwards - Edge starting from version 17 onwards - Chrome from version 70 onwards - Firefox starting from version 64 onwards - Safari (Desktop and mobile) from version 11 onwards - Android native browser from Android 5.0 onwards - All the latest versions of IOS starting from iPhone 4S.

TLS 1.2 must be supported by the browser.

We proactively test most browsers, both mobile and desktop. But it is possible that a mobile + browser combination escapes our vigilance. In this case, please contact our support.

If you want to support a combination that is not taken into account by our JavaScript client, you can implement our redirect form.

Also note that some antivirus or ad-blocker software may block our solution.Please contact support if you notice any improper detection.

Initialization parameters

Various parameters can be defined while loading the JavaScript client. For example, in order to define **kr-public-key** and **kr-post-url-success** :

The available parameters are:

General settings:

| PARAMETER | required | Description | |---|---|---| | kr-public-key | YES | Public key for making a payment. | | kr-language | Language used for displaying the form in Culture format (en-US). | | | kr-post-url-success | URL to which the form is submitted (POST method) if successful. | | | kr-get-url-success | URL to which the form is submitted (POST method) if successful. | | | kr-post-url-refused | URL called when all attempts have failed (POST method). | | | kr-get-url-refused | URL called when all attempts have failed (GET method). | | | kr-clear-on-error | Disables the removal of CVV in case of rejected transaction (true or false). | | | kr-hide-debug-toolbar | Hides the debug sidebar in test mode (true or false). | | | kr-spa-mode | If the value is true, the form is not automatically initialized (false is the default value). |

Use these classes for customizing fields:

| PARAMETER | Description | |---|---| | kr-pan | Card number. | | kr-expiry | Expiry date | | kr-security-code | Security code (CVV) | | kr-installment-number | Number of instalments (LATAM specific field) | | kr-first-installment-delay | Differed Number of months to apply on the first instalment (LATAM specific field) | | kr-identity-document-type | Type of identity document (LATAM specific field) | | kr-identity-document-number | ID number (LATAM specific field) | | kr-card-holder-name | Name of the cardholder (LATAM specific field) | | kr-card-holder-mail | Cardholder's e-mail address (LATAM specific field) | | kr-do-register | the card token creation confirmation checkbox |

| PARAMETER | required | Description | |---|---|---| | kr-placeholder-expiry | Placeholder of the kr-expiry field (expiry date). |

Classes to be used for customizing placeholders:

| PARAMETER | required | Description | LATAM case | |---|---|---|---| | kr-placeholder-expiry | Placeholder of the kr-expiry field (expiry date). | || | kr-placeholder-pan | Placeholder of the kr-pan field (card number). | || | kr-placeholder-security-code | Placeholder of the kr-security-code field (CVV). | || | kr-placeholder-identity-document-type | Placeholder of the ID type field, if required | YES | | | kr-placeholder-identity-document-number | Placeholder of the ID number field, if required | YES | | | kr-placeholder-installments-default | Default payment method field placeholder | || | kr-placeholder-installments-single-payment | Placeholder of the field Payment method when there is only one | || | kr-placeholder-installments-single-payment-credit | Placeholder of the 'Payment method' field when there is only credit | || | kr-placeholder-first-installment-delay | Placeholder for the "Payment delay" field | YES | | | kr-placeholder-card-holder-mail | Placeholder of the e-mail field, if required | YES | | | kr-placeholder-card-holder-name | Placeholder of the holder name field if required | YES |

Classes to be used for customizing animated labels (specific to Material Theme):

| PARAMETER | required | Description | Template | LATAM case | |---|---|---|---|---| | kr-label-expiry | Animated label of the kr-expiry field (expiry date). | ||| | kr-label-pan | Animated label of the kr-pan field (card number). | ||| | kr-label-security-code | Animated label of the kr-security-code (CVV) field. | ||| | kr-label-identity-document-type | Animated label of the ID type field, if required | YES | || | kr-label-identity-document-number | Animated label of the ID number field, if required | YES | || |

kr-label-card-holder-mail | Animated label of the e-mail field, if required | YES | || |
kr-label-card-holder-name | Animated label of the cardholder name field, if required | YES | || |
kr-installments-label-singular | Animated label of the 'Payment method' field when there is only one | My Label [COUNT] with [CURRENCY] [AMOUNT] | || | kr-installments-label-plural | Animated label of the "Payment mode" field when there are several | My Label [COUNT] with [CURRENCY] [AMOUNT] | || |
kr-first-installment-delay-label-plural | Animated label of the "Payment delay" field when there is only one | My Label [COUNT] | YES | | | kr-first-installment-delay-label-without | Animated label of the "Payment delay" field when there are several | My Label [COUNT] | YES |

These parameters can also be overridden with the [KR.setFormConfig()](/en/rest/V4.0/javascript/features/reference.html#setFormConfig) method. For example, to override the **kr-post-url-success** parameter:

```
Code Example: KR.setFormConfig({ 'kr-post-url-success':
'https://my.site' }).then(({ KR }) => { /* there is no error */ );
```

Warning, if the **kr-post-url-success** and **kr-get-url-success** parameters are defined simultaneously, the POST method will have priority. The same applies to **kr-post-url-refused** and **kr-get-url-refused***.

Events

It is possible to attach custom JavaScript callbacks to different events. The JavaScript client supports the following events:

| PARAMETER | Description | |---|---| | KR.onBlur() | One of the form fields loses focus. see |

[KR.onBrandChanged()](/en/rest/V4.0/javascript/features/reference.html#kronbrandchanged)[KR.onError()](/en/rest/V4.0/javascript/features/reference.html#kronerror)[KR.onFocus()](/en/rest/V4.0/javascript/features/reference.html#kronfocus)[KR.onFormValid()](/en/rest/V4.0/javascript/features/reference.html#kronformvalid)[KR.onInstallmentChanged()](/en/rest/V4.0/javascript/features/reference.html#kroninstallmentchanged)[KR.onSubmit()](/en/rest/V4.0/javascript/features/reference.html#kronsubmit)[Buyer wallet management](/en/rest/V4.0/api/kb/wallets.html))[KR.onTransactionCreated()](/en/rest/V4.0/javascript/features/reference.html#krontransactioncreated)[KR.button.onClick()](/en/rest/V4.0/javascript/features/reference.html#krbuttononclick)[KR.on3dSecureAbort()](/en/rest/V4.0/javascript/features/reference.html#kron3dsecureabort)The following events are obsolete and are no longer supported. **They should not be used** :.

- KR.onFormReadyListener() - KR.onFormCreateListener()

All events return promises, allowing you to integrate them into a chain. See [Working in asynchronous environment](/en/rest/V4.0/javascript/spa/index_smart.html#promises) for more information.

KR.onBrandChanged()

The callback defined in **KR.onBrandChanged()** is called each time a card brand is detected. Example of integration:

The **cardInfo** object contains the **brand** property which can take the following values:

- amex - cb - mastercard - maestro - visa_electron - Visa

It also contains the **BIN** of the card, i.e. the first 6 characters of the PAN.

To view the complete list, see the reference documentation of the **effectiveBrand** parameter here: [Payment](/en/rest/V4.0/api/playground/?ws=Payment).

KR.onFormValid()

The callback defined in **KR.onFormValid()** is called once all the required fields are filled in and the data is valid. Integration example

The **cardInfo** object contains the **brand** property which can take the following values:

- amex - cb - mastercard - maestro - visa_electron - Visa

It also contains the **BIN** of the card, i.e. the first 6 characters of the PAN.

To view the complete list, see the reference documentation of the **effectiveBrand** parameter here: [Payment](/en/rest/V4.0/api/playground/?ws=Payment).

KR.onError()

KR.onError() allows to intercept errors before they appear.

Example of interception of error messages:

After that, error messages will automatically appear in the following element, if it is present:

When several errors are generated, they are merged into one single error. The **children** property will contain the details of all errors:

{ "errorCode": "CLIENT_300", "errorMessage": "Invalid form data", "children": [{ "errorCode": "CLIENT_301", "errorMessage": "Invalid card number", "field": "pan", (...) }, { "errorCode": "CLIENT_302", "errorMessage": "Invalid expiry date", "field": "expiryDate", (...) }, { "errorCode": "CLIENT_303", "errorMessage": "Invalid security code", "field": "securityCode", (...) }], "detailedErrorCode": null, "detailedErrorMessage": null, (...) }

For more information on errors, go to: [Managing errors (JS client)](/en/rest/V4.0/javascript/features/js_error_management.html)

KR.onFocus()

Kr.onFocus() allows to be notified when a form field is in focus:

Example of integration:

If the card number field is in focus, the **event** variable will take the following value:

{ "field":"pan", "formId":"F73867", "_type":"krypton/focus" }

You can also use **KR.onBlur()** to detect the loss of focus of a field. Its operation is similar to **KR.onFocus()**

KR.onInstallmentChanged()

KR.onInstallmentChanged() allows you to be notified when the buyer selects the number of installments.

```
Code Example:
```

$(document).ready(function(){ KR.onInstallmentChanged(function({KR, installmentInfo}){ console.log(installmentInfo) }); });

The **installmentInfo** object has the following attributes:

brand=> Brand of the card. Example: "VISA"**hasInterests**=> Boolean indicating whether an interest rate applies. Example: false**installmentCount**=> Total number of installments. Example: 1**totalAmount**=> Total amount, including interest. Example: 20000

KR.onSubmit()

KR.onSubmit() allows to intercept information about the authorized transaction before the form makes a POST to the URL defined with **kr-post-success-url**.

Example of integration:

The callback receives an object with 2 parameters:

- KR: Reference to the library - event: Object that contains the newly created transaction.

L'objet contenu dans **event** est le même que celui posté par le formulaire. Pour plus de détails, rendez-vous ici : [retour à la boutique](/en/rest/V4.0/kb/payment_done.html).

The behavior varies depending on the boolean returned by your function:

| Return value | Behavior | |---|---| | true | The JavaScript client performs a POST on kr-post-success-url. | | false | The POST with kr-post-success-url is not made. You manage the post-payment action yourself. |

KR.button.onClick()

KR.button.onClick() allows for custom processing before the JavaScript client validates the form and makes the call to create a transaction. **KR.button.onClick()** accepts either a callback or a promise as a parameter.

You can stop the execution string by returning **false** at the end of processing:

| Return value | Behavior | |---|---| | false | The execution is interrupted. Error handling does not take place. The transaction is not created. | | true | The execution continues normally when the callback is executed. |

KR.onTransactionCreated()

There are 2 callbacks that allow intercepting a newly created transaction:

[KR.onSubmit()](/en/rest/V4.0/javascript/features/reference.html#kronsubmit): when an accepted transaction is created.[KR.onError()](/en/rest/V4.0/javascript/features/reference.html#kronerror): When a rejected transaction is created.

KR.onTransactionCreated() intercepts all newly created transactions, whether they are accepted or rejected. **KR.onTransactionCreated()** accepts either a callback or a promise as a parameter.

The callback receives an object with 2 parameters:

- KR: Reference to the library - event: Object that contains the newly created transaction.

You can stop the execution string by returning **false** at the end of processing:

| Return value | Behavior | |---|---| | false | execution is interrupted. Error handling or redirection does not take place. | | true | The execution continues normally when the callback is executed. |

KR.on3dSecureAbort()

During 3D-Secure authentication, the user can choose to cancel. This will cause the 3DS-Secure popIn to close and an abort error. This event will intercept the user's abanbon.

Note that in this case the transaction is not created in real time. A rejected transaction with a 149 error will be created asynchronously, and only if the user has not made a new payment attempt. The creation will take place after a maximum of 10 minutes. To intercept this transaction, you must use IPNs. See [General about IPN](/en/rest/V4.0/api/kb/ipn.html) for more details.

Methods

Several methods are available for interacting with the JavaScript client:

| Method | Description | |---|---| |

[KR.setFormConfig()](/en/rest/V4.0/javascript/features/reference.html#setFormConfig)[KR.setBrand()](/en/rest/V4.0/javascript/features/reference.html#krsetbrand)[KR.validateForm()](/en/rest/V4.0/javascript/features/reference.html#validateForm)The following methods are obsolete and are no longer supported. **They should not be used** :.

- KR.validate(): use KR.validateForm() - KR.registerPlugin()

Gestion de l'affichage en mode PopIn (voir undefined)

| Method | Description | |---|---| | KR.closePopin() | Closes the pop-in (if open). | | KR.openPopin() | Opens the pop-in (if closed). | | KR.setShopName() | Changes the name of the shop defined in the pop-in header. |

Dynamic form management (add, remove DOM):

| Method | Description | |---|---| | KR.addForm(CSS class or id) | Adds a form to a DOM element. Returns a formId. | | KR.attachForm(CSS class or id) | Enable the form on an existing DOM. Returns a formId. | | KR.hideForm(formId) | Hide the form | | KR.removeEventCallbacks() | Deletes all previously attached callbacks using KR.on[*] functions. | | KR.removeForms() | Removes all forms from the DOM (automatically calls KR.removeEventCallbacks()) | | KR.showForm(formId) | Displaying a form |

you can see [the Embedded Form Glue GitHub repository](https://github.com/lyra/embedded-form-glue) for more information.

Managing the payment form submission button:

| PARAMETER | Description | |---|---| | KR.button.setLabel('MON LABEL %amount-and-currency% ') | Defines a label where %amount-and-currency% will be replaced by the amount and currency | | KR.button.showSpinner() | Displays the waiting animation | | KR.button.hideSpinner() | Hides the waiting animation | | KR.button.disable() | Disables the button (not clickable) | | KR.button.enable() | Activate the button |

All events return promises, allowing you to integrate them into a chain. See [Working in asynchronous environment](/en/rest/V4.0/javascript/spa/index_smart.html#promises) for more information.

KR.field.focus()

To give focus to a form field, you can use the **KR.field.focus(FIELD_CLASS)** method. You must pass the class of the embedded form field as a parameter.

For example, to add a button that will set focus on the expiry date field:

<button type="button" onclick="KR.fields.focus('kr-expiry')">Focus expiry field</button>

KR.setFormConfig()

This method allows to override [the initialization parameters](/en/rest/V4.0/javascript/features/reference.html#initParams) , as well as the following elements:

| Use | Description | |---|---| | KR.setFormConfig({formToken: "NEW_FORM_TOKEN"}); | Changes the current formToken. | | KR.setFormConfig({language: "fr-FR"}); | Changes the language of the payment form and error messages. | | KR.setFormConfig({'kr-label-do-register': 'custom'}) | Sets the label of the "Save my card" checkbox |

This method returns a promise.

KR.setBrand()

Forces the detection of the payment method.The method takes as parameter the name of the payment method. The supported values are (not exhaustive):.

- VISA - VISA_DEBIT - VISA_ELECTRON - MASTERCARD - MASTERCARD_DEBIT - MAESTRO - AMEX - CB - (...)

Caution: If you set another value, it will be sent unchanged to the payment platform. If the value is not supported by your store configuration, no transaction will be created.

To re-enable automatic detection, call **KR.setBrand()** without any parameters.

KR.validateForm()

This method checks locally if the form is valid. It returns a promise:.

then()is called when the form is valid.**result**will be set to null.**catch()**is called when the form is invalid.**result**contains the details of the error.

Example of a call:

KR.validateForm().then(({KR, result}) => { /* there is no error */ /* result == null */ }) .catch(({KR, result}) => { /* Get the error message */ var code = result.errorCode; var message = result.errorMessage; var myMessage = code + ": " + message; console.log(myMessage); /* if you have defined a callback using */ /* result.onError(), you can trigger it calling: */ return result.doOnError(); });

Once you have intercepted the errors, you can trigger the KR.onError() event manually by calling `result.doOnError();`

.

Button personalization

The payment button is automatically added to your form using the following code:

There are numerous benefits to using our payment button:

- The labels are translated automatically. - The amount is automatically formatted and updated. - We handle the pending transition animation for you - The button automatically changes to read-only mode if necessary.

If you want to manage the button label yourself, all you need to do is add it as follows:

<button class="kr-payment-button">Custom label</button>

You can also insert a variable that represents the amount and currency. The JavaScript client will automatically replace them:

<button class="kr-payment-button">this will cost %amount-and-currency% !!</button>

Frequently Asked Questions

Here you will find the FAQ about the JavaScript client.

How to correct CORS or "Unable to post message" errors upon payment?

First of all, you must develop from a web server. The access to the test page must be done with **http://** and not with **file://**

If you use the ** ionic** or

cordovaframework, you must explicitly declare the domain names that can be called by the JavaScript client. To do this, add the folowing in

config.xml:

`<allow-navigation href="https://static.lyra.com/static/js/krypton-client/V4.0" />`

More information [here](https://cordova.apache.org/docs/en/latest/guide/appdev/allowlist/index.html).

How to configure the CSP (Content Security Policy)

If your website uses CSP (Content Security Policy), you must allow the JavaScript client to create its IFrames. To do this, you must add the following 3 directives:

| CSP directive | Values | |---|---| | connect-src | https://static.lyra.com | | frame-src | https://static.lyra.com | | script-src | https://static.lyra.com |

Example of adding your page head to the meta section:

```
Code Example: <meta http-equiv="Content-Security-Policy" content="
connect-src https://static.lyra.com; frame-src https://static.lyra.com;
script-src https://static.lyra.com;" />
```

If you use an external fraud detection engine (such as monitor+, clearsale, etc.), you should also add:

| CSP directive | Values | |---|---| | connect-src | https://secure.lyra.com | | frame-src | https://secure.lyra.com | | script-src | https://secure.lyra.com |

Advanced use

To see all the advanced use cases, go to the following articles:

| Description | Article | |---|---| | Transmit the fields based on your needs | |

[Additional fields](/en/rest/V4.0/javascript/features/custom_fields.html)[Error handling](/en/rest/V4.0/javascript/features/js_error_management.html)[Handling retries](/en/rest/V4.0/javascript/features/retry.html)[General settings](/en/rest/V4.0/javascript/features/general_parameter.html)[Code samples](/en/rest/V4.0/javascript/guide/example/code_smartForm.html)## Examples of integration

Depending on the JavaScript framework that you use on your merchant website, several examples of integration are available here

| Framework | Description | |---|---| |

Step 6: Shift in Production

Source: https://docs.lyra.com/en/rest/V4.0/javascript/guide/embedded/testing.html

I. Performing mandatory tests

It is essential to perform the mandatory tests in order to go to PRODUCTION.

Tests must be performed with:

- Test password - Public test key - HMAC-SHA-256 test key

The list of tests to be performed is displayed in the **Settings** > **Shop** > **REST API keys**.

- You must simulate several payments using the test cards in the table below, using the debug bar. - Using one card on each line is sufficient. - The test card numbers are available from the debug bar, **Test cards**tab.

- When the test is validated, the icon color of the **"test status" column**changes to green. - Once the 4 tests have been validated, the **Generate the password and the HMAC-SHA-256 production key**button becomes available.

II. Generate the PRODUCTION keys

Clickthe**"Generate the password and the HMAC-SHA-256 production key"**button. Note that the TEST mode is still available, even after the production key is generated.**Insert**the PRODUCTION keys into your implementation:

- The production password. - The public production key. - The HMAC-SHA-256 production key for calculating the signature contained in the kr-hash field.

Setthe notification URL correctly at the end of the payment in PRODUCTION mode from the**Setting**>**Notification Rules**menu. (link[Set up notification URL](/en/rest/V4.0/javascript/guide/notification/ipn.html)).**Store the keys**.

We recommend you to store the information (password, public key, HMAC-SHA-256 production key) securely because as soon as the first payment is made in PRODUCTION mode, it will be marked in theMerchant Back Office .

III. Making the first production payment

- Make an actual transaction of at least 2 €.

. This transaction can subsequently be canceled from the **Management** > **Transactions** > **Current Transactions** tab. This transaction will therefore not be remitted to the bank.

- Make sure the notification URL at the end of payment (IPN) is working

Display the transaction details in the **Sent**.

IV. Simplified example in PHP

The example file is given as an indication, in order to understand the integration mechanism.

Renseignez les clés d'API REST de Test dans le fichier

`config.php`

(lien[Fichier d'exemple : config.php](/en/rest/V4.0/javascript/guide/keys/file.html)).Si besoin, modifiez les données du paiement dans le fichier

`formToken.php`

(lien[Fichier d'exemple : formToken.php](/en/rest/V4.0/javascript/guide/formToken/file.html)).Display the payment form in a browser according to the desired display mode:

`embedded.php`

([Sample files: embedded.php and popin.php](/en/rest/V4.0/javascript/guide/embedded/file.html))`popin.php`

([Sample files: embedded.php and popin.php](/en/rest/V4.0/javascript/guide/embedded/file.html)

Once the form is displayed, click the Test cards tab of the debug bar and select the type of card you want to use.

See the test list in the

Merchant Back Office to identify the card number that will be used.When a test is validated, its status is updated in the list. Use the

Refresh Tablebutton if the status did not refresh automatically.Once the 4 tests have been validated, the

Generate the password and the HMAC-SHA-256 production keybutton becomes available.Click

Generate the password and the HMAC-SHA-256 production keyand accept the various warning messages.

Adding custom fields to your form

Source: https://docs.lyra.com/en/rest/V4.0/javascript/features/custom_fields.html

In order to insert a custom field to the payment form, all you need to do is add standard HTML tags to the **kr-embedded** div:

This sample code will insert:

- The **e-mail**field must be the 1st field of the form. - A mandatory checkbox at the end of the form. - a hidden field

Once the payment is made, the custom fields will be added in the POST parameters transmitted to the Merchant:

{ "kr-hash": "a8bbe12b1562696677a2bd93ce04d676e4aba47e538cb97abb8b8ad2b418cb08", "kr-hash-algorithm": "sha256_hmac", "kr-answer-type": "V4\/Payment", "kr-answer": "{...}", "acme-email": "hello@email.acme", "acme-terms": "true", "acme-hidden": "my hidden value" }

HTML5 validation

You can add HTML5 validation directives to your custom field:

Validation errors are processed in the same way as form field errors:

- Validations will be applied locally. - The errors will appear in the error area of the form. - The error handling callback defined via **KR.onError()**is taken into account. - The theme will be applied automatically.

Currently, the supported HTML5 directives are:

| Directive | Description | |---|---| | required | This field is mandatory. |

When a mandatory field is submitted empty, the JavaScript library raises a **CLIENT_304** type error.

Themes and CSS

Additional fields use default CSS on your website. It is possible to automatically apply the payment form theme by adding the **kr-theme** class to the element:

Adding **kr-theme** allows to automatically apply:

- The current theme of the payment form, - Error handling (animations, colors) for the field.

Custom icons

You can add custom icons in additional fields using the **kr-icon** parameter:

Icons from [Font-Awesome](https://fontawesome.com/) are also supported. You must first load the library in the **<head>** section of your page:

Other libraries will be added in upcoming versions.

Tabbing

You can also set the tab order by adding the **kr-tab-order** parameter:

<div class="kr-embedded" kr-form-token="<?php echo $formToken;?>"> <!-- custom fields --> <input type="text" name="acme-email" placeholder="email" class="kr-theme" kr-icon="fas fa-envelope" kr-tab-order="1" required/> <!-- payment form fields -->   <!-- payment form submit button --> <button class="kr-payment-button"></button> <!-- error zone -->

This parameter is only necessary when an additional field is present.

Default value.

You can also set the default value by adding the **value** html attribute:

<div class="kr-embedded" kr-form-token="<?php echo $formToken;?>"> <!-- custom fields --> <input type="text" name="acme-email" placeholder="email" class="kr-theme" kr-icon="fas fa-envelope" value="mail@example.com" required/> <!-- payment form fields -->   <!-- payment form submit button --> <button class="kr-payment-button"></button> <!-- error zone -->

Managing errors (JS client)

Source: https://docs.lyra.com/en/rest/V4.0/javascript/features/js_error_management.html

This page describes the processing of integration errors.

If you would like to customize how you handle errors that occur when a transaction is rejected, see the following article:

How are errors displayed?

For JavaScript form, errors are automatically displayed in the **kr-form-error** div:

All you need to do is add it to your code.

Managing errors manually

To handle errors manually, simply intercept the default error message and customize it. In the following example, the custom error message will be placed in the **customerror** class div:

JS error codes

The REST API has many error codes described here:

The error codes that start with **CLIENT_** correspond to errors occurring in the Buyer's browser. They occur before the payment data is sent to our servers. This concerns, for example, local form data validation errors or JavaScript client integration errors.

| CODE | Definition | |---|---| |

[CLIENT_100](/en/rest/V4.0/javascript/features/js_error_management.html#client100)[CLIENT_101](/en/rest/V4.0/javascript/features/js_error_management.html#client101)[CLIENT_300](/en/rest/V4.0/javascript/features/js_error_management.html#client300)[CLIENT_301](/en/rest/V4.0/javascript/features/js_error_management.html#client301)[CLIENT_302](/en/rest/V4.0/javascript/features/js_error_management.html#client302)[CLIENT_303](/en/rest/V4.0/javascript/features/js_error_management.html#client303)[CLIENT_304](/en/rest/V4.0/javascript/features/js_error_management.html#client304)[CLIENT_305](/en/rest/V4.0/javascript/features/js_error_management.html#client305)[CLIENT_500](/en/rest/V4.0/javascript/features/js_error_management.html#client500)[CLIENT_501](/en/rest/V4.0/javascript/features/js_error_management.html#client501)[CLIENT_502](/en/rest/V4.0/javascript/features/js_error_management.html#client502)[CLIENT_505](/en/rest/V4.0/javascript/features/js_error_management.html#client505)[CLIENT_508](/en/rest/V4.0/javascript/features/js_error_management.html#client508)[CLIENT_997](/en/rest/V4.0/javascript/features/js_error_management.html#client997)[CLIENT_998](/en/rest/V4.0/javascript/features/js_error_management.html#client998)[CLIENT_999](/en/rest/V4.0/javascript/features/j

s_error_management.html#client999)## Warning codes

The error codes comprised between **CLIENT_700** and **CLIENT_799** are warning messages. They help you integrate the JavaScript client:

| CODE | Definition | |---|---| |

[CLIENT_705](/en/rest/V4.0/javascript/features/js_error_management.html#client705)## Error code details

CLIENT_004

Code | CLIENT_004 | DEFINITION | Invalid public key | Category | Errors |

The public key defined in **kr-public-key** is not valid, it must have the following format: **[NUMBER]:[STRING]**

for example: **69876357:testpublickey_DEMOPUBLICKEY95me92597fd28tGD4r5**

For more information, go to: [Retrieving my keys](/en/rest/V4.0/api/get_my_keys.html).

CLIENT_100

Code | CLIENT_100 | DEFINITION | Invalid formToken | Category | Errors |

The formToken that you have defined in **kr-form-token** is invalid. For more information on creating a formToken, go to: [Integration guide](/en/rest/V4.0/javascript/guide/start.html).

CLIENT_101

Code | CLIENT_101 | DEFINITION | Aborted | Category | Errors |

Transaction abandoned by the buyer. This error occurs, for example, when the Buyer closes the 3D Secure pop-in before authenticating him/herself.

If the Buyer does not make another attempt, a rejected transaction is automatically created when the formToken expires.

CLIENT_300

Code | CLIENT_300 | DEFINITION | Invalid form data | Category | Errors |

When several form fields are invalid, a general CLIENT_300 error is returned. The detailed list of all detected errors will be presented in the **children** field:

{ "errorCode": "CLIENT_300", "errorMessage": "Invalid form data", "children": [{ "errorCode": "CLIENT_301", "errorMessage": "Invalid card number", "field": "pan", (...) }, { "errorCode": "CLIENT_302", "errorMessage": "Invalid expiry date", "field": "expiryDate", (...) }, { "errorCode":

"CLIENT_303", "errorMessage": "Invalid security code", "field": "securityCode", (...) }],
"detailedErrorCode": null, "detailedErrorMessage": null, (...) }

CLIENT_301

Code | CLIENT_301 | DEFINITION | Invalid card number | Category | Errors |

The **kr-pan** field (card number) of the payment form is invalid.

CLIENT_302

Code | CLIENT_302 | DEFINITION | Invalid expiry date | Category | Errors |

The **kr-expiry** field (expiry date) of the payment form is invalid.

CLIENT_303

Code | CLIENT_303 | DEFINITION | Invalid security code | Category | Errors |

The **kr-security-code** field (security number or CVV) of the payment form is invalid.

CLIENT_304

Code | CLIENT_304 | DEFINITION | value is required | Category | Errors |

An additional field declared as mandatory is empty. For more information, go to: [Custom form fields](/en/rest/V4.0/javascript/features/custom_fields.html).

CLIENT_305

Code | CLIENT_305 | DEFINITION | No formToken defined | Category | Errors |

Le **formToken** n'existe pas ou n'est pas conforme. Pour plus de détails, rendez-vous ici : undefined.

CLIENT_500

Code | CLIENT_500 | DEFINITION | No form or button defined | Category | Errors |

No payment form buttons have been found in the HTML code. Check if the required tags are present, or if there are no syntax errors.

CLIENT_501

Code | CLIENT_501 | DEFINITION | Kr-public-key is empty or not defined | Category | Errors |

The public key is not defined in **kr-public-key**.

CLIENT_502

Code | CLIENT_502 | DEFINITION | Form already submitted (browser back button not supported) | Category | Errors |

The application has detected that the Buyer returned to the payment page using the return button in their Internet browser. The payment form has been locked.

Note that this information can be detected only in some browsers.

CLIENT_505

Code | CLIENT_505 | DEFINITION | SmartForm not supported with the current theme | Category | Errors |

Le thème`material`

n'est pas supporté par le formulaire**smartForm**.

Il faut**impérativement**changer et choisir entre le thème`néon`

ou le thème`classic`

. Voir :"[Themes](/en/rest/V4.0/javascript/redirection/themes.html)".

CLIENT_508

Code | CLIENT_508 | DEFINITION | Card payment is not available | Category | Errors |

Make sure you have an active card payment contract that is associated with your shop.

CLIENT_704

Code | CLIENT_704 | DEFINITION | you need to load Font Awesome in your | Category | Warnings |

An additional field uses icons with the Awesome font but the library has not loaded. For more information, go to: [Custom form fields](/en/rest/V4.0/javascript/features/custom_fields.html).

CLIENT_705

Code | CLIENT_705 | DEFINITION | viewport definition is missing (| Category | Warnings |

The `<meta>`

tag, via the "viewport" directive tells the browser how to control the dimensions and scale of the page to be displayed. It is recommended to use it on all HTML5 pages.

CLIENT_997

Code | CLIENT_997 | DEFINITION | Endpoint configuration mismatch | Category | Errors |

The formToken has been created via a gateway different from the one where the JavaScript client has been downloaded. The URL of the call to the REST Web Service must be the same as the one of the JavaScript client.

CLIENT_998

Code | CLIENT_998 | DEFINITION | The used FormToken is for demo purposes only, payment is disabled. Check the documentation to get a real formToken. | Category | Errors |

The payment form uses a demo formToken that does not allow to interact with the server. Use a valid formToken: [Charge/CreatePayment](/en/rest/V4.0/api/playground/?ws=Charge%2FCreatePayment).

CLIENT_999

Code | CLIENT_999 | DEFINITION | Technical error | Category | Errors |

Unknown error, please contact tech support and give them the following information:

- Shop number - URL of the form - Name and version of the browser - Type and version of the operating system - Device used (iPhone 6S, PC, iPad Pro, etc.) - Date and time of the error.

New payment attempt (retry)

Source: https://docs.lyra.com/en/rest/V4.0/javascript/features/retry.html

Some orders are not accepted simply because the Buyer makes a mistake when entering their card number. In this case, the payment is rejected, and very often the Buyer abandons their cart. This situation is detrimental for the Merchant as he or she loses the Buyer's order.

In case of rejected payment, the retry option allows the Merchant to immediately offer their client to redo the payment on the same page, which simplifies the Buyer's journey and significantly increases the conversion rate.

By default, the retry is set to "1", which means that the Buyer can make 2 payment attempts.

When the Buyer reaches the maximum number of authorized payment attempts, a message appears on the page to inform them about it and the fields can no longer be filled.

Le marchand peut aussi choisir de rediriger l'acheteur vers une page spécifique. Pour cela, dans les paramètres d'initialisation du Client JavaScript, il suffit de valoriser le champ **kr-post-url-refused** avec l'URL de la page de retour à la boutique souhaitée.

How to change the number of authorized payment attempts?

{ "amount": 990, "currency": "EUR", "transactionOptions": { "cardOptions": { "retry": 1 } }

Code samples

Source: https://docs.lyra.com/en/rest/V4.0/javascript/guide/example/code_smartForm.html

'sample' file in GitHub

For each step of the integration, we provide a simplified example **in PHP** (link from GitHub: [sample folder](https://github.com/lyra/rest-php-examples/tree/master/www/sample))

Renseignez les clés d'API REST de Test ou de Production dans le fichier

`config.php`

([Fichier d'exemple : config.php](/en/rest/V4.0/javascript/guide/keys/file.html)).Si besoin, modifiez les données du paiement dans le fichier

`formToken.php`

([Fichier d'exemple : formToken.php](/en/rest/V4.0/javascript/guide/formToken/file.html)).Display the payment form in a browser according to the desired display mode:

`smartForm.php`

([Example file: smartForm.php](/en/rest/V4.0/javascript/guide/display/file.html))

Once the form is displayed, proceed with the payment. In Test mode, click on the Test cards tab of the debug bar and select the card type.

Récupérez les données de l'IPN, lors de l'appel de serveur à serveur grâce au fichier

`ipn.php`

([Fichiers d'exemple : ipn.php et paid.php](/en/rest/V4.0/javascript/guide/analyse/file.html)).Redirigez l'acheteur en cas de paiement réussi vers le site marchand grâce au fichier

`paid.php`

([Fichiers d'exemple : ipn.php et paid.php](/en/rest/V4.0/javascript/guide/analyse/file.html)).

Another example in PHP

Here is a link for another [Example PHP code](https://github.com/lyra/rest-php-examples/tree/master/www/smartform).

Integration examples: C# / Python (Flask) / Symfony

| Framework | Description | |---|---| |

Getting started: single payment

Source: https://docs.lyra.com/en/rest/V4.0/javascript/guide/start.html

Three steps are required for integrating a payment form on the merchant website:

- Initialize the embedded form - Display the embedded form - Check the transaction status and the end of payment

The following diagram presents the interactions between the different actors (embedded form, merchant server, payment gateway) at each of these stages:

Merchant server

Formulaire JavaScript (Navigateur)

Payment gateway server

1. Initializing the form

Before displaying a new payment form on the merchant website, you must initialize the form by generating a **formToken** that summarizes all the payment related options.

In order to create a formToken, you must call the [Charge/CreatePayment](/en/rest/V4.0/api/playground?ws=Charge%2FCreatePayment) REST Web Service via your server:

| Step | Description | |---|---| | 1 | Call to the |

formToken.The formToken is valid for 15 minutes.

See the list of [use cases](/en/rest/V4.0/javascript/guide/features.html) that will help you initialize your form.

2. Displaying the form

To display the payment form, you must include our JavaScript library **in the header of your payment page**.

It is imperative for the main library to be loaded very early on, well before the other JS libraries used on your page :.

The **formToken** is to be added to the integration code. It will be used by the JavaScript library to display the form using the formToken defined in the previous step:

| Step | Description | |---|---| | 3 | Download of the JavaScript library. Request made via the Buyer's browser. | | 4 | Retrieval of JavaScript, then display of the page containing the payment form. |

Once the form is displayed, the buyer can enter their bank details. If the payment is rejected, the buyer stays on the merchant website. If the payment is accepted, the transaction details are sent to the merchant server.

3. Checking the transaction status

When the transaction is accepted or the maximum number of attempts has been reached, the JavaScript client POSTs the payment form. It is made in exactly the same way as for a classic HTML form. You retrieve the payment information in the POST parameters sent to your server.

| Step | DESCRIPTION | |---|---| | 5 | The Buyer has clicked the “pay” button: submission of the form via the Buyer’s browser to our servers. This call is made automatically by our JavaScript client. | | 6 | Once the transaction has been processed, we make a call via our servers to the URL specified by you. The complete Transaction object will be sent to allow you to update your information system before the return to the shop. The IPN (Instant Payment Notification). | | 7 | Our servers return the payment result to the JavaScript client. | | 8 | The JavaScript client POSTs the payment form to your servers. |

Processing the IPN allows you to update your information system more securely and guarantees that you will not lose any payments if the Buyer loses their Internet connection. For more information, see the article [IPN: Presentation](/en/rest/V4.0/api/kb/ipn.html).

Let’s get started

Après cet entremet théorique, il est temps de passer aux choses concrètes : undefined

404 Error...

Source: https://docs.lyra.com/en/rest/V4.0/javascript/spa/index_smart.html

Sorry, but the page you were trying to view does not exist — perhaps you can try searching for it. It looks like this was the result of either:

- A mistyped address - An out-of-date link [A VADS payment error](/en-EN/form-payment/standard-payment/vads-payment-error.html)

[Back to english homepage](/en-EN/)

Page introuvable

Impossible de trouver la page que vous essayez d'atteindre. Une raison possible est :

- La page n’existe pas. - La page existe mais vous n’êtes pas autorisé à l’afficher. [une erreur de paiement VADS](/fr-FR/form-payment/standard-payment/vads-payment-error.html)

[Retour à l'accueil en français.](/fr-FR/)

require.js

Source: https://docs.lyra.com/en/rest/V4.0/javascript/spa/requirejs.html

Require.js allows to load scripts asynchronously. For more information on the functioning of **require.js** , go to: https://requirejs.org

Example of integration

To load the embedded form library with **requirejs** , add the following code in the **<head>** section of your page:

https://github.com/lyra/rest-php-examples/blob/master/www/js_examples/requirejs/minimalEmbeddedForm.php#L53-L77

require.js will asynchronously load the JavaScript client (krypton path) and the classic theme (kryptonTheme path).

Then, in the **body** section, add the payment form:

https://github.com/lyra/rest-php-examples/blob/master/www/js_examples/requirejs/minimalEmbeddedForm.php#L85-L102

Note that the configuration directives (**kr-public-key** , etc.) are defined in a div within the form (**kr-embedded** class), by contrast with classic integration.

Complete example

https://github.com/lyra/rest-php-examples/blob/master/www/js_examples/requirejs/minimalEmbeddedForm.php