

Klarna®

Klarna Payments for SG

Version 21.1.0



[Table of Contents](#)

1. Summary	5
2. Component Overview	6
2.1. Functional Overview	6
2.2. Locales	8
2.3. Use Cases	9
2.3.1. Enable Klarna Payments Across International Sites (NA, EU, OC)	9
2.3.2. Multitude of Payment Options for Customers	9
2.3.3. Authorize/Place Klarna Order in Checkout	9
2.3.4. Refusal of Klarna Payments on Payment Method – Authorization	12
2.3.5. Klarna Payment Option Not Available for Current Purchase - Billing Page	13
2.3.6. Klarna Payments Not Available - Checkout	14
2.3.7. Handling Notifications	14
2.3.8. Virtual Cards Settlements	16
2.3.9. Auto-Capture	20
2.3.10. Widget Customizations	21
2.3.11. Customizing Payment Method Name	22
2.3.12. Klarna On-Site Messaging	23
2.3.13. Klarna Payment Method Based Promotions	27
2.3.14. Price Adjustment Taxation Handling	27
2.3.15. Buy Online, Pickup in Store (BOPIS)	28
2.4. Compatibility	29
2.5. Privacy, Payment	30
2.5.1. GDPR Compliance	30
2.5.2. EMD (Extra Merchant Data)	30
2.5.3. PCI-DSS Compliance	33
3. Implementation Guide	34
3.1. Setup of Business Manager	34
3.1.1. Cartridge Upload & Assignment	34
3.1.2. Metadata Import	35
3.2. Configuration	36
3.3. Template Updates	38
3.4. Jobs	38

3.4.1.	Job “OrderCleanUp”	38
3.5.	<i>Custom Code</i>	42
3.5.1.	Template Modifications	42
3.5.1.1.	default/checkout/summary/summary.isml	43
3.5.1.2.	default/checkout/billing/billing.isml	44
3.5.1.3.	default/checkout/billing/paymentmethods.isml	45
3.5.1.4.	default/checkout/shipping/minishipments.isml	46
3.5.1.5.	default/components/footer/footer_UI.isml	47
3.5.1.6.	default/components/footer/footer.isml	48
3.5.1.7.	default/product/producttopcontentPS.isml	49
3.5.1.8.	default/product/productcontent.isml	49
3.5.1.9.	js/pages/product/variant.js	49
3.5.1.10.	default/checkout/cart/cart.isml	50
3.5.1.11.	default/components/header/header.isml	50
3.5.1.12.	default/mail/orderconfirmation.isml	51
3.5.1.13.	default/components/order/ordrdetailsemail.isml	52
3.5.2.	Controller Modification	53
3.5.2.1.	COBilling.js	53
3.5.2.2.	COSummary.js	55
3.5.2.3.	OrderModel.js	55
3.6.	<i>External Interfaces</i>	56
4.	Testing	56
5.	Operations, Maintenance	57
5.1.	<i>Data Storage</i>	57
5.1.1.	System Object Extensions	57
5.1.1.1.	Order	57
5.1.1.2.	Order Payment Instrument	58
5.1.1.3.	Payment Transaction	58
5.1.1.4.	Site Preferences	58
5.1.2.	Custom Objects	61
5.1.2.1.	KlarnaCountries	61
5.1.3.	Library	63
5.1.4.	Services	64

5.2.	<i>Logs</i>	64
5.3.	<i>Availability</i>	64
5.4.	<i>Failover/Recovery Process</i>	64
5.5.	<i>Support</i>	64
5.5.1.	Customer Service	64
5.5.2.	Merchant Support	65
6.	User Guide	66
6.1.	<i>Roles, Responsibilities</i>	66
6.2.	<i>Storefront Functionality</i>	66
7.	Known Issues	70
8.	Release History	71
9.	Additional Information	72
9.1.	<i>Klarna API Information</i>	72
9.1.1.	Live Environment	72
9.1.2.	Testing Environment	72
9.2.	<i>Generate Key Pair and Key Id for Virtual Card Settlements (VCN)</i>	72
9.3.	<i>Decrypt VCN Card Details</i>	74

1. Summary

The **Klarna Payments SG cartridge** enables integration of Klarna Payment solution on Commerce Cloud Storefront. The integration provides merchants the flexibility to offer choice of multiple Klarna Payment products on the Commerce Cloud checkout.

This document contains the instructions for a developer to install the cartridge and integrate it on the Salesforce Commerce Cloud site. The cartridge is fully compatible with the SiteGenesis JavaScript Controllers (SGJS).

Merchant teams are required also to configure the cartridge with the valid merchant credentials and site configurations in Commerce Cloud Business Manager to enable Klarna payments methods in the checkout.

The integration consists of an archive, which contains the following contents:

- Cartridge called “int_klarna_payments” and “int_klarna_payments_controllers” to be imported.
- A site-template archive containing new attributes and settings.
- This document for Site Genesis (Klarna Payments Integration Guide).
- The integration is based on the SiteGenesis demo store provided by Commerce Cloud.

It is a requirement that Merchant sign a contract for integration support and production go-live with Klarna.

Klarna offers a playground (test) environment, so the integration can be tested before switching to the Klarna production environment. Based on the contract, Klarna shall provide assistance with integration and testing prior to sign-off for go-live.

2. Component Overview

2.1. Functional Overview

Key Features:

- Integrate Klarna Payments using best practices on international sites based (markets in North America, Europe, Oceania)
- Enable multiple payment products for customer in Pay Now, Pay Later and Pay Over Time categories
- Fast integration/go-live with virtual card-based integration approach
- Handle Notification: pending status updates (reject/accept) for suspected orders post review
- Site managers can customize the Klarna Payments widget styling displayed in checkout, to match the style guide of merchant website(s)
- GDPR (EU) compliant checkout flow
- Multi Shipping Address support
- Support Klarna authorize+finalize for Bank Transfer methods (Pay Now)
- Enable Onsite Messaging placements on PDP, Cart, Header, Footer, and dedicated Info page
- BOPIS (Buy Now, Pay in Store) support including extra merchant data
- Support for Klarna Payment Method based promotions
- Support for adjusted price promotions with Gross Tax Policy
- Support disabling Payment method for authorization rejection
- Support Auto-Capture

Klarna Payment cartridge makes use of the Klarna Payments JSON REST API and a JavaScript SDK to integrate on the storefront. Klarna Payment enables consumers to choose from the different payment method products offered by Klarna. Multiple Klarna products are available within the categories “Pay Now”, “Pay Later” and “Pay Over Time”. The cartridge integration displays payment options via a widget (iframe) added inline on the billing page, referred to as Klarna widget or just “the widget”. The

widget with information about the payment method is displayed to the customer when the individual clicks on the Klarna payment method.

Customers can authorize the payment after reviewing the payment method terms and clicking Place Order button. The Klarna order is confirmed once order is placed and customer re-directed to the confirmation page.

Orders successfully placed with Klarna return a Fraud Status: ACCEPTED and displayed in Business manager (BM).

With ACCEPTED status, order creation in SCC proceeds as usual. Klarna payment status is saved in a custom attribute with id kpFraudStatus in the PaymentTransaction system object, and can be seen in BM on the order details Payment tab as below:

The screenshot displays the 'Payment' tab for Order '00005332' in the Business Manager interface. The breadcrumb trail at the top reads: Merchant Tools > Ordering > Orders > Order: 00005332(SiteGenesisGlobal). The 'Payment' tab is selected, showing the following details:

Order Total:	£195.83
Amount Paid:	£0.00
Balance Due:	£195.83
Invoice Number:	00024518
Payment Status:	Paid
Payment Method:	Klarna Processor: KLARNA_PAYMENTS Transaction: 8aa32699-20ac-2f76-a5ee-1e554e6cc7cd Amount: £195.83 Klarna Payment Category ID: pay_over_time Klarna Payment Category Name: Buy now, pay later Fraud Status: ACCEPTED

At the bottom left, there is a button labeled '<< Back to List'.

Figure 1 Klarna Payment Details in BM

An alternate flow when PENDING status is returned for Klarna order creation, the SCC order creation proceeds with modified statuses. If later a Klarna notification with updated fraud status FRAUD_RISK_ACCEPTED is returned, SCC order status is updated and returns to the usual flow. Klarna

payment status is saved in a custom attribute with id kpFraudStatus in the PaymentTransaction system object, and can be seen in BM on the order details Payment tab as below:

Merchant Tools > Ordering > Orders > Order: 00005339(SiteGenesisGlobal)

General Attributes **Payment** Notes History

Payment Information for Order '00005339'

Order Total:	£62.35
Amount Paid:	£0.00
Balance Due:	£62.35

Invoice Number:	00024525
Payment Status:	Paid

Payment Method:	Klarna
	Processor: KLARNA_PAYMENTS
	Transaction: 62a33ce0-3bde-2657-bdc4-31f0ba994c45
	Amount: £62.35
	Klarna Payment Category ID: pay_over_time
	Klarna Payment Category Name: Buy now, pay later
	Fraud Status: FRAUD_RISK_ACCEPTED

<< Back to List

Figure 2 Klarna Payment Details in BM

2.2. Locales

The cartridge supports most locales including:

- English
- German
- Danish
- Spanish
- Finnish
- French
- Italian
- Dutch

For a list of more [supported](#) locales, contact Klarna.

2.3. Use Cases

2.3.1. Enable Klarna Payments Across International Sites (NA, EU, OC)

Klarna Payments SG can be configured independently on each site by locale.

2.3.2. Multitude of Payment Options for Customers

On the checkout billing step, configured Klarna payment options are dynamically loaded based on customer's cart information and the payment method categories returned for the current Klarna session.

The screen below shows an example of the payment options displayed in the Checkout billing page:

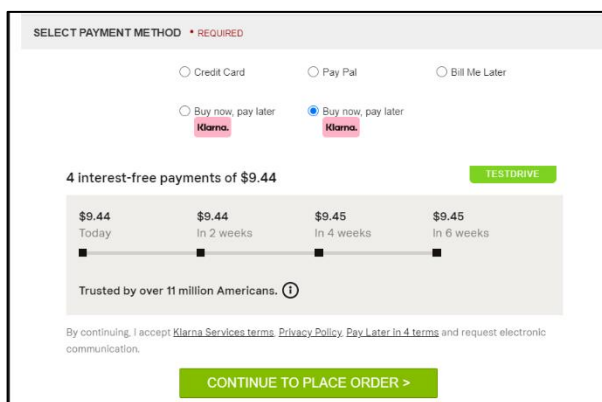


Figure 3 Payment Option

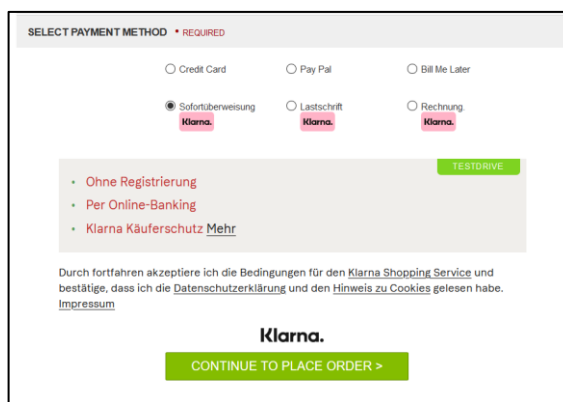


Figure 4 Payment Option

When customer selects (clicks) payment method, a widget with additional information of the Klarna product is displayed as shown above.

Note: The payment methods displayed are based on market and contractual agreement with Klarna.

2.3.3. Authorize/Place Klarna Order in Checkout

Cartridge provides best practice implementation and options to include Extra Merchant Data (EMD) to optimize acceptance rate for Klarna Payment (methods)products. This includes customer info and (Buy Online, Pickup in Store) BOPIS details included as merchant data when custom site preference

“attachments” is enabled for the site. The EMD data sent can be extended but should be reviewed case by case and optimized and validated based on merchant data & privacy requirements prior to go-live.

The customer’s information (personally identifiable information) is sent once customer chooses a Klarna payment method and authorizes (clicks: “Continue to Place Order”). The required customer information facilitates assessment and verification of customer data to display payments method options available for the customer.

When customer clicks “Continue to Place Order” button, the authorization is initiated, and a successful authorization takes customer to summary page. Note that customer may be prompted for additional information prior to completion of authorization based on the payment method selected and market. If the authorization is not successful (“approved = false”), then the customer will stay on the billing page, the “Continue to Place Order” button will be disabled and depending on the Business Manager settings the payment option may be hidden or grayed out (refer to section **2.3.4 Refusal of Klarna Payments on Payment Method – Authorization**).

In some cases, the Pay Now payment category requires additional “finalize” call to be triggered when the customer clicks on the “Place Order” button in the review page. This is needed to ensure that the funds will be transferred only when the customer has finalized the order.

STEP 1: Shipping STEP 2: Billing

PRODUCT

Straight Fit S
Item No.: 883386
Color: Gray
Size: 30

Order Discount: TestOFF

Klarna.

Help? 800-555-0199

Demo Bank PSD2

Ihre Bank (oder Loginmethode)
Demo Bank PSD2

Demo Bank BIAS. Es wird keine Überweisung durchgeführt.

Benutzername
test

Passwort
.....

Ich akzeptiere die Verwendung des Zahlungsinitilierungsdienstes und/oder des Kontoinformationsdienstes von Sofort, damit Sofort eine Zahlung initiieren und/oder auf mein Zahlungskonto zugreifen kann.
Es gelten unsere [Datenschutzhinweise](#)

Verifikation Kontakt Deutsch

Weiter

ORDER SUMMARY Edit

Subtotal	112,00 €
Order Discount	- 11,20 €
Edit Shipping Ground	7,99 €
Order Total:	108,79 €

SHIPPING ADDRESS Edit

Testperson-de Approved
Vogelstundeich 123
Hamburg, HH 21107
Germany
Method: Ground

BILLING ADDRESS Edit

Testperson-de Approved
Vogelstundeich 123
Hamburg, HH 21107
Germany

PAYMENT METHOD Edit

Rechnung und Ratenkauf
Amount: 108,79 €

Figure 5 Finalize call screen

The Klarna Order is only placed prior to final step before SFCC order is created.

Successful payment authorizations (Klarna Payments authorization status: APPROVED) followed by placement of Klarna and SCC orders leads to a “Paid” order payment status and “Ready for Export” export status. Refused and pending Klarna Payment orders (Klarna Payments order statuses REJECTED and PENDING) lead to a “Not Paid” order payment status, and “Not Exported” export status.

Cancelled orders are set to a “Not Paid” order payment status (even if the status was “Paid” before), and “Not Exported” export status.

When customers have selected Klarna payment option as a payment method for the order, successfully authorized the amount on order and later return to the billing page to choose a different payment method that is non-Klarna payment method. In such scenarios, when customer authorizes (non-Klarna payment method) and reaches the review page – automatic “**cancelAuthorization**” call will be triggered to release the authorized funds (Klarna related) & free up the available purchase amount for this customer.

Merchants have the option to utilize this function to cancel prior authorization when required for specific use-cases apart from the above scenario. If enabled, the checkout flow should be tested thoroughly as part of integration, considering the valid checkout scenarios (e.g., relevant Klarna session flow, Klarna payment method switching, order amount updates, checkout with external payment method, etc.).

```
379 /**
380  * Saves/Updates Klarna Payments authorization token in the current session
381  */
382  * @return {void}
383  */
384 function saveAuth() {
385  // Cancel any previous authorizations
386  // cancelAuthorization();
387 }
388 Transaction.wrap( function() {
389  session.privacy.KlarnaPaymentsAuthorizationToken = request.headers['x-auth'];
390  session.privacy.KlarnaPaymentsFinalizeRequired = request.headers['finalize-required'] === 'true';
391  });
392 }
393 response.setStatus( 200 );
394 }
395 }
396 /**
397  * Deletes the previous authorization
398  * @param {string} authToken Authorization Token
399  * @return {string} Service call result
400  */
401 function cancelAuthorization( authToken ) {
402  var klarnaAuthorizationToken = authToken || session.privacy.KlarnaPaymentsAuthorizationToken;
```

Figure 6 Cancel Authorization Call

2.3.4. Refusal of Klarna Payments on Payment Method – Authorization

Upon selecting one of Klarna's options as the payment method on billing step of checkout and based on the customer information provided, Klarna Payments can be refused as a payment method.

If the payment method was rejected with “**show_form=false**” & “**approved=false**” (i.e., hard reject), the merchant has the option to choose what happens with the payment option display in Billing page using BM preference “Hide Payment Methods on Deny” (**kpRejectedMethodDisplay**):

No – Leave the payment visible to the customers

Hide – The payment option will be hidden from customer

Grey Out – The payment option will be greyed out and not clickable

Please note that reloading the page will show the denied Klarna payment method again

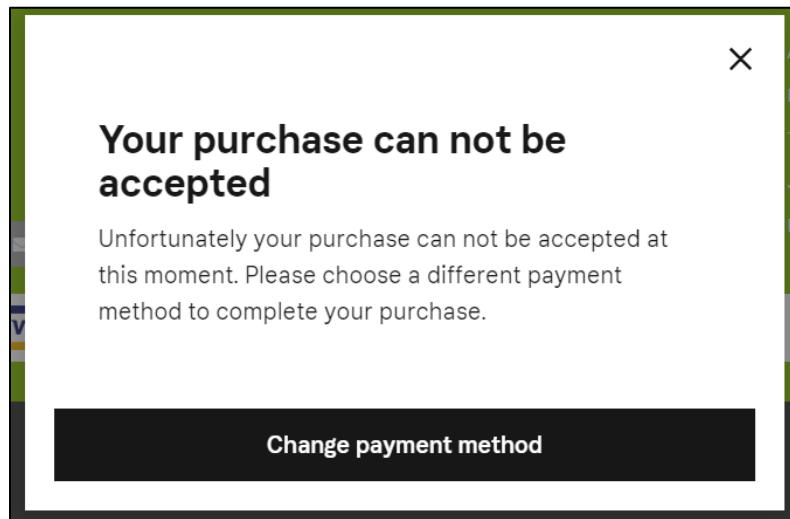


Figure 7 Denied Order Popup

SELECT PAYMENT METHOD • REQUIRED

☐ Credit Card ☐ Pay Pal ☐ Bill Me Later

☒ Buy now, pay later ☐ Buy now, pay later

Klarna. **Klarna.**

Your purchase can not be accepted **TESTDRIVE**

Unfortunately your purchase can not be accepted at this moment. Please choose a different payment method to complete your purchase.

CONTINUE TO PLACE ORDER >

Figure 8 Greyed Out Payment Option

2.3.5. Klarna Payment Option Not Available for Current Purchase - Billing Page

The customer is presented with an appropriate message in the Klarna widget when customer attempts to choose Klarna in the billing page.

SELECT PAYMENT METHOD • REQUIRED

☐ Credit Card ☐ Pay Pal ☐ Bill Me Later

☒ Buy now, pay later ☐ Buy now, pay later

Klarna. **Klarna.**

Option not available **TESTDRIVE**

Unfortunately this option is not available. Please choose a different payment method.

CONTINUE TO PLACE ORDER >

Figure 9 Option Not Available

2.3.6. Klarna Payments Not Available - Checkout

If Klarna API is not available or the site/storefront is not applicable, Klarna is not presented as a payment option in the billing page. It is recommended that Klarna session must not be created when customer chooses a non-Klarna market or merchant store, as well as in cases when merchants have multicurrency storefront with basket currency that is not supported by Klarna.

2.3.7. Handling Notifications

In scenarios where the Klarna Order has been created but instead of immediately accepting the order, the Klarna order is flagged for additional review. This results in Commerce Cloud order staying in "Created" status with Fraud Status: PENDING. This order is marked with EXPORT_STATUS_NOTEXPORTED, confirmation status NOTCONFIRMED and NOTPAID.

For Klarna orders with Fraud Status PENDING, once review is complete, updates are sent to the pre-configured notification_url on the merchant Commerce Cloud site. The updated Fraud status depends on the fraud screening, the returned Fraud Status (e.g., FRAUD_RISK_ACCEPTED) is displayed in BM. The push notification is repeatedly sent (up-to 24 hours, every 10 mins) until the POST request is acknowledged with a 200 response.

Klarna sends one of the following event types in the notification to SFCC to update risk status: FRAUD_RISK_ACCEPTED, FRAUD_RISK_REJECTED, FRAUD_RISK_STOPPED.

The notification updates are generally received within 4-24 hours. The order's payment transaction is updated (see **kpFraudStatus**). This can be seen in BM on the order details Payment tab as below:

[Merchant Tools](#) > [Ordering](#) > [Orders](#) > Order: 00005339(SiteGenesisGlobal)

General Attributes **Payment** Notes History

Payment Information for Order '00005339'

Order Total:	£62.35
Amount Paid:	£0.00
Balance Due:	£62.35

Invoice Number:	00024525
Payment Status:	Paid

Payment Method:	Klarna Processor: KLARNA_PAYMENTS Transaction: 62a33ce0-3bde-2657-bdc4-31f0ba994c45 Amount: £62.35 Klarna Payment Category ID: pay_over_time Klarna Payment Category Name: Buy now, pay later Fraud Status: FRAUD_RISK_ACCEPTED
-----------------	--

<< Back to List

Figure 10 Fraud Status in Payment Details

If the order is “FRAUD_RISK_ACCEPTED” upon notification, the order will be placed in SFCC (order status changes to OPEN). The order is marked with status CONFIRMATION_STATUS_CONFIRMED and export status EXPORT_STATUS_READY. If the order was placed with auto-capture, the payment status will be set to PAYMENT_STATUS_PAID and the full order amount will be captured.

If the order is “FRAUD_RISK_REJECTED” or “FRAUD_RISK_STOPPED” upon notification, the order is failed (FAIL) in SFCC.

Note: The Klarna pending functionality availability is dependent on markets and enabled based on the contractual agreement with Klarna.

2.3.8. Virtual Cards Settlements

This option is disabled by default. However, if standard order management is not a reasonable option for a Klarna integration, then Klarna’s Merchant Card Service API based virtual card solution may be utilized via site preference **“kpVCNEnabled”**:

The screenshot shows the Klarna Merchant Tools interface. At the top, there's a header with the Klarna logo, a dropdown menu showing 'Sandbox - klarna01 SiteGenesisGlobal', and navigation links for 'Merchant Tools', 'Administration', and 'Storefront'. On the right, there are icons for home, user, and notifications, along with a pagination indicator '1-20 of 20'. The main content area is a table of settings. The table has three columns: 'Name', 'Value', and 'Default Value'. The settings listed are: 'Secondary Text Color Preference' (CSS hex color), 'Border Radius Preference' (Size in pixels), 'Attachments' (Flag to switch on/off), 'Not available message on billing page' (JSON string), and 'Virtual Card Network Enabled' (If set to true SFCC will create Virtual Card Network settlement). The 'Virtual Card Network Enabled' setting is highlighted with a red border and has its value set to 'Yes'. Each row has an 'Edit Across Sites' link on the right.

Name	Value	Default Value
Secondary Text Color Preference	CSS hex color to be used in Klarna Payments iFrame	#333333
Border Radius Preference	30	5px
Attachments	Yes	No
Not available message on billing page	{ "GB": "Klarna Payment not available", "FR": "String in French", "AT": "Klarna Rechnung nicht verfügbar", "default": "Klarna Payment not available" }	
Virtual Card Network Enabled	Yes	No

Figure 11 VCN Enablement Setting

The virtual card (see note) issued is for capture of the exact order amount for the given merchant.

When a customer places an order, the order is first booked in SFCC. Once an order has been accepted by Klarna, the cartridge integration creates a virtual card based settlement, utilizing the merchant card services (MCSv3) API.

Once a settlement has been created (virtual card returned), the merchant platform can authorize the virtual card until the Klarna order is valid. Then, once the order has been fulfilled, the card funds should be captured. (For delays in capture, or other special use cases, please speak with the Klarna Key Account Manager in advance). While Klarna is the original payment method of the order, the order amount will be settled with a credit card instead of direct bank account transfer. Refer to the below code in “controllers/KlarnaPayments.js” and update accordingly to the integrated cards processor.

Note: If the Klarna order has a “**fraud_status**” of “**PENDING**”, action is not taken on the order until receiving Klarna’s push notification that the “**fraud_status**” has changed to “**FRAUD_RISK_ACCEPTED**”.

The virtual card issued is limited to 1 single successful authorization per order for a given MID.

For decrypting the credit card details refer to section **9.3 Decrypt VCN Card Details**.

```
107 //
108 Transaction.wrap(function() {
109     paymentInstrument.paymentTransaction.transactionID = session.privacy.KlarnaPaymentsOrderID;
110     paymentInstrument.paymentTransaction.paymentProcessor = paymentProcessor;
111     session.privacy.OrderNo = orderNo;
112     args.Order.custom.kpOrderID = session.privacy.KlarnaPaymentsOrderID;
113     args.Order.custom.kpIsVCN = empty( vcnEnabled ) ? false : vcnEnabled;
114 });
115
116 if ( session.privacy.KlarnaPaymentsFraudStatus === 'PENDING' ) {
117     return { authorized: true };
118 }
119
120 if ( vcnEnabled ) {
121     var isSettlementCreated = _handleVCNSettlement( args.Order, session.privacy.KlarnaPaymentsOrderID, localeObject ); // es1
122     if ( isSettlementCreated ) {
123         // Plug here your Credit Card Processor
124         return require( '*/cartridge/scripts/payment/processor/BASIC_CREDIT' ).Authorize( { 'OrderNo': args.Order.getOrderNo(),
125         });
126     }
127     var klarnaPaymentsCancelOrderHelper = require( '*/cartridge/scripts/order/klarnaPaymentsCancelOrder' );
128     klarnaPaymentsCancelOrderHelper.cancelOrder( localeObject, args.Order );
129     return { error: true };
130 }
131 return { authorized: true };
132 }
```

Figure 12 Credit Card Authorization Call

To utilize virtual card integration option the merchant should:

- Enable VCN option in Site Preferences as shown above
- Enter the VCN Public Key ID. Unique UUIDv4 value, which should be different for playground testing and Production (live site)

Name	Value	Default Value
The Klarna Payments not available message on billing page. JSON string holdin...		
Virtual Card Network Enabled	<input type="text" value="Yes"/>	No
If set to true SFCC will create Virtual Card Network settlement from every Klarn...		
VCN Public Key ID	<input type="text" value="6c5b99c5-b0de-4689-b569-1ae12ec898eb"/>	
Unique identifier for the public key used for encryption of the card data		
VCN Private Key	<input type="text" value="MIUkQIBAAKCAgEAtirQ01705IOj5GLAGvynckEyK3V3eXZMxjhg05XSSAGKS1aY6mpACVNSLjYRrh4K2QV1C5n8vm1Dz56bUkNKIfyUwSgEusn1ewYkP20audwjyUYYWbWkQc50aHTYAlaekUGC5wAh5ix5c+o25Xr6wx88M3Rv57mQta45WMGeu1z9ZbpbUOTKl738b3/6cxCO9usU2Z9FC4/2RIUGD8KyIV7TVwmbDSIC8DlelpKbZl3wQB1FDw08uNIE+TjYUUKXpCmw8yScR+Xe3gPkL8yeM7RQcOKLJLlLSI8LKREIKCKLMk3rbaNTsPN+1N5LqxrRdjmh2HnaDFkKbBlnHqZn93jY57ZfZaH5F02O6"/>	
Your 4096 bit RSA Private Key		
	<input type="text" value="MIICjANBgkqhkiG9w0BAQEFAAACgAMIIICgKCAgEAtirQ01705IOj5GLA"/>	

Figure 13 VCN Public Key ID

- Generate a 4096-bit RSA key pair (Refer to section **Generate Key Pair and Key Id for Virtual Card Settlements**). Set the custom preference “**vcnPublicKey**” with the value of the public key without the header and footer lines (begin and end public key) and the custom preference “**vcnPrivateKey**” with the value of the private key without the header and footer lines (begin and end private key)

The screenshot shows the 'Merchant Tools' section of the Klarna Payments for SG v21.1.0 interface. Under the 'Administration' tab, the 'Virtual Card Network Enabled' setting is set to 'Yes'. Below this, there are two text input fields for keys, both highlighted with red boxes:

- VCN Private Key:** Contains a long string of characters, including 'V3enexco3d5AoiBAGU5qEYp0jz1+9D15a18LWZCABcizH+9g4A8A20s0X5q7W'. Below the input field, it says 'Your 4096 bit RSA Private Key'.
- VCN Public Key:** Contains a long string of characters, including 'MIICjANBgqhkiG9w0BAQEFAAOCBgAIIICgKAgEAAoNYG712Gbn2a+22oBYZk'. Below the input field, it says 'Your 4096 bit RSA Public Key'.

Each key field has an 'Edit Across Sites' link to its right.

Figure 14 VCN Public & Private Keys

- Update the VCN settlement retry setting “**kpVCNRetryEnabled**”. By default, this is disabled. However, if enabled and in cases when a VCN creation error is returned, the application will retry the settlement request once again with order_id as the idempotency key.
- Finally, you need to send the generated unique key_id + public key combination in JWK format to Klarna prior to testing and go-live. It will be used to encrypt the aes key which is used for encrypting the pci data on Klarna side when settlement request is made. After confirmation from Klarna that the key has been successfully added to your merchant profile you would be able to use virtual card-based settlement option for Klarna payment methods

If enabled and fully configured, virtual card settlement request is made successfully. For orders placed with the VCN settlement option, the related custom attributes are shown below:

[Merchant Tools](#) > [Ordering](#) > [Orders](#) > Order: 00049309(RefArch)

General **Attributes** Payment Notes History

Attributes for Order '00049309'

On this page you can edit the attributes of the order. Fields with a red asterisk (*) are mandatory. Click **Apply** to save changes. Click **Reset** to revert your changes.

Klarna Payments

Klarna Payments Order ID:

Is VCN Used: ☒

VCN Card ID:

[<< Back to List](#)

Figure 15 VCN Details in Order

If required, the additional virtual card details can be assigned to this group in Administration > Site Development > System Object Types > select “Order”. In the Attribute Grouping tab select Klarna_Payments and click on “edit”. Assign the new attributes and save the data.

[Administration](#) > [Site Development](#) > [System Object Types](#) > [Order - Attribute Groups](#) > Klarna Payments

Object Type 'Order' - Attribute Definition Assignments

On this page you can assign existing attribute definitions to your attribute group.

Assign Attribute Definition

ID: [Add](#)

Select All	ID	Name	Type	Attribute Settings	Sorting
<input type="checkbox"/>	kpOrderID	Klarna Payments Order ID	String		
<input type="checkbox"/>	kpIsVCN	Is VCN Used	Boolean		
<input type="checkbox"/>	kpVCNCardID	VCN Card ID	String		
<input type="checkbox"/>	kpVCNHolder	VCN Holder	String		
<input type="checkbox"/>	kpVCNBrand	VCN Brand	String		
<input type="checkbox"/>	kpVCNPCIData	VCN PCI Data	String		
<input type="checkbox"/>	kpVCNIV	VCN Initialization Vector	String		
<input type="checkbox"/>	kpVCNAESKey	VCN AES Key	Text		

[Unassign](#)

[<< Back](#)

Figure 16 Full List of VCN Attributes

Please work with Klarna Account Manager and Delivery contact in advance to select the appropriate virtual card product based on your business requirements and use-cases. You can find information [here](#) around other use cases supported.

Important Note!

*DO NOT SAVE DECRYPTED PCI DATA ON THE SERVER. It is the responsibility of the merchant to ensure PCI-DSS compliance and to ensure the card data is handled securely in co-ordination with required partners/Payment Service Provider/Acquirer. Please review in advance the order export details required for virtual card-based Klarna orders. Any historical decrypted PCI data should also be expunged, regardless of the validity date (see section **3.4.1 Job "OrderCleanup")**.*

2.3.9. Auto-Capture

Auto-capture is enabled via a site preference “**kpAutoCapture**” located in “**Klarna_Payments**” preference group.

When the preference is enabled (disabled by default), a full amount capture is attempted. If the capture is successful, the SFCC order's payment transaction is marked as Paid, and viewable in the Business Manager.

The order will be marked as *"Captured"* in the Klarna's Merchant Portal.

Merchant reference 1

00029205

Kiarna reference

BJJGH9P1

Created

Jun 25, 2019, 5:00 PM

Expires

Jul 23, 2019, 3:00 AM

Merchant 1

K500726

Customer

Shipping address

Angela Gill

4939 Wyatt Street

West Palm Beach

SW42 4RG Florida

GB

Tel

01222 555 555

Email

Ivan.zanev@tryzens.com

Edit shipping address

Billing address

Additional Info

Order lines (2)

	Qty	Item	Reference	Unit price	Discount	Tax	Amount
<input type="checkbox"/>	1	Black Flat Front Wool Suit	7505187030...	191.99	0.00	5% 9.14	191.99
<input type="checkbox"/>	1	Наземен транспорт	GBP001	7.99	0.00	5% 0.38	7.99

Refund

PAYMENT DETAILS

Initial Payment Method

Statement

Resend statement

ORDER TOTAL

Captured

Refunded

Not Captured

CUSTOMER BILLED

E199.98

E199.98

E0.00

E0.00

E199.98

Activity Log

Jun 25, 2019

9 minutes ago

Order acknowledged

Via API

5:00 PM

Captured: E199.98

Via API

5:00 PM

Order placed: E199.98

Figure 17 Order Details in Klarna Portal

If the capture is unsuccessful, an error will be logged in the custom error log. The setting must be reviewed with Klarna delivery team before testing and go-live.

Note: Auto-capture is possible for orders when VCN is not enabled!

2.3.10. Widget Customizations

Note: The merchant will need a configured Klarna Payments account.

The merchant can style the Klarna Payments widget (skin), to match the marketing and branding needs of their store. The list with the graphic elements that can be customized out of the box through site preferences are listed below:

"color_details" (site preference kpColorDetails): "#COFFEE"

"color_button" (site preference kpColorButton): "#COFFEE"

"color_button_text" (site preference kpColorButtonText): "#COFFEE"

"color_checkbox" (site preference kpColorCheckbox): "#COFFEE"

"color_checkbox_checkmark" (site preference kpCheckboxCheckmark): "#COFFEE"

"color_header"(site preference kpColorHeader): "#COFFEE"

"color_link"(site preference kpColorLink): "#COFFEE"

"color_border"(site preference kpColorBorder): "#COFFEE"

"color_border_selected"(site preference kpBorderSelected): "#COFFEE"

"color_text"(site preference kpColorText): "#COFFEE"

"color_text_secondary"(site preference kpColorTextSecondary): "#COFFEE"

"radius_border"(site preference kpRadiusBorder): "0px"

2.3.11. Customizing Payment Method Name

The payment method name “Klarna Payments” may be customized via the “Merchant Tools > Ordering > Payment Methods” section in Business Manager.

The screenshot below shows the “Klarna” method selected and the administrator choosing a language from the drop-down.

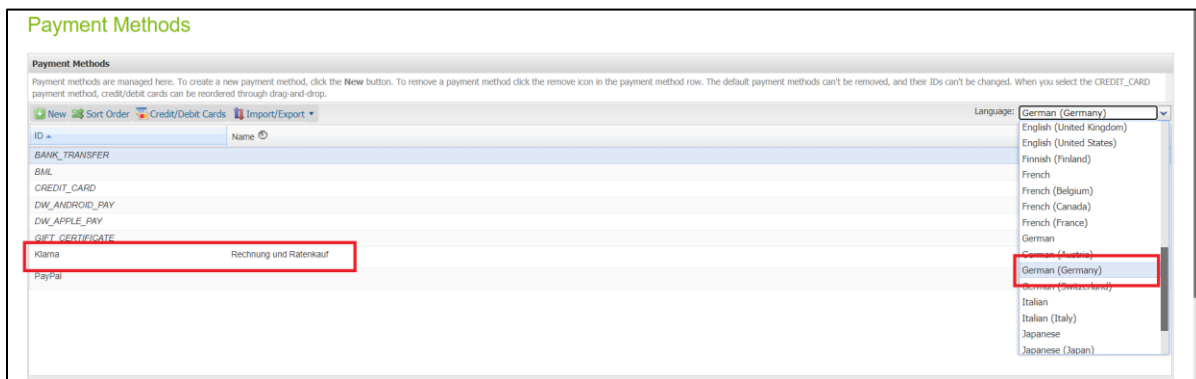


Figure 18 Customize Payment Name

The payment method name would then be visible in the mini summary & confirmation screens, the confirmation emails and My Account Order Details section.

Order Number: 00055604

Payment Information		
Billing Address Testperson-de Approved Vogelhüttendeich 123 Hamburg, HH 21107 Germany Phone: +494086687781	Payment Method Rechnung und Ratenkauf Klarna order reference: 76aa2465-33dd-2ee0-b3f9-5a038dbe0a71 Amount: 108,79 €	Payment Total Subtotal 112 Order Discount - 11 € Shipping 7,99 Ground Order Total: 108

Shipment#1

Item	Quantity	Price	Shipping To
Straight Fit Shorts	1	112,00 €	Testperson-de Approved

Figure 19 Payment Method Name in Email

2.3.12. Klarna On-Site Messaging

Klarna On-Site Messaging (OSM) is configured by site and by locale via the **KlarnaCountries** custom object. To configure the OSM settings for a locale, you must visit “**Merchant Tools – Custom Object Editor**” and search for **KlarnaCountries** custom object. Provide a valid locale for the OSM tag based on the country being configured. The OSM Data Client ID and Data keys required are available in Klarna Merchant Portal within the On-site Messaging App:

The screenshot shows the 'Manage 'US' (KlarnaCountries)' configuration page in the Business Manager. The page is titled 'Merchant Tools > Custom Objects > Custom Objects > US - General'. Below the title, there is a 'General' tab and a note: 'Fields with a red asterisk (*) are mandatory. You can view and edit the name and description in other languages, if required. Click Apply to save the details.' The configuration form includes the following fields and checkboxes:

- Country Code:** US
- On-site Messaging Data Default Locale:** en-US
- Service Credential ID:** klarna.http.uscredentials
- On-site messaging Data Client ID:** 60a22a39-c2fd-5d09-bfe1-771459318a4d
- Cart Placement Tag Enabled:** ☒
- Cart Placement Data Key:** info-page-standard
- PDP Placement Tag Enabled:** ☒
- PDP Placement Data Key:** credit-promotion-small
- Header Placement Tag Enabled:** ☒
- Header Placement Data Key:** top-strip-promotion-standard
- Footer Placement Tag Enabled:** ☒
- Footer Placement Data Key:** footer-promotion-auto-size
- Info Page Placement Tag Enabled:** ☒
- Info Page Placement Data Key:** info-page
- Library URL:** https://na-library.playground.klarnaservices.com/lib.js

Figure 20 OSM Settings in Business Manager

To enable Placement tag for the Cart Page, the “Cart Placement Data Key” must be filled with Data Key value and the “Cart Placement Tag Enabled” must be checked.

To enable Placement tag for the PDP Page, the “PDP Placement Data Key” must be filled with Data Key value and the “PDP Placement Tag Enabled” must be checked.

To enable Placement tag for header, the “Header Placement Data Key” must be filled with Data Key value and the “Header Placement Tag Enabled” must be checked.

To enable Placement tag for footer, the “Footer Placement Data Key” must be filled with Data Key value and the “Footer Placement Tag Enabled” must be checked.

To enable Placement tag for the Info Page, the “Info Page Placement Data Key” must be filled with Data Key value and the “Info Page Placement Tag Enabled” must be checked.

In Library URL, please input the full URL to the On-Site Messaging JavaScript Library.

In On-site messaging Data Client ID, please input the On-Site Messaging Data client ID value.

On-site messaging Data locale, please input the valid On-Site Messaging Data locale.

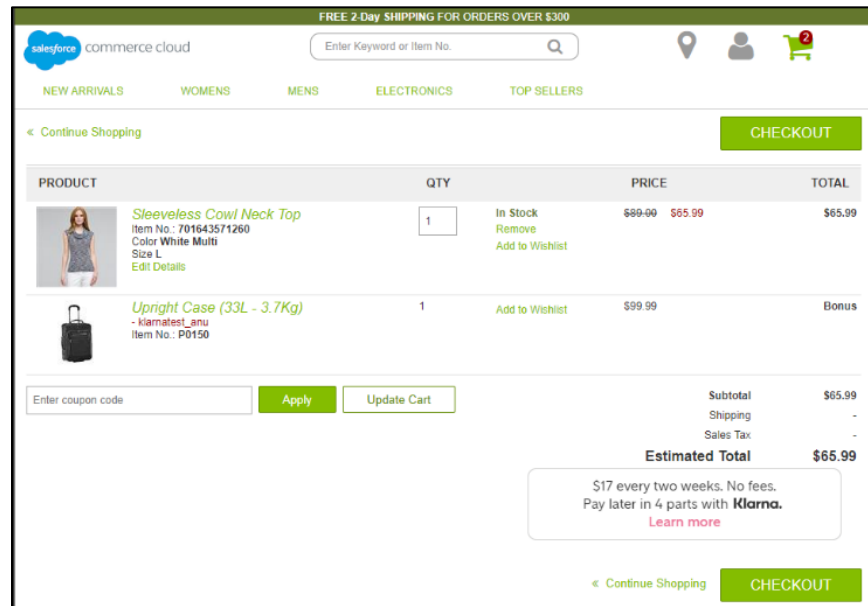


Figure 21 On-Site Messaging Enabled on Cart Page

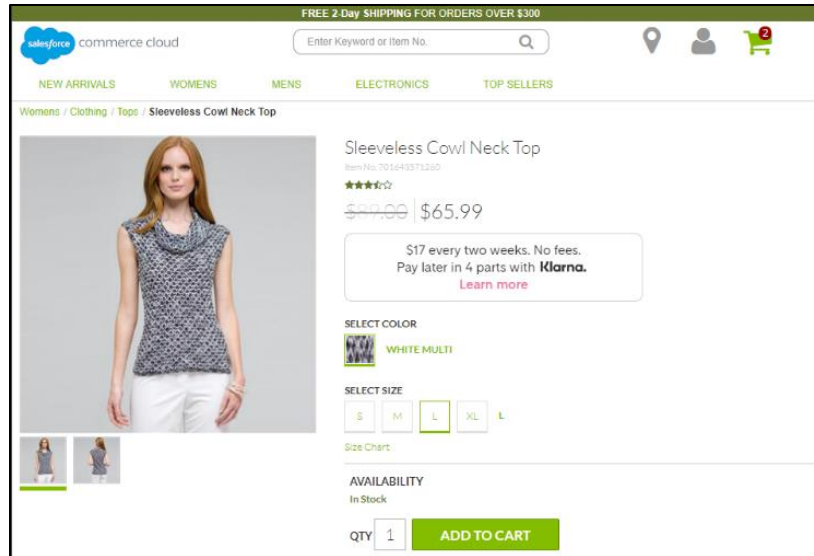


Figure 22 On-Site Messaging Enabled on PDP Page

In addition to the above, if you wish to display the dedicated (custom) Klarna info OSM page you can use the following controller endpoint “**KlarnaPayments-InfoPage**”. For example, you should update the “**footer-about**” content asset to include this line of code as shown on the screenshot.

```
<li><a href="$url('KlarnaPayments-InfoPage')$" title="Go to Klarna Info">Klarna Info</a></li>
```

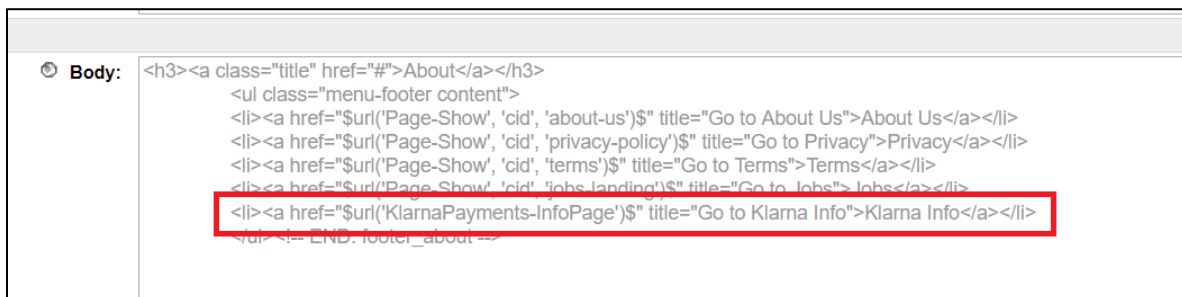


Figure 23 Footer Asset Update

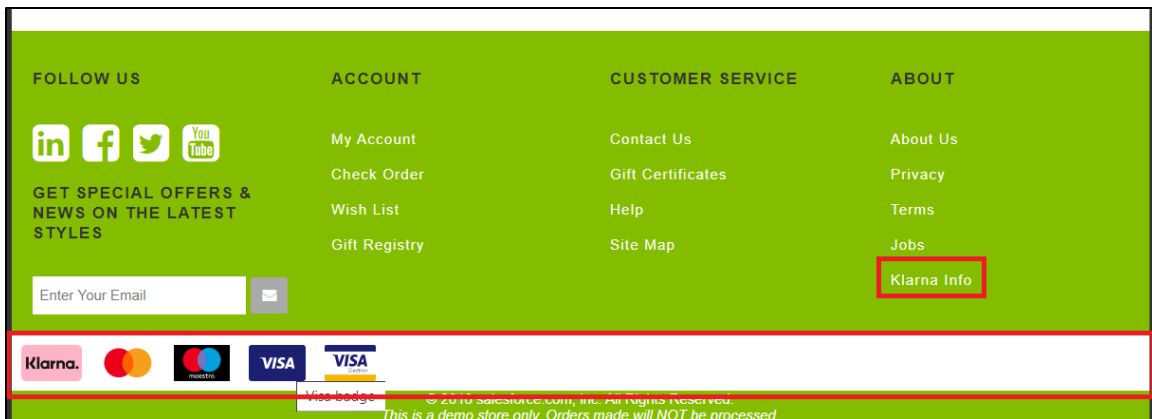


Figure 24 On-Site Messaging Enabled on Footer and link to Klarna Info Page

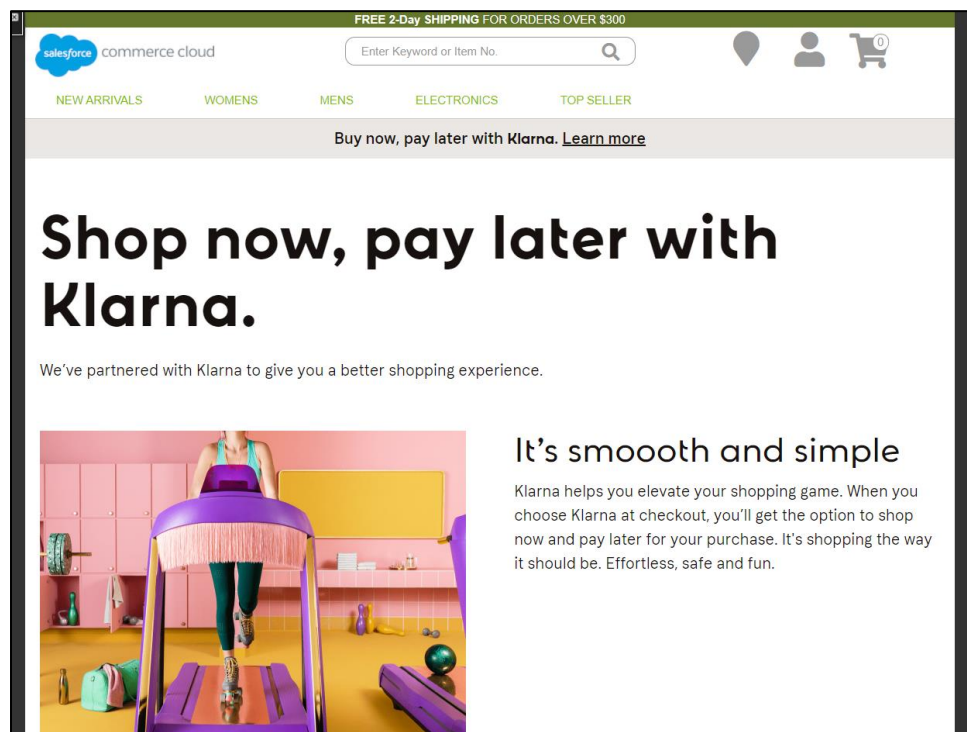


Figure 25 On-Site Messaging Enabled on Header & dedicated Klarna Info Page

For more information regarding OSM customizations and best practices, please refer to the Klarna Developer: <https://developers.klarna.com/documentation/on-site-messaging/customization/>

Integration Best practice + information about Klarna Branding and Co-marketing options [here](#).

2.3.13. Klarna Payment Method Based Promotions

As of B2C 20.7 release, merchants can include payment methods as qualifier for product, order & shipping promotions.

OOTB when such promotion is set up to use a payment method as qualifier, the total order amount will be visible to the customer once they reach the review page. This means that the Klarna authorization call will be made for larger amount than the final one.

To address this issue, once the customer clicks on any of the payment options in the billing section – a call will be made to the backend. This will re-calculate the basket totals if any promotions are applicable and will update the Klarna session details. As a result, the Klarna iframe widgets and the mini summary section on the storefront will update to show the final order details.

***Note:** When the selected payment method is non-Klarna one, this logic should be customized by the merchant to handle any 3rd payment integrations.*

2.3.14. Price Adjustment Taxation Handling

OOTB the Klarna API calls will send the product / shipping method details and the relevant discounts as separate lines items as shown below:

```
"order_lines": [  
  {  
    "type": "discount",  
    "name": "5 Off Ties Promotion",  
    "reference": "682875540326M_$5_off_ties_promotion",  
    "quantity": 1,  
    "merchant_data": "5ties",  
    "unit_price": -500,  
    "tax_rate": 500,  
    "total_amount": -500,  
    "total_tax_amount": 0,  
    "total_discount_amount": 0,  
    "product_url": null,  
    "image_url": null  
  },  
  {  
    "type": "physical",  
    "name": "Checked Silk Tie",  
    "reference": "682875540326M",  
    "quantity": 1,  
    "merchant_data": "",  
    "unit_price": 1919,  
    "tax_rate": 500,  
    "total_amount": 1919,  
    "total_tax_amount": 68,  
    "total_discount_amount": 0,  
  }  
]
```

Figure 26 Line Items with Default Taxation Setting

In some cases, merchants using gross taxation might enable the “Tax Products and Shipping Only Based on Adjusted Price” preference under “Merchant Tools > Site Preferences > Pricing and Promotion” where the price adjustments are not taxed.

The setting, “**kpPromoTaxation**” has been introduced, where merchants should update this to match the promotion setting below:

- price (Based on Price) – The product, shipping and their discounts will be sent as separate lines. This is the default setting.
- adjustment (Based on Adjusted Price) – When this is selected, the product or shipping method line item will be sent with attribute “total_amount” matching the prorated price and attribute “total_discount_amount” – matching the total sum of all discounts for this item.

```
"order_lines": [  
  {  
    "type": "physical",  
    "name": "Checked Silk Tie",  
    "reference": "682875540326M",  
    "quantity": 1,  
    "merchant_data": "",  
    "unit_price": 1919,  
    "tax_rate": 2200,  
    "total_amount": 1419,  
    "total_tax_amount": 256,  
    "total_discount_amount": 500,  
  }  
]
```

Figure 27 Line Items with "Based on Adj." Taxation Setting

Note: Enabling this setting is not required for storefronts with net taxation as the tax is not included in the products base price. The total order sales tax is sent as a separate line item to Klarna and not on product/shipping line-item level.

2.3.15. Buy Online, Pickup in Store (BOPIS)

When store pickup has been enabled on the storefront, the integration will send stores details to Klarna in the authorization request and when placing the Klarna order. Store information is not sent prior the interaction of the customer with the Klarna payment method widgets.

The store address(es) is always included in the EMD attachment “other_delivery_address” when applicable.

The address included on the shipping address in the Klarna order with store pick-up, is as below:

- Orders that have 1 or more store pickup shipments (no home delivery address), the first store shipment details will be set as the shipping address
- Order with store pick-up(s) and home delivery shipment, home delivery address will be used as the shipping address in the Klarna calls
- If the order contains no store pickups, no information is sent in “other_delivery_addresses” attribute

```
{
  "attachment": {
    "content_type": "application/vnd.klarna.internal.emd-v2+json",
    "body": {
      "other_delivery_address": [{
        "shipping_method": "store pick-up",
        "shipping_type": "normal",
        "first_name": "Test",
        "last_name": "Customer",
        "street_address": "1487 Bay St",
        "street_number": "",
        "postal_code": "01109",
        "city": "Springfield",
        "country": "US"
      }]
    }
  }
}
```

For more information on the options refer [here](#)

2.4. Compatibility

This cartridge has been tested against API Version 21.3 (Compatibility Mode: 19.10) and SG version 105.0.0.

2.5. Privacy, Payment

2.5.1. GDPR Compliance

The cartridge is compliant with GDPR recommendation and follows the best practice mentioned here and implementation transmits only required (PII) data to authorize payment method.

2.5.2. EMD (Extra Merchant Data)

The cartridge supports sending additional information on the customer's past purchase history, as well as “Buy Online, Pickup in Store” (BOPIS) store addresses when turned on in custom preferences: “Attachments” (**kpAttachments**). The type of data that can be send as an attachment is mentioned here. EMD is required for certain types of merchant orders and the inclusion of EMD is (e.g.customer_account_info: past interaction with merchant store) generally beneficial to improve acceptance rates.

EMD is included as part of Authorization step in Commerce Cloud checkout. The data send to Klarna is customizable & can be seen in “**int_klarna_payments/scripts/payments/additionalCustomerInfo.js**”. This script should return a JSON string to be used as a value for the body sub-field of the attachment field as [described here](#) .

If the example additionalCustomerInfo.js file is used unchanged the data send to Klarna is by the following schema:

```
{
  "$schema": "http://json-schema.org/draft-03/schema#",
  "id": "http://klarna.com/v2/emd#",
  "description": "Extended Merchant Data Payload Schema",
  "type": "object",
  "properties": {
    "customer_account_info": {
      "type": "array",
      "items": {
        "type": "object",
        "properties": {
          "unique_account_identifier": {
            "type": "string",
            "maxLength": 24
          },
          "account_registration_date": {
            "description": "ISO 8601 e.g. 2012-11-24T15:00",
            "type": "string",
            "format": "date-time",
            "pattern": "^[0-9][0-9][0-9][0-9]-[0-1][0-9]-[0-3][0-9]T[0-2][0-9]:[0-5][0-9](:[0-5][0-9]){0,1}Z{0,1}$"
          },
          "account_last_modified": {
            "description": "ISO 8601 e.g. 2012-11-24T15:00",

```

```

        "type": "string",
        "format": "date-time",
        "pattern": "^[0-9][0-9][0-9][0-9]-[0-1][0-9]-[0-3][0-9]T[0-2][0-9]:[0-5][0-9](:[0-5][0-9]){0,1}Z{0,1}$"
    }
}
},
"payment_history_full": {
    "type": "array",
    "items": {
        "type": "object",
        "additionalProperties": false,
        "properties": {
            "unique_account_identifier": {
                "type": "string"
            },
            "payment_option": {
                "type": "string",
                "enum": ["card", "direct banking", "non klarna credit", "sms", "other"]
            },
            "number_paid_purchases": {
                "type": "integer"
            },
            "total_amount_paid_purchases": {
                "type": "number"
            },
            "date_of_last_paid_purchase": {
                "description": "ISO 8601 e.g. 2012-11-24T15:00",
                "type": "string",
                "format": "date-time",
                "pattern": "^[0-9][0-9][0-9][0-9]-[0-1][0-9]-[0-3][0-9]T[0-2][0-9]:[0-5][0-9](:[0-5][0-9]){0,1}Z{0,1}$"
            },
            "date_of_first_paid_purchase": {
                "description": "ISO 8601 e.g. 2012-11-24T15:00",
                "type": "string",
                "format": "date-time",
                "pattern": "^[0-9][0-9][0-9][0-9]-[0-1][0-9]-[0-3][0-9]T[0-2][0-9]:[0-5][0-9](:[0-5][0-9]){0,1}Z{0,1}$"
            }
        }
    }
},
"other_delivery_address": {
    "type": "array",
    "items": {
        "type": "object",
        "additionalProperties": false,
        "properties": {
            "shipping_method": {
                "type": "string",
                "enum": ["store pick-up", "pick-up point", "registered box", "unregistered box"]
            },
            "shipping_type": {
                "type": "string",
                "enum": ["normal", "express"]
            }
        }
    }
},

```

```

        "first_name": {
          "type": "string"
        },
        "last_name": {
          "type": "string"
        },
        "street_address": {
          "type": "string"
        },
        "street_number": {
          "type": "string"
        },
        "postal_code": {
          "type": "string"
        },
        "city": {
          "type": "string"
        },
        "country": {
          "type": "string"
        }
      }
    }
  }
}

```

Example data:

```

{
  "attachment": {
    "content_type": "application/vnd.klarna.internal.emd-v2+json",
    "body": {
      "customer_account_info": [{
        "unique_account_identifier": "5509d9f7c8720c0e4575154b",
        "account_registration_date": "2015-03-18T20:03:03Z",
        "account_last_modified": "2015-03-18T20:03:03Z"
      }],
      "purchase_history_full": [{
        "unique_account_identifier": "5509d9f7c8720c0e4575154b",
        "payment_option": "card",
        "number_paid_purchases": "23",
        "total_amount_paid_purchases": "140023",
        "date_of_last_paid_purchase": "2015-03-18T20:03:03Z",
        "date_of_first_paid_purchase": "2015-03-18T20:03:03Z"
      }],
      "other_delivery_address": [{
        "shipping_method": "store pick-up",
        "shipping_type": "normal",
        "first_name": "Test",
        "last_name": "Customer",
        "street_address": "1487 Bay St",
        "street_number": "",
        "postal_code": "01109",
        "city": "Springfield",
        "country": "US"
      }]
    }
  }
}

```




2.5.3. PCI-DSS Compliance

Important Note: DO NOT SAVE DECRYPTED PCI DATA ON THE SERVER!

The virtual card (MCSv3) solution enables settlements using individual virtual card issued against a Klarna order. To be compliant with PCI-DSS requirements, merchant must ensure the data is securely maintained and transmitted as part of their operation in their live store environment. The required steps to ensure this, must be done in consultation with your payment service provider/acquirer and completed prior to go-live. Please review in advance the order export details required for virtual card-based Klarna orders. Any historical decrypted PCI data should also be expunged, regardless of the VCN validity date.

3. Implementation Guide

3.1. Setup of Business Manager

The Klarna Payments LINK Cartridge contains 2 cartridges that are required for full functionality. Additionally, Controller and SFRA support is broken out into two separate cartridges, thereby facilitating the installation and use of one or the other models.

int_klarna_payments – Implements the core storefront functionality.

int_klarna_payments_controllers – Implements the storefront functionality with SG code.

3.1.1. Cartridge Upload & Assignment

Import the two cartridges into UX studio and associate them with a Server Connection.

- Import the “**int_klarna_payments**” cartridge into the SCC Studio Workspace:
 - Open SCC Studio
 - Click File -> Import -> General -> Existing Projects into Workspace
 - Browse to the directory where you saved the “**int_klarna_payments**” cartridge.
 - Click Finish.
 - Click OK when prompted to link the cartridge to the sandbox.
- Import the “**int_klarna_payments_controllers**” cartridge into the SCC Studio Workspace:
 - Open SCC Studio
 - Click File -> Import -> General -> Existing Projects into Workspace
 - Browse to the directory where you saved the “**int_klarna_payments_controllers**” cartridge.
 - Click Finish.
 - Click OK when prompted to link the cartridge to the sandbox.
- Prepend the Klarna cartridges to the effective site cartridge path:
 - Log into the SCC Business Manager.

- Click Administration -> Sites -> Manage Sites.
- Select the desired site.
 - Click on the Settings tab.
 - Prepend “**int_klarna_payments_controllers:int_klarna_payments**” to the “**Cartridges**” field.
 - Click Apply

Click **Apply** to save the details. Click **Reset** to revert to the last saved state.

Instance Type: Sandbox/Development ▾

Deprecated. The preferred way of configuring HTTP and HTTPS hostnames is by using new features of the site aliases configuration ("SEO > Aliases Configuration"). The HTTP/HTTPS hostnames are intended only to support an older configuration style.

HTTP Hostname:

HTTPS Hostname:

Instance Type: All

Cartridges:

Effective Cartridge Path:

- app_storefront_controllers
- app_storefront_core
- int_klarna_payments_controllers
- int_klarna_payments
- plugin_apple_pay
- plugin_facebook
- plugin_payments
- plugin_pinterest_commerce
- plugin_web_payments
- bc_content
- core

[<< Back to List](#)

Figure 28 Effective Cartridge Path

3.1.2. Metadata Import

- Go to main directory “**metadata**” folder, review the site-template content, and edit if needed. (Site template is prepared to setup “**SiteGenesis**” and “**RefArch**” sites - you may want to change that to your actual sites and delete the ones that are not needed). Zip the directory and you will have “**site-template.zip**” installation package.
- Log into the SCC Business Manager.
- Click Administration -> Sites Development -> Site Import & Export
- Browse to the directory where you saved the “**site-template.zip**”.
- Click “**Upload**”
- Select the uploaded site zip and click “**Import**”.

3.2. Configuration

- Add your account settings to the KlarnaCountries Custom Objects.
 - Log into the SCC Business Manager.
 - Select the desired site from the tabs across the top of the page.
 - Click Custom Objects -> Custom Object Editor
 - Change the Object Type dropdown to “**KlarnaCountries**”.
 - Click the “**Find**” button.
 - Click the desired country you wish to edit (See screenshot below).
 - Update the required fields as mentioned in “**KlarnaCountries**” section
 - Repeat for the other countries.

Merchant Tools > Custom Objects > Custom Objects > US - General

General

Manage 'US' (KlarnaCountries)

Fields with a red asterisk (*) are mandatory. You can view and edit the name and description in other languages, if required. Click **Apply** to save the details.

custom

Country Code:	US
On-site Messaging Data Default Locale:	en-US
Service Credential ID:	klarna.http.uscredentials
On-site messaging Data Client ID:	60a22a39-c2fd-5d09-bfe1-771459318a4d
Cart Placement Tag Enabled:	<input checked="" type="checkbox"/>
Cart Placement Data Key:	info-page-standard
PDP Placement Tag Enabled:	<input checked="" type="checkbox"/>
PDP Placement Data Key:	credit-promotion-small
Header Placement Tag Enabled:	<input checked="" type="checkbox"/>
Header Placement Data Key:	top-strip-promotion-standard
Footer Placement Tag Enabled:	<input checked="" type="checkbox"/>
Footer Placement Data Key:	footer-promotion-auto-size
Info Page Placement Tag Enabled:	<input checked="" type="checkbox"/>
Info Page Placement Data Key:	info-page
Library URL:	https://na-library.playground.klarnaservices.com/lib.js

Figure 29 KlarnaCountries Settings

- Configure Klarna Payment Custom Preferences using the SCC Business Manager
 - Log into the SCC Business Manager

- Select the desired site from the tabs across the top of the page.
 - Click Site Preferences -> Custom Preferences -> KlarnaPayment
 - Fill out the settings as desired. Descriptions of the site preferences are in the **Site Preferences** section.
- Configure Klarna Payment Service using the SCC Business Manager
 - Log into the SCC Business Manager
 - Click Administration > Operations > Services.
 - Click the Credentials tab.
 - Each Klarna credential correspond to one of the **KlarnaCountries** custom objects. Click on the one you want to edit.
 - Enter the MID API username and API password you received from Klarna.
 - Edit URL field if Production environment. Klarna API URLs information - <https://developers.klarna.com/api/#api-urls>.

[Administration](#) > [Operations](#) > [Services](#) > [Service Credentials](#) > klarna.http.gbcredentials - Details

klarna.http.gbcredentials

Fields with a red asterisk (*) are mandatory. Click **Apply** to save the details. Click **Reset** to revert to the last saved state.

These credentials are used by 0 services.

Name: *	<input type="text" value="klarna.http.gbcredentials"/>
URL:	<input type="text" value="https://api.playground.klarna.com/"/>
User:	<input type="text" value="your Merchant ID"/>
Password:	<input type="password" value="••••••••"/>

Figure 30 Service Settings

3.3. Template Updates

Templates have been updated to support On-site messaging and Addresses forms for Klarna. To be used as reference but feel free to customize the templates to match your specific needs. Final review and sign-off as per project requirements and contract agreements.

3.4. Jobs

3.4.1. Job “OrderCleanUp”

This 1-time clean-up job is only applicable to merchants integrated with Klarna Payments cartridge version (< 19.1.6), utilizing (or previously used) virtual card-based settlement (VCN) and stored decrypted card details within Business Manager.

The job iterates over orders with status “**Exported**” and attribute “**custom.kplsVCN=true**” to remove the sensitive details saved in fields kpVCNPAN, kpVCNCSC, kpVCNExpirationMonth, kpVCNExpirationYear as part of the previous releases. There are no parameters passed to the script.

Upon successful run, the job will log the result of processed orders in the custom debug log located in “**webdav/Sites/Logs**”. Depending on the setup, you will receive a message for the processed orders count for each storefront or message that there are no orders needing update.



```
Wed, 09 Sep 2020 09:45:38 GMT  DEBUG CustomJobThread[1740004063]OrderCleanUp|executeRefArch custom [] Job [OrderCleanUp] - [RefArch] No orders require processing
Wed, 09 Sep 2020 09:45:38 GMT  DEBUG CustomJobThread[1740004063]OrderCleanUp|executeRefArchGlobal custom [] Job [OrderCleanUp] - [RefArchGlobal] No orders require processing
Wed, 09 Sep 2020 09:45:38 GMT  DEBUG CustomJobThread[1740004063]OrderCleanUp|executeSiteGen custom [] Job [OrderCleanUp] - [SiteGenesis] Orders processed: 2
Wed, 09 Sep 2020 09:45:38 GMT  DEBUG CustomJobThread[1740004063]OrderCleanUp|executeSiteGenGlobal custom [] Job [OrderCleanUp] - [SiteGenesisGlobal] No orders require processing
```

Figure 31 Job Logs

Upon error, the cause of the failure (message and stack) will be logged in the standard error log.

```
Wed, 09 Sep 2020 09:05:22 GMT ERROR CustomJobThread[618864818]OrderCleanup|execute com.demandware.beehive.core.internal.do
main.SystemObjectQueryMgrImpl Sites-RefArch-Site 308 43e0d91e96 e67f8cb0ade56d99ae0c4bc12b
1648625161284573384 - Exception while parsing system object query.
System
RequestID: e67f8cb0ade56d99ae0c4bc12b
SessionType: 308
Truncated SessionID: 43e0d91e96
User Profile UUID: 03e37ff77529fda91ae75408ab
```

Figure 32 Job Logs

To setup the job, go to **Administration > Operations > Import & Export** and import file “**jobs.xml**”. Out of the box, the XML file includes only the RefArch scope, but it can be configured with multiple flows if you have more than one site using this functionality as seen bellow. Each site should be added as a separate flow.

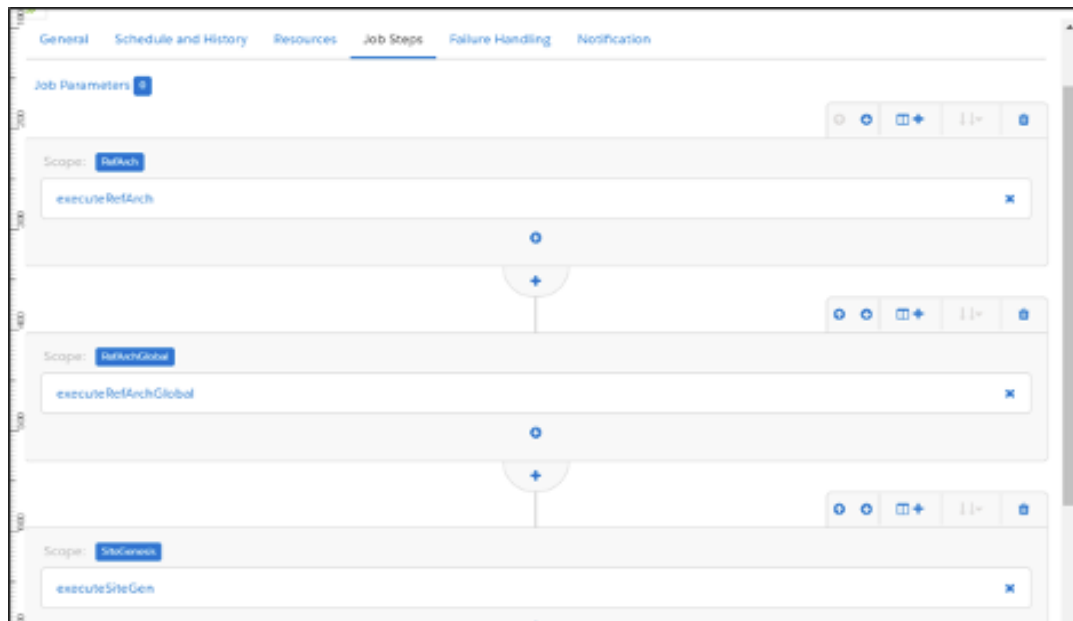


Figure 33 Job Steps

To set up the required job parameters and add new flow, follow the bellow steps. If you only have one storefront and need to change the scope to the correct one – proceed to steps 4-5 directly.

1. Click on the “Add a sequential flow” button at the bottom of the current flow.



Figure 34 Add New Job Step

2. Once done, click on “Configure a step” button within the flow. In the flyout that has just opened search for “script” and select “ExecuteScriptModule”.

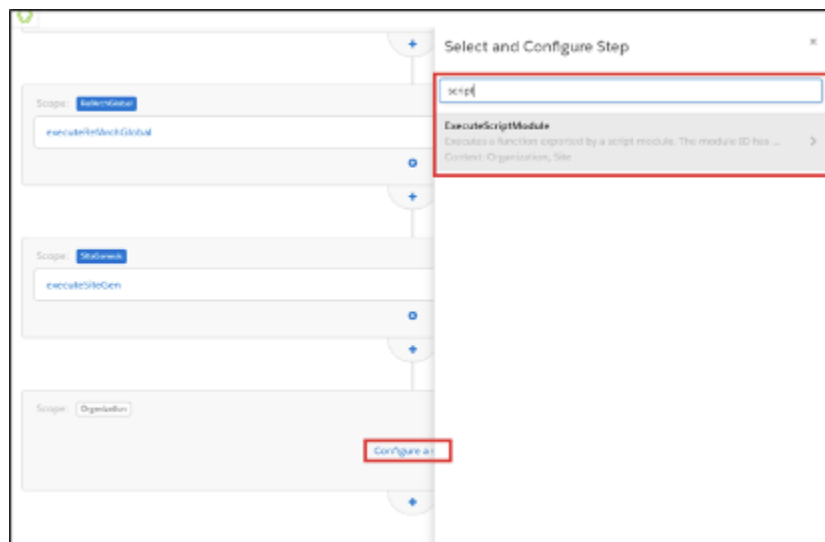


Figure 35 Configure Step

3. In the flyout populate these fields and click the “Assign” button.
 - a. ID – Enter any meaningful name in the field. In case you have multiple flows, this should not be a duplicate one. If you enter a duplicate name, the details won’t be saved and you will see an error message.

- b. ExecuteScriptModule.Module – Enter the location of the “OrderCleanUpJob.js” file. Out of the box it should be “int_klarna_payments/cartridge/scripts/job/OrderCleanUpJob.js” or the location where you have placed the script file.
- c. ExecuteScriptModule.FunctionName – Leave the field value to “execute”

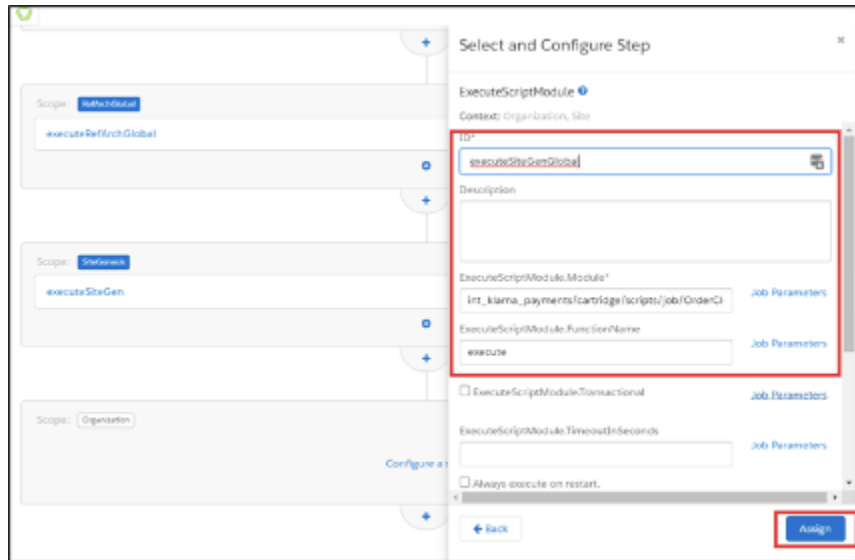


Figure 36 Configure Step (cont.)

4. Once the step has been added, you should make sure to assign it to the correct site scope. Click on “Organization” and select “Specific Sites” from the drop-down.

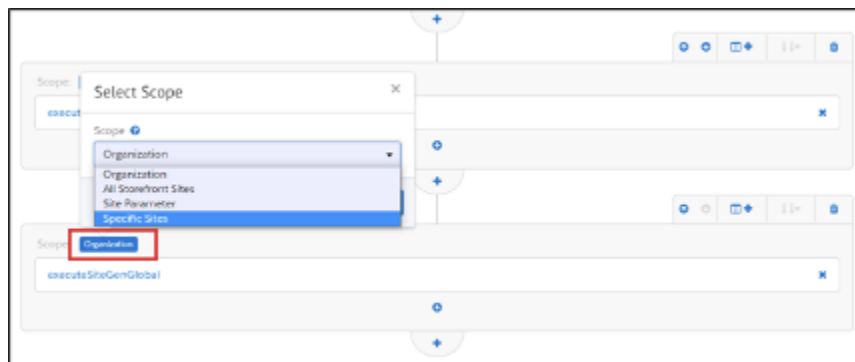


Figure 37 Job Scope

5. From the list of sites, select the respective site ID (i.e. SiteGenesisGlobal) and click on “Assign”.

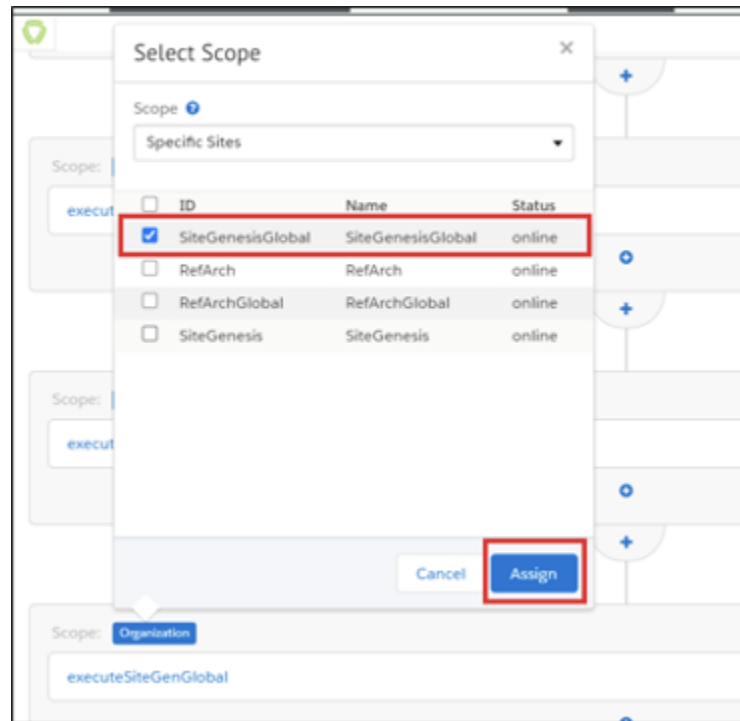


Figure 38 Job Scope (cont.)

6. Repeat steps 1-5 for each site/storefront that you have using Klarna VCN and need additional configuration.

3.5. Custom Code

Integration may vary based on the customers' storefront. Site Genesis version 105.0.0 is used as a reference to demonstrate Klarna Payments integration.

3.5.1. Template Modifications

The following template changes should be made regardless of whether a controller or a pipeline integration approach are being used:

- default/checkout/billing/billing.isml
- default/checkout/summary/summary.isml
- default/checkout/billing/paymentmethods.isml

- default/checkout/shipping/minishipments.isml
- default/components/header/header.isml
- default/components/footer/footer.isml
- default/components/footer/footer_UI.isml
- default/product/producttopcontentPS.isml
- default/product/productcontent.isml
- default/product/productcontent.isml
- default/checkout/cart/cart.isml
- default/mail/orderconfirmation.isml
- default/components/order/ordrdetailsemail.isml

3.5.1.1. default/checkout/summary/summary.isml

Add Code:

```
<isif condition="${session.privacy.KlarnaPaymentsFinalizeRequired}">
  <script>
    <isinclude template="/resources/klarnapaymentsresources.isml"/>
  </script>
  <script type="text/javascript" src="${URLUtils.staticURL('/js/klarna-
payments-finalize.js')}"></script>
  <script src="https://x.klarnacdn.net/kp/lib/v1/api.js" async></script>
</isif>
```

Before `</isdecorate>` closing tag as shown below:

```

214
215
216 <form action="{URLUtils.https('COSummary-Submit')}}" method="post" class="submit-order">
217   <fieldset>
218     <div class="form-row">
219       <a class="back-to-cart" href="{URLUtils.url('cart-show')}}">
220         <isprint value="{Resource.msg('summary.editcart','checkout',null)}" encoding="off" />
221       </a>
222       <button class="button-fancy-large" type="submit" name="submit" value="{Resource.msg('global.submitorder','locale',null)}">
223         {Resource.msg('global.submitorder','locale',null)}
224       </button>
225     </div>
226     <input type="hidden" name="{dw.web.CSRFProtection.getTokenName()}" value="{dw.web.CSRFProtection.generateToken()}" />
227   </fieldset>
228 </form>
229
230 </div>
231
232 <isif condition="{session.privacy.KlarnaPaymentsFinalizeRequired}">
233   <script><isinclude template="/resources/klarnapaymentsresources.isml"/></script>
234   <script type="text/javascript" src="{URLUtils.staticURL('/js/klarna-payments-finalize.js')}"></script>
235   <script src="https://x.klarnacdn.net/kp/lib/v1/api.js" async></script>
236 </isif>
237
238 </isdecorate>
239

```

Figure 39 Modifications in summary.isml

3.5.1.2. default/checkout/billing/billing.isml

Add Code:

```

<script><isinclude
template="/resources/klarnapaymentsresources.isml"/></script>

<script type="text/javascript" src="{URLUtils.staticURL('/js/klarna-
payments.js')} "></script>

<script src="https://x.klarnacdn.net/kp/lib/v1/api.js" async></script>

<link rel="stylesheet" href="{URLUtils.staticURL('/css/klarna-payments.css')} ">
/>

```

Before </isdecorate> closing tag as shown below:

```

203 <isinclude template="checkout/billing/paymentmethods"/>
204 <isbonusdiscountlineitem p_alert_text="{Resource.msg('billing.bonusproductalert','checkout',null)}" p_discount_line_item="{p_discount_line_item}">
205 </isbonusdiscountlineitem>
206 </isif>
207 </isif>
208 </isif>
209 <div class="form-row form-row-button">
210 <button class="button-fancy-large" type="submit" name="{pdict.CurrentForms.billing.save.htmlName}" value="{Resource.msg('global.submitorder','locale',null)}">
211 {Resource.msg('global.submitorder','locale',null)}
212 </button>
213 <input type="hidden" name="{dw.web.CSRFProtection.getTokenName()}" value="{dw.web.CSRFProtection.generateToken()}" />
214 </div>
215 </form>
216 </isdecorate>
217 <script><isinclude template="/resources/klarnapaymentsresources.isml"/></script>
218 <script type="text/javascript" src="{URLUtils.staticURL('/js/klarna-payments.js')} "></script>
219 <script src="https://x.klarnacdn.net/kp/lib/v1/api.js" async></script>
220 <link rel="stylesheet" href="{URLUtils.staticURL('/css/klarna-payments.css')} ">
221 </isdecorate>
222

```

Figure 40 Modifications in billing.isml

3.5.1.3. default/checkout/billing/paymentmethods.isml

Add code:

```
<isif condition="$${paymentMethodType.value === 'Klarna'}">hide</isif>
```

After `<div class="form-row label-inline` close to line 18 as shown below:



```
paymentmethods.isml 13
1 <iscontent type="text/html" charset="UTF-8" compact="true"/>
2 <iscomment> TEMPLATENAME: paymentmethods.isml </iscomment>
3 <isinclude template="util/modules"/>
4 <isif condition="$${pdict.OrderTotal > 0}">
5   <fieldset>
6
7     <legend>
8       $(Resource.msg('billing.paymentheader', 'checkout', null))
9     <div class="dialog-required"> <span class="required-indicator">#8226; <em>$(Resource.msg('global.requiredfield', 'locale', null))</em></span></div>
10    </legend>
11
12    <div class="payment-method-options form-indent">
13      <isloop items="$${pdict.CurrentForms.billing.paymentMethods.selectedPaymentMethodID.options}" var="paymentMethodType">
14
15        <iscomment>Ignore GIFT_CERTIFICATE method, GCs are handled separately before other payment methods.</iscomment>
16        <isif condition="$${paymentMethodType.value.equals(dw.order.PaymentInstrument.METHOD_GIFT_CERTIFICATE)}"><iscontinue/></isif>
17
18        <div class="form-row label-inline" <isif condition="$${paymentMethodType.value === 'Klarna'}">hide</isif>
19          <islet name="radioID" value="$${paymentMethodType.value}" scope="page"/>
20          <div class="field-wrapper">
21            <input id="is-$${radioID}" type="radio" class="input-radio" name="$${pdict.CurrentForms.billing.paymentMethods.selectedPaymentMethodID.htmlName}" />
22          </div>
23          <label for="is-$${radioID}"><isprint value="$${Resource.msg(paymentMethodType.label, 'forms', null)}"/></isif condition="$${empty(dw.order.PaymentMgr.getI
24        </div>
25
26      </isloop>
27
28    </div>
```

Figure 41 Modifications in paymentmethods.isml

Add Code:

```
<isinclude template="klarnapayments/klarnapaymentcategories.isml"/>
```

After `</isloop>` close to line 28 as shown below:



```
paymentmethods.isml 33
6
7
8 <legend>
9   $(Resource.msg('billing.paymentheader', 'checkout', null))
10  <div class="dialog-required"> <span class="required-indicator">#8226; <em>$(Resource.msg('global.requiredfield', 'locale', null))</em></span></div>
11 </legend>
12
13 <div class="payment-method-options form-indent">
14   <isloop items="$${pdict.CurrentForms.billing.paymentMethods.selectedPaymentMethodID.options}" var="paymentMethodType">
15
16     <iscomment>Ignore GIFT_CERTIFICATE method, GCs are handled separately before other payment methods.</iscomment>
17     <isif condition="$${paymentMethodType.value.equals(dw.order.PaymentInstrument.METHOD_GIFT_CERTIFICATE)}"><iscontinue/></isif>
18
19     <div class="form-row label-inline" <isif condition="$${paymentMethodType.value === 'Klarna'}">hide</isif>
20       <islet name="radioID" value="$${paymentMethodType.value}" scope="page"/>
21       <div class="field-wrapper">
22         <input id="is-$${radioID}" type="radio" class="input-radio" name="$${pdict.CurrentForms.billing.paymentMethods.selectedPaymentMethodID.htmlName}" />
23       </div>
24       <label for="is-$${radioID}"><isprint value="$${Resource.msg(paymentMethodType.label, 'forms', null)}"/></isif condition="$${empty(dw.order.PaymentMgr.getI
25     </div>
26
27   </isloop>
28
29   <isinclude template="klarnapayments/klarnapaymentcategories.isml"/>
30 </div>
31
32 <div class="form-row form-row-button">
```

Figure 42 Modifications in paymentmethods.isml (cont.)

Add Code:


```
<iscomment>
```

```
    Klarna Payments
```

```
</iscomment>
```

```
<isinclude template="klarnapayments/klarnapaymentblock.isml"/>
```

Before `</fieldset>` closing tag, close to line 150 as shown below:



```
124 <inputfield formfield="{pdict.CurrentForms.billing.paymentMethods.bml.year}" type="select" rowclass="year"/>
125 <inputfield formfield="{pdict.CurrentForms.billing.paymentMethods.bml.month}" type="select" rowclass="month"/>
126 <inputfield formfield="{pdict.CurrentForms.billing.paymentMethods.bml.day}" type="select" rowclass="day"/>
127
128 <inputfield formfield="{pdict.CurrentForms.billing.paymentMethods.bml.ssn}" type="input"/>
129
130 <div class="bml-terms-and-conditions form-caption">
131   <iscontentasset aid="bml-tc"/>
132 </div>
133
134 <div class="form-row form-caption">
135   <inputfield formfield="{pdict.CurrentForms.billing.paymentMethods.bml.termsandconditions}" type="checkbox"/>
136 </div>
137
138 </div>
139
140
141 <iscomment>
142   Custom processor
143   -----
144 </iscomment>
145
146 <div class="payment-method <isif condition="{!empty(pdikt.selectedPaymentID) && pdikt.selectedPaymentID!='PayPal'}">paym
147   <!-- Your custom payment method implementation goes here. -->
148   ${Resource.msg('billing.custompaymentmethod','checkout',null)}
149 </div>
150
151 <iscomment>
152   Klarna Payments
153   -----
154 </iscomment>
155
156 <isinclude template="klarnapayments/klarnapaymentblock.isml"/>
157
158 </fieldset>
159 <iselse/>
160 <div class="gift-cert-used form-indent">
161   <isif condition="{pdict.gcPITotal>0}">${Resource.msg('billing.giftcertnomethod','checkout',null)}<iselse/>${Resource.msg
162   <input type="hidden" name="{pdict.CurrentForms.billing.paymentMethods.selectedPaymentMethodID.htmlName}" value="{pdict.Cr
```

Figure 43 Modifications in paymentmethods.isml (cont.)

3.5.1.4. default/checkout/shipping/minishipments.isml

Add the following code:

```
<isinclude template="klarnapayments/modules.isml"/>
```

In the end beginning of the file as shown below:

```

minishipments.isml
1 <iscontent type="text/html" charset="UTF-8" compact="true"/>
2 <isinclude template="util/modules.isml"/>
3 <isinclude template="klarnapayments/modules.isml"/>
4
5 <iscomment>
6     This template renders a summary of all shipments of the basket which is
7     used below the order summary at the right hand side in the checkout
8     process.
9 </iscomment>
10 <isset name="Shipments" value="${pdict.Basket.shipments}" scope="page"/>
11
12 <iscomment>the url to edit shipping addresses depends on the checkout scenario</iscomment>
13 <isset name="editUrl" value="${URLUtils.https('COShipping-Start')}" scope="page"/>
14 <isif condition="${pdict.CurrentForms.multishipping.entered.value}">
15     <isset name="editUrl" value="${URLUtils.https('COShippingMultiple-Start')}" scope="page"/>
16 </isif>
17

```

Figure 44 Modifications in minishipments.isml

Add the following code:

```
<iskpaddresshelper p_address="${shipment.shippingAddress}"/>
```

After `<isminicheckout_address p_address="${shipment.shippingAddress}"/>` as shown below:

```

minishipments.isml
29 <isif condition="${Shipments.size() > 1 && pdict.Basket.productLineItems.size() > 0}"><div class="name">${Resource.msg
30
31 <h3 class="section-header">
32 <isif condition="${shipment.giftCertificateLineItems.size() > 0}">
33     ${Resource.msg('minishipments.shipping','checkout',null)} <span>${Resource.msg('minishipments.giftcertdeliver
34 <iselseif condition="${shipment.custom.shipmentType == 'instore'}">
35     <isset name="editUrl" value="${URLUtils.https('Cart-Show')}" scope="page"/>
36     <a href="${editUrl}" class="section-header-note">${Resource.msg('global.edit','locale',null)}</a>
37     ${Resource.msg('cart.store.instorepickup','checkout',null)}
38 <iselseif condition="${shipment.shippingAddress != null && pdict.Basket.productLineItems.size() > 0}">
39     <a href="${editUrl}" class="section-header-note">${Resource.msg('global.edit','locale',null)}</a>
40     ${Resource.msg('minishipments.shippingaddress','checkout',null)}
41 </isif>
42 </h3>
43
44 <div class="details">
45 <iscomment>
46     render the detail section depending on whether this is a physical shipment with products
47     (shipped to an address) or if this is a gift certificate (send via email)
48 </iscomment>
49 <isif condition="${shipment.giftCertificateLineItems.size() > 0}">
50 <isloop items="${shipment.giftCertificateLineItems}" var="giftCertLI">
51     <div><isprint value="${giftCertLI.recipientName}"/></div>
52     <div><isprint value="${giftCertLI.recipientEmail}"/></div>
53 </isloop>
54 <iselseif condition="${shipment.shippingAddress != null && pdict.Basket.productLineItems.size() > 0}">
55     <isminicheckout_address p_address="${shipment.shippingAddress}"/>
56     <iskpaddresshelper p_address="${shipment.shippingAddress}"/>
57 <isif condition="${!empty(shipment.shippingMethod)}">
58     <div class="minishipments-method">
59         <span>${Resource.msg('order.orderdetails.shippingmethod','order',null)}</span>
60         <span><isprint value="${shipment.shippingMethod.displayName}"/></span>
61     </div>
62 </isif>
63 </div>
64
65 </div>
66 </isif>
67 </isloop>
68 </isif>
69

```

Figure 45 Modifications in minishipments.isml (cont.)

3.5.1.5. default/components/footer/footer_UI.isml

Add Code:

```
<isinclude template="klarnapayments/scripts.isml"/>
```

Before `<script src="{URLUtils.staticURL('/js/app.js')}"></script>` script tag as shown below:

```
27
28 <isinclude template="klarnapayments/scripts.isml"/>
29
30 <script src="{URLUtils.staticURL('/js/app.js')}"></script>
```

Figure 46 Modifications in footer_UI.isml

3.5.1.6. default/components/footer/footer.isml

Add the following code:

```
<!-- Klarna OSM footer -->

<div class="klarna-footer">

    <isinclude template="klarnapayments/modules.isml"/>

    <iskosmfooter />

</div>

<!-- /Klarna OSM footer -->
```

right after the `</footer>` end tag and before `<iscontentasset aid="footer-copy"/>` as shown below:

```
19 </div>
20 <div class="footer-item">
21 <iscontentasset aid="footer-about"/>
22 </div>
23 </div>
24 </footer>
25
26 <!-- Klarna OSM footer -->
27 <div class="klarna-footer">
28 <isinclude template="klarnapayments/modules.isml"/>
29 <iskosmfooter />
30 </div>
31 <!-- /Klarna OSM footer -->
32
33 <iscontentasset aid="footer-copy"/>
34
35 <iscomment>
36 Customer registration can happen everywhere in the page flow. As special tag in the pdict
37 is indicating it. So we have to check on every page, if we have to report this event for
38 the reporting engine.
39 </iscomment>
40 <isinclude template="util/reporting/ReportUserRegistration.isml"/>
41
42 <isinclude template="components/footer/footer_UI"/>
43
```

Figure 47 Modifications in footer.isml

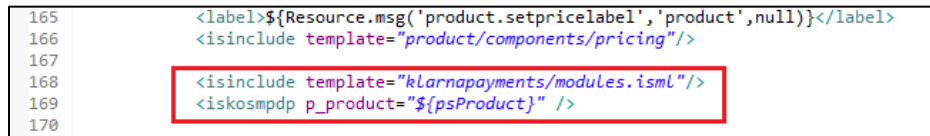
3.5.1.7. default/product/producttopcontentPS.isml

Add the following code:

```
<isinclude template="klarnapayments/modules.isml"/>

<iskosmpdp p_product="${psProduct}" />
```

, right under the `pricing` template include as shown below:



```
165 <label>${Resource.msg('product.setpricelabel','product',null)}</label>
166 <isinclude template="product/components/pricing"/>
167
168 <isinclude template="klarnapayments/modules.isml"/>
169 <iskosmpdp p_product="${psProduct}" />
170
```

Figure 48 Modifications in producttopcontentPS.isml

3.5.1.8. default/product/productcontent.isml

Add the following code:

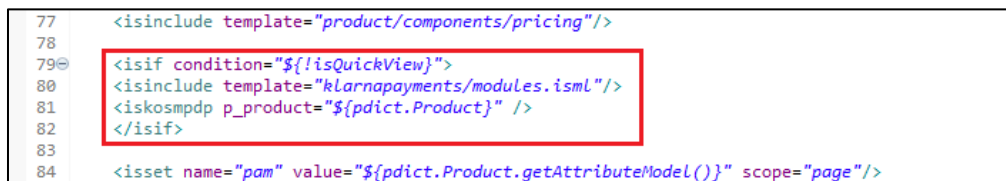
```
<isif condition="${!isQuickView}">

<isinclude template="klarnapayments/modules.isml"/>

<iskosmpdp p_product="${pdict.Product}" />

</isif>
```

, right under the `pricing` template include as shown below:



```
77 <isinclude template="product/components/pricing"/>
78
79 <isif condition="${!isQuickView}">
80 <isinclude template="klarnapayments/modules.isml"/>
81 <iskosmpdp p_product="${pdict.Product}" />
82 </isif>
83
84 <isset name="pam" value="${pdict.Product.getAttributeModel()}" scope="page"/>
```

Figure 49 Modifications in productcontent.isml

3.5.1.9. js/pages/product/variant.js

Add the following code:

```
window.KlarnaOnsiteService = window.KlarnaOnsiteService || [];

window.KlarnaOnsiteService.push({eventName: 'refresh-placements'});
```

in the variation update callback, on line 35

```

29      callback: function () {
30          if (SitePreferences.STORE_PICKUP) {
31              productStoreInventory.init();
32          }
33          image.replaceImages();
34          tooltip.init();
35          window.KlarnaOnsiteService = window.KlarnaOnsiteService || [];
36          window.KlarnaOnsiteService.push({eventName: 'refresh-placements'});
37      }

```

Figure 50 Modifications in variant.js

3.5.1.10. *default/checkout/cart/cart.isml*

Add the following code:

```
<isinclude template="klarnapayments/modules.isml"/>
```

```
<iskosmcart p_lineitemctnr="${pdict.Basket}" />
```

, right after the `<isordertotals>`:

```

820      <div class="cart-order-totals">
821          <isordertotals p_lineitemctnr="${pdict.Basket}" p_totallabel="${Resource.msg('global.estimatedtotal', 'locale', null)}">
822              <isinclude template="klarnapayments/modules.isml"/>
823              <iskosmcart p_lineitemctnr="${pdict.Basket}" />
824          </isordertotals>
825      </div>

```

Figure 51 Modifications in cart.isml

3.5.1.11. *default/components/header/header.isml*

Add the following code:

```
<!-- Klarna OSM header -->
```

```
<isinclude template="klarnapayments/modules.isml"/>
```

```
<iskosmheader />
```

```
<!-- /Klarna OSM header -->
```

At the end of the file as shown below:

```

32 <span>${Resource.msg('global.header.storelocator', 'locale', null)}</span>
33 </a>
34 </li>
35
36 <iscomment>INCLUDE: Customer login information, login, etc. (contains personal information, do not cache)</iscomment>
37 <isinclude url="${URLUtils.url('Home-IncludeHeaderCustomerInfo')}" />
38
39 </ul>
40
41 <iscomment>Country Selector</iscomment>
42 <isinclude template="components/header/countryselector" />
43
44 </nav>
45
46 <iscomment>INCLUDE: Mini-cart, do not cache</iscomment>
47 <div id="mini-cart">
48 <isinclude url="${URLUtils.url('Cart-MiniCart')}" />
49 </div>
50
51 </div><!-- /header -->
52
53 <!-- Klarna OSM header -->
54 <isinclude template="klarnapayments/modules.isml" />
55 <iskosmheader />
56 <!-- /Klarna OSM header -->
57

```

Figure 52 Modifications in header.isml

3.5.1.12. *default/mail/orderconfirmation.isml*

Add the following code:

```

<tr>

  <td style="font-size:12px;font-family:arial;padding:20px 10px;vertical-align:top;">

    <isset name="confirmationAsset"
value="${require('*/cartridge/scripts/util/klarnaHelper').getConfirmationEmailAsset()}" scope="page" />

    <isprint value="${confirmationAsset}" encoding="off" />

  </td>

</tr>

```

Before the `</table>` closing tag as shown:

```

34  ${Resource.msg('order.orderconfirmation-email.storephone','order',null)}
35  >
36
37
38  colspan="2" style="font-size:12px;font-family:arial;padding:20px 10px;vertical-align:top;"
39
40  <table style="background:#ffffff;border:1px solid #999999;width:680px;"
41  <tr>
42  <th style="background:#cccccc;padding:5px 20px;font-size:12px;font-family:arial;text-align:left;">${Resource.msg('confirmation.thankyou','cl
43  </tr>
44  <tr>
45  <td style="font-size:12px;font-family:arial;padding:20px 10px;vertical-align:top;"
46  <p>${Resource.msg('confirmation.message','checkout',null)}</p>
47  <p>${Resource.msg('confirmation.contact','checkout',null)}</p>
48  </td>
49  </tr>
50  <tr>
51  <td style="font-size:12px;font-family:arial;padding:20px 10px;vertical-align:top;"
52  <asset name="confirmationAsset" value="${require('*/cartridge/scripts/util/KlarnaHelper').getConfirmationEmailAsset()}" scope="page" />
53  <isprint value="${confirmationAsset}" encoding="off" />
54  </td>
55  </tr>
56 </table>
57
58 >
59

```

Figure 53 Modifications in orderconfirmation.isml

3.5.1.13. default/components/order/ordrdetailsemail.isml

Add the following code:

```

<iselseif condition="${paymentInstr.paymentMethod.equals('Klarna')}" >
  ${Resource.msg('email.order.reference','klarnapayments',null)}:
  ${Order.custom.kpOrderID} <br />

```

Before the closing `</isif>` tag on line 51 as shown below:

```

39  <div>${Resource.msg('order.orderdetails.paymentmethod','order',null)}</div>
40  <iselseif>
41  <b>${Resource.msg('order.orderdetails.paymentmethods','order',null)}</b>
42  </iselseif>
43
44  <iscomment>Render All Payment Instruments</iscomment>
45  <isloop items="${Order.getPaymentInstruments()}" var="paymentInstr" status="piloopstate">
46  <div><isprint value="${dw.order.PaymentMgr.getPaymentMethod(paymentInstr.paymentMethod).name}" /></div>
47  <isif condition="${dw.order.PaymentInstrument.METHOD_GIFT_CERTIFICATE.equals(paymentInstr.paymentMethod)}>
48  <isprint value="${paymentInstr.maskedGiftCertificateCode}" /><br />
49  <iselseif condition="${paymentInstr.paymentMethod.equals('Klarna')}" >
50  ${Resource.msg('email.order.reference','klarnapayments',null)}: ${Order.custom.kpOrderID} <br />
51  </iselseif>
52  <isminicreditcard card="${paymentInstr}" show_expiration="${false}" />
53  <div>
54  <span class="label">${Resource.msg('global.amount','locale',null)}</span>
55  <span class="value"><isprint value="${paymentInstr.paymentTransaction.amount}" /></span>
56  <br />
57  </div><!-- END: payment-amount -->
58  </isloop>
59  </td>

```

Figure 54 Modifications in orderdetailsemail.isml

3.5.2. Controller Modification

If using a controller based SiteGenesis integration, additionally follow the instructions in this chapter.

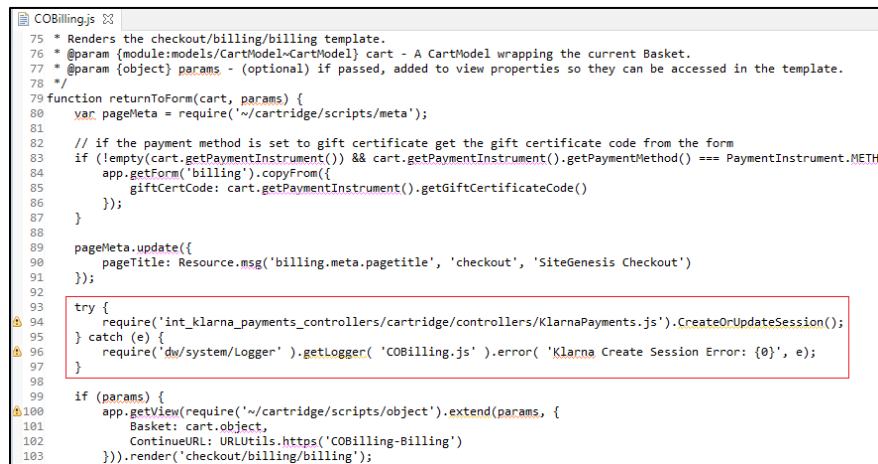
3.5.2.1. COBilling.js

Go to COBilling.js controller, `returnToForm()` method and add the following code block:

```
try {
  require('int_klarna_payments_controllers/cartridge/controllers/KlarnaPayments.
  js').CreateOrUpdateSession();
} catch ( e ) {

  require( 'dw/system/Logger' ).getLogger( 'COBilling.js' ).error( 'Klarna
  Create Session Error: {0}', e );
}
```

after `pageMeta.update()` function (see screen shot below)



The screenshot shows a code editor for COBilling.js. The code is as follows:

```
75 * Renders the checkout/billing/billing template.
76 * @param {module:models/CardModel~CardModel} cart - A CardModel wrapping the current Basket.
77 * @param {object} params - (optional) if passed, added to view properties so they can be accessed in the template.
78 */
79 function returnToForm(cart, params) {
80   var pageMeta = require('~/cartridge/scripts/meta');
81
82   // if the payment method is set to gift certificate get the gift certificate code from the form
83   if (!empty(cart.getPaymentInstrument()) && cart.getPaymentInstrument().getPaymentMethod() === PaymentInstrument.METHOD_GIFT_CERTIFICATE) {
84     app.getForm('billing').copyFrom({
85       giftCertCode: cart.getPaymentInstrument().getGiftCertificateCode()
86     });
87   }
88
89   pageMeta.update({
90     pageTitle: Resource.msg('billing.meta.pagetitle', 'checkout', 'SiteGenesis Checkout')
91   });
92
93   try {
94     require('int_klarna_payments_controllers/cartridge/controllers/KlarnaPayments.js').CreateOrUpdateSession();
95   } catch (e) {
96     require('dw/system/Logger').getLogger('COBilling.js').error('Klarna Create Session Error: {0}', e);
97   }
98
99   if (params) {
100     app.getView(require('~/cartridge/scripts/object').extend(params, {
101       Basket: cart.object,
102       ContinueURL: URLUtils.https('COBilling-Billing')
103     })).render('checkout/billing/billing');
```

Figure 55 Modifications in COBilling.js

In the method `resetPaymentForms()`, add the command for the three conditions:

```
cart.removePaymentInstruments(cart.getPaymentInstruments('Klarna'));
```

```

349  */
350  function resetPaymentForms() {
351    //
352    var cart = app.getModel('Cart').get();
353    //
354    var status = Transaction.wrap(function () {
355      if (app.getForm('billing').object.paymentMethods.selectedPaymentMethodID.value.equals('PayPal')) {
356        app.getForm('billing').object.paymentMethods.creditCard.clearFormElement();
357        app.getForm('billing').object.paymentMethods.bml.clearFormElement();
358      }
359      cart.removePaymentInstruments(cart.getPaymentInstruments(PaymentInstrument.METHOD_CREDIT_CARD));
360      cart.removePaymentInstruments(cart.getPaymentInstruments(PaymentInstrument.METHOD_BML));
361      cart.removePaymentInstruments(cart.getPaymentInstruments('Klarna'));
362    } else if (app.getForm('billing').object.paymentMethods.selectedPaymentMethodID.value.equals(PaymentInstrument.METHOD_CREDIT_CARD)) {
363      app.getForm('billing').object.paymentMethods.bml.clearFormElement();
364    }
365    cart.removePaymentInstruments(cart.getPaymentInstruments(PaymentInstrument.METHOD_BML));
366    cart.removePaymentInstruments(cart.getPaymentInstruments('PayPal'));
367    cart.removePaymentInstruments(cart.getPaymentInstruments('Klarna'));
368    } else if (app.getForm('billing').object.paymentMethods.selectedPaymentMethodID.value.equals(PaymentInstrument.METHOD_BML)) {
369      app.getForm('billing').object.paymentMethods.creditCard.clearFormElement();
370    }
371    if (!app.getForm('billing').object.paymentMethods.bml.ssn.valid) {
372      return false;
373    }
374    //
375    cart.removePaymentInstruments(cart.getPaymentInstruments(PaymentInstrument.METHOD_CREDIT_CARD));
376    cart.removePaymentInstruments(cart.getPaymentInstruments('PayPal'));
377    cart.removePaymentInstruments(cart.getPaymentInstruments('Klarna'));
378    }
379    return true;
380  });
381

```

Figure 56 Modifications in COBilling.js (cont.)

In method `handlePaymentSelection(cart)`, go to line 441 and add the following code block:

```

if (app.getForm('billing').object.paymentMethods.selectedPaymentMethodID.value !== 'Klarna') {
  require('int_klarna_payments_controllers/cartridge/controllers/KlarnaPayments.js').CancelAuthorization();
}

```

```

431  //
432  //
433  //
434  if (empty(PaymentMgr.getPaymentMethod(app.getForm('billing').object.paymentMethods.selectedPaymentMethodID.value).paymentProcessor)) {
435    result = {
436      error: true,
437      MissingPaymentProcessor: true
438    };
439    //
440    //
441    if (!result) {
442      if (app.getForm('billing').object.paymentMethods.selectedPaymentMethodID.value !== 'Klarna') {
443        require('int_klarna_payments_controllers/cartridge/controllers/KlarnaPayments.js').CancelAuthorization();
444      }
445      result = app.getModel('PaymentProcessor').handle(cart.object, app.getForm('billing').object.paymentMethods.selectedPaymentMethodID.value);
446    }
447    return result;
448  }
449  //
450  /**
451  * Gets or creates a billing address and copies it to the billingaddress form. Also sets the customer email address

```

3.5.2.2. COSummary.js

Go to COSummary.js controller, `submit()` method, and add the following code block:

```
try {  
  
require('int_klarna_payments_controllers/cartridge/controllers/KlarnaPayments.  
js').Redirect();  
  
} catch( e ) {  
  
    require( 'dw/system/Logger' ).getLogger( 'COSummary.js' ).error( 'Klarna  
Redirect Error: {0}', e );  
  
}
```

before `showConfirmation(placeOrderResult.Order)` (see screen shot below)

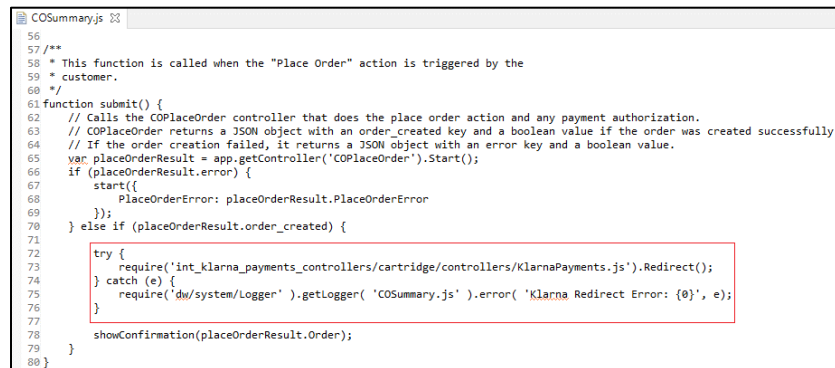


Figure 57 Modifications in COSummary.js

3.5.2.3. OrderModel.js

Go to OrderModel.js, placeOrder method and add the following code block:

```
if ( session.privacy.KlarnaPaymentsFraudStatus === 'PENDING' ) {  
    try {  
        require('int_klarna_payments_controllers/cartridge/controllers/KlarnaPayments.js').  
        PendingOrder(order);  
    } catch( e ) {  
        require( 'dw/system/Logger' ).getLogger( 'OrderModel.js' ).error( 'Klarna  
Payments Pending Order Error: {0}', e );  
    }  
    return;  
}
```

before `var placeOrderStatus = OrderMgr.placeOrder(order);` (see screen shot below)

```

OrderModel.js
7
8 /* API Includes */
9 var AbstractModel = require('./AbstractModel');
10 var Order = require('dw/order/Order');
11 var OrderMgr = require('dw/order/OrderMgr');
12 var Resource = require('dw/web/Resource');
13 var Status = require('dw/system/Status');
14 var Transaction = require('dw/system/Transaction');
15
16 /**
17  * Place an order using OrderMgr. If order is placed successfully,
18  * its status will be set as confirmed, and export status set to ready.
19  * @param {dw.order.Order} order
20  */
21 function placeOrder(order) {
22
23     if(session.privacy.KlarnaPaymentsFraudStatus === 'PENDING') {
24         try {
25             require('int_klarna_payments_controllers/cartridge/controllers/KlarnaPayments.js').PendingOrder(order);
26         } catch(e) {
27             require('dw/system/Logger').getLogger('OrderModel.js').error('Klarna Payments Pending Order Error: {0}', e);
28         }
29         return;
30     }
31
32     var placeOrderStatus = OrderMgr.placeOrder(order);
33     if (placeOrderStatus === Status.ERROR) {
34         OrderMgr.failOrder(order);
35         throw new Error('Failed to place order.');
```

Figure 58 Modifications in OrderModel.js

3.6. External Interfaces

All requests are done through Klarna's REST API and encrypted using SHA-256 with the shared secret provided by Klarna. Only HTTPS is allowed. JSON is used across all communications.

The full reference guide, along with the resource structure for requests & responses, can be found in the developer portal - <https://developers.klarna.com/api/#payments-api>

4. Testing

Klarna has a set of testing credentials and triggers that can be used.

Please, refer to the following URL: <https://developers.klarna.com/documentation/testing-environment/>

5. Operations, Maintenance

5.1. Data Storage

5.1.1. System Object Extensions

5.1.1.1. Order

Parameter Name	Attribute ID	Description
Klarna Payments Order ID	kpOrderID	The Klarna payments Order ID for Klarna payment method selected by customer
VCN Brand	kpVCNBrand	Klarna Payments virtual card scheme name
VCN Holder	kpVCNHolder	Klarna Payments virtual card holder name
VCN Card ID	kpVCNCardID	Klarna Payments Virtual Card - Card ID
VCN PCI Data	kpVCNPCIData	Klarna Payments Virtual Card PCI Data in encrypted format
VCN Initialization Vector	kpVCNIV	Klarna Payments Virtual Card Initialization Vector
VCN AES Key	kpVCNAESKey	Klarna Payments Virtual Card AES Key
Is VCN Used	kpIsVCN	True if virtual card is enabled & used for payment of the order, otherwise false

Table 1 Order Attributes

5.1.1.2. Order Payment Instrument

Parameter Name	Attribute ID	Description
Klarna Payment Category ID	klarnaPaymentCategoryID	ID of Klarna payment category
Klarna Payment Category Name	klarnaPaymentCategoryName	Name of Klarna payment category

Table 2 Order Payment Instrument Attributes

5.1.1.3. Payment Transaction

Parameter Name	Attribute ID	Description
Fraud Status	kpFraudStatus	Klarna Payments order fraud status

Table 3 Payment Transaction Attributes

5.1.1.4. Site Preferences

The site custom preferences have been extended with a new group called “*Klarna_Payments*”. The table below describes the preferences within that group:

Parameter Name	Attribute ID	Description
Auto-capture	kpAutoCapture	When enabled “Yes”, a full order capture will be attempted automatically. The standalone order management API capture request will include total order amount value for “captured_amount”. Default value is “No”
Klarna Payments Service Name	kpServiceName	The service name used for the current site
Send product_url and image_url	sendProductAndImageURLs	If set to true, product_url and image_url fields will be included in the Klarna session and order API calls. This enhances shopper experience post purchase. Default value is “Yes”

Parameter Name	Attribute ID	Description
Merchant Reference 2 Mapping	merchant_reference2_mapping	<p>The field from SCC order (basket) object that is mapped to merchant_reference2 field from klarna API request.</p> <p>Has to be one of the class attributes of SCC LineltemCtnr.</p> <p>Note that for complex data structures result may vary.</p> <p>Note: Merchant Reference 1 value is always set to the SCC order ID</p>
Border Color Preference	kpColorBorder	CSS (hex value) color set for Border in Klarna Payments iFrame
Border Selected Color Preference	kpColorBorderSelected	CSS (hex value) color set for selected element Border in Klarna Payments iFrame
Button Color Preference	kpColorButton	CSS (hex value) color set for Button in Klarna Payments iFrame
Button Text Color Preference	kpColorButtonText	CSS (hex value) color set for Button text in Klarna Payments iFrame
Checkbox Color Preference	kpColorCheckbox	CSS (hex value) color set for Checkbox in Klarna Payments iFrame
Checkbox Checkmark Color Preference	kpColorCheckboxCheckmark	CSS (hex value) color set for checkbox checked(selected) in Klarna Payments iFrame
Details Color Preference	kpColorDetails	CSS (hex value) color set for details in Klarna Payments iFrame
Header Color Preference	kpColorHeader	CSS (hex value) color set for Header in Klarna Payments iFrame
Link Color Preference	kpColorLink	CSS (hex value) color set for link in Klarna Payments iFrame
Text Color Preference	kpColorText	CSS (hex value) color set for text in Klarna Payments iFrame
Secondary Text Color Preference	kpColorTextSecondary	CSS (hex value) color set for secondary text in Klarna Payments iFrame
Border Radius Preference	kpRadiusBorder	Value (in pixels) of the border radius to be used in Klarna Payments iFrame

Parameter Name	Attribute ID	Description
Attachments	kpAttachments	Toggle (Yes/No) for the inclusion of attachments when creating an order. Specific to inclusion of EMD (customer_account_info, other_delivery_address) when applicable. Default is “No”.
Not available message on billing page	kpNotAvailableMessage	The Klarna Payment not available message on billing page. JSON string holding country code and corresponding message string. For example: <pre>{ "GB": "Klarna Payment not available", "default": "Klarna Payment not available" }</pre> Note: This is deprecated and will be removed in next releases!
Virtual Card Number Enabled	kpVCNEnabled	If this option is set to “Yes”, Klarna settlement request will generate a Virtual Card Number for every Klarna order. Note: the option will only work if VCN private/public keys are configured properly as below and public key shared in advance with Klarna
VCN Public Key ID	kpVCNkeyId	UUIDv4 value corresponding to the key pair. Shared with Klarna respectively for Production & Playground (test) env.
VCN Private Key	vcnPrivateKey	SSL private key used only to decode Virtual Card information (used with kpVCNEnabled). Refer to section 9.3 Decrypt VCN Card Details

Parameter Name	Attribute ID	Description
VCN Public Key	vcnPublicKey	SSL public key used with Virtual Card integration (used with kpVCNEnabled). Shared with Klarna and stored here for reference.
VCN Settlement Retry Enabled	kpVCNRetryEnabled	If set to "Yes", SFCC will retry the VCN settlement once again in case of service error. Default is "No"
Promotion Price Taxation	kpPromoTaxation	Only use "Based on Adjusted Price" value if you have enabled the corresponding value in "Merchant Tools > Site Preferences > Promotions > Discount Taxation" and use gross taxation. Default: Based on Price
Hide Payment Methods on Deny	kpRejectedMethodDisplay	If set to value other than "No", the Klarna payment method options on the checkout will be greyed out or not displayed to customer in the current view when Klarna authorization request is rejected in the response (.i.e hard reject - "show_form" and "approved" values are both "false")

Table 4 Site Preferences

5.1.2. Custom Objects

5.1.2.1. KlarnaCountries

The respective object is dynamically selected based on the request locale country, e.g., SFCC site with locale **"de_DE"** or **"en_DE"** will use the **"DE"** custom object. In cases when the request locale country can't be dynamically resolved (i.e. with "default" SFCC locale) – attribute **"klarnaLocale"** can be utilized to pass the proper locale to Klarna. For all other cases, this field can be left blank and will not be taken into consideration.

Even if you have locales that are not supported by Klarna Payments, we recommend creating a corresponding entry in the custom object for that locale. Thus, on the billing page of the unsupported locale you will have the Klarna Payments widget showing an appropriate message.

The custom objects store data such as Klarna default locale, service credential IDs and Klarna Payments placement data keys to ensure that Klarna Payments integration is correctly configured.

Note: The same custom object is used by Klarna Checkout cartridge integration!

Select All	ID	Name	Type	Attribute Settings	Values	
<input type="checkbox"/>	uuid	UUID	String	*	0	Edit
<input type="checkbox"/>	country	Country Code	String	*	0	Edit
<input type="checkbox"/>	creationDate	Creation Date	Date+Time	*	0	Edit
<input type="checkbox"/>	credentialID	Service Credential ID	String		0	Edit
<input type="checkbox"/>	klarnaLocale	Klarna Locale	String		0	Edit
<input type="checkbox"/>	lastModified	Last Modified	Date+Time	*	0	Edit
<input type="checkbox"/>	osmCartEnabled	Cart Placement Tag Enabled	Boolean		0	Edit
<input type="checkbox"/>	osmCartTagId	Cart Placement Tag ID	String		0	Edit
<input type="checkbox"/>	osmLibraryUrl	Library URL	String		0	Edit
<input type="checkbox"/>	osmPDPEnabled	PDP Placement Tag Enabled	Boolean		0	Edit
<input type="checkbox"/>	osmPDPTagId	PDP Placement Tag ID	String		0	Edit
<input type="checkbox"/>	osmUCI	On-site messaging UCI	String		0	Edit

Figure 59 KlarnaCountries Attributes

The table below describes attributes of the **KlarnaCountries** custom object:

Attribute Name	Attribute ID	Description
Country Code	country	Two-letter country code
On-site Messaging Data Default Locale	klarnaLocale	Fallback, if the request locale can't be dynamically resolved, i.e., when using "default" SFCC locale
Service Credential ID	credentialID	The ID of service credentials for this locale.
On-site messaging Data Client ID	osmUCI	The Klarna On-site Messaging "data-client-id" applicable for a given country
Cart Placement Tag Enabled	osmCartEnabled	To enable Cart Placement for a given locale.
Cart Placement Tag ID	osmCartTagId	The Klarna On-site Messaging "data-key" of placement applicable for Cart Page for a given locale.

Attribute Name	Attribute ID	Description
PDP Placement Tag Enabled	osmPDPEnabled	To enable PDP Placement for a given locale.
PDP Placement Tag ID	osmPDPTagId	The Klarna On-site Messaging “data-key” of placement applicable for Product Display page for a given locale.
Header Placement Tag Enabled	osmHeaderEnabled	To enable Klarna Header Placement in a given storefront
Header Placement Data Key	osmHeaderTagId	The Klarna On-site Messaging “data-key” of placement applicable for the Header
Footer Placement Tag Enabled	osmFooterEnabled	To enable Klarna footer Placement in a given storefront
Footer Placement Data Key	osmFooterTagId	The Klarna On-site Messaging “data-key” of placement applicable for the footer
Info Page Placement Tag Enabled	osmInfoPageEnabled	To enable Klarna Info Page Placement in a given storefront
Info Page Placement Data Key	osmInfoPageTagId	The Klarna On-site Messaging “data-key” of placement applicable for the Info Page
URL to On-Site Messaging Library URL	osmLibraryUrl	URL to On-Site Messaging Library URL, applicable for testing and production

Table 5 KlarnaCountries Attributes

***Note:** The data-client-id and data-key values used in the OSM placements are available in the Klarna Merchant Portal (Europe/US (CA included)/Oceania) within the On-site Messaging App. When selecting the data-key values, ensure that the filter is set to the right country and language.*

5.1.3. Library

In addition to the configurations, the following 2 library assets will be added:

- **“footer-about”** – Updated OOTB asset including link to Klarna OSM dedicated page in the footer.
- **“klarna-email-info”** – Asset containing links to review the Klarna Payment information. Used in the confirmation email sent to the customers.

5.1.4. Services

An HTTP service “**klarna.http.defaultendpoint**” has been added with “**klarna.http.service**” profile and service credentials for each country (described in **KlarnaCountries** custom object).

5.2. Logs

The integration includes the following logs:

- Service communication logs – starts with “service-klarna-***”. These logs contain every request and response to the Klarna endpoints. Personal information, i.e. emails & names required for the Klarna API calls are masked in the logs.
- Custom errors and debug info are logged under “customerror-***”, “custodebug-***” & “custominfo-***” files depending on the case.

5.3. Availability

Cartridge functionality will be dependent on the availability of the Klarna API service. Current Klarna operational status can be viewed here - <http://status.klarna.com/>

5.4. Failover/Recovery Process

If Klarna API is not available, Klarna is not presented as a payment option. In case of any failure within the Klarna API, contact Klarna for support.

5.5. Support

Klarna’s service center is responsible for handling operational tasks. The service center is divided into the following two teams: Customer Service and Merchant Support.

5.5.1. Customer Service

A customer service workshop can be conducted during the implementation process before going live to align the operational processes and ensure customer satisfaction. Klarna provides all customers with the possibility to log into Klarna App via website: <https://app.klarna.com/login> or download the Klarna App (free) on a mobile (Android/iOS). The customers can contact support, view their statements, pay for their purchase, track delivery updates, and prolong the due dates if they have chosen to pay after delivery.

5.5.2. Merchant Support

Reporting core SFCC functionality issues in the Klarna cartridge technical integration – please contact commercecloud@klarna.com

For production issue related to Klarna API availability, merchant representative should reach their Klarna Account manager after reviewing the current operational status at <http://status.klarna.com/>. Report the problem in Production (Post Go-live) if you have a suspicion about degraded performances or issues with Klarna's service. The Klarna contact would then be able to report this internally to the incident management team who have established routines to handle and resolve reported incidents. The Klarna contact may request additional information from the individual reporting the problem to help internal team ascertain and identify the issue. The KAM may also advise the merchant to follow the updates on the status page if it is a known incident with on-going updates.

Pre-requisite information to be provided by merchant when reporting incident to help with speedy investigation and resolution:

- Merchant's affected MID or market
- Impact and examples of customer orders (order_id or Klarna session_id if available)
- Screenshots, timeframe, additional information as required

6. User Guide

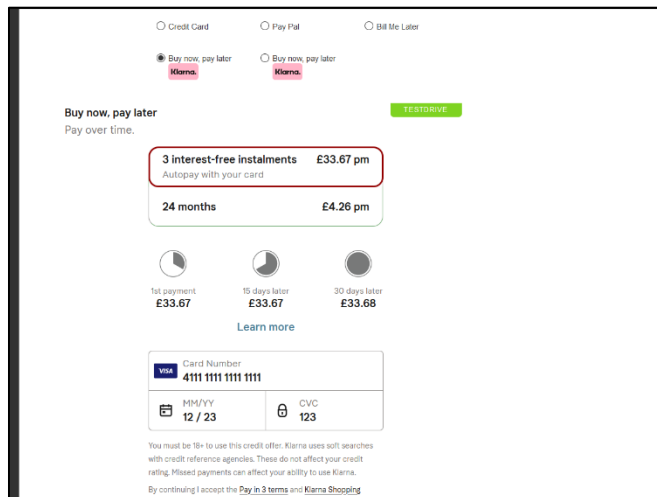
6.1. Roles, Responsibilities

There are no recurring tasks required by the merchant. Once configurations are set up, the functionality runs on demand.

6.2. Storefront Functionality

When Klarna has been setup, the Klarna Payments options and iframe widgets will be shown on the billing step. All the SFCC OOTB checkout functionality remains in place, such as but not limited to cart updates during checkout, checkout with applied coupon(s) code(s), checkout with applied product level promotion, checkout with applied order level promotion, checkout with applied shipping level promotion, checkout with applied order level promotion with bonus product.

Select one of Klarna's payment options as the payment method on billing step of checkout process and click the “Next: Place Order” button:



The screenshot displays the Klarna payment interface on a checkout page. At the top, there are three radio button options: "Credit Card", "Pay Pal", and "Bill Me Later". Below these, there are two "Buy now, pay later" options, each with a Klarna logo. The first option is selected and highlighted with a red box. Below this, the text "Buy now, pay later" and "Pay over time." is shown. A green "Try it now" button is visible. The main section shows "3 interest-free instalments" for £33.67 pm, with a note "Autopay with your card". Below this, a table shows the installment schedule: 24 months, £4.26 pm. A timeline shows three payments: 1st payment (£33.67), 15 days later (£33.67), and 30 days later (£33.68). A "Learn more" link is present. At the bottom, there is a card number field (4111 1111 1111 1111), an expiry date field (12 / 23), and a CVC field (123). A disclaimer at the bottom states: "You must be 18+ to use this credit offer. Klarna uses soft searches with credit reference agencies. These do not affect your credit rating. Missed payments can affect your ability to use Klarna. By continuing I accept the Pay in 3 terms and Klarna Shopping".

Figure 60 Payment Options on Checkout

Depending on the payment method selected and the region, you will see one of Klarna's popup windows to provide the details. Follow the steps on the screen:

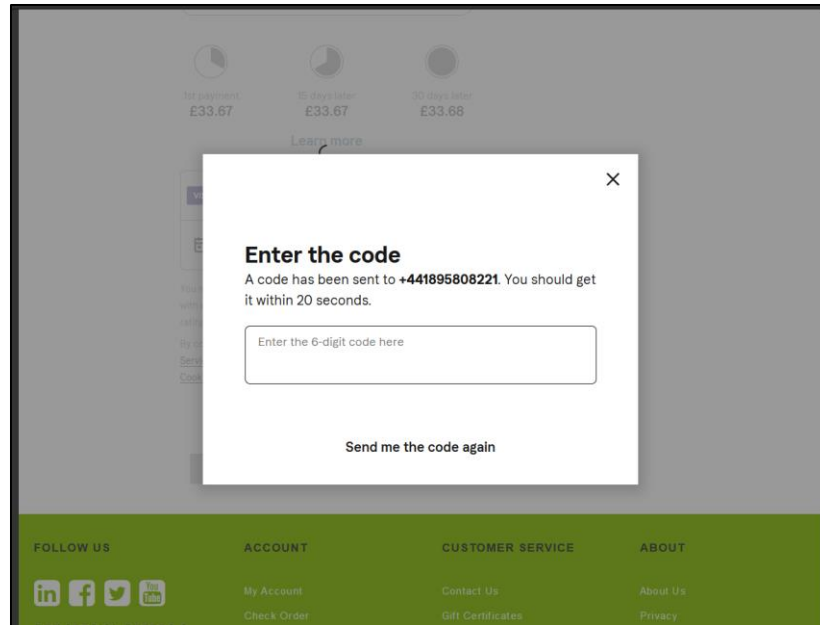


Figure 61 Klarna Popup Screen

On the Review step click on “Place Order” button:

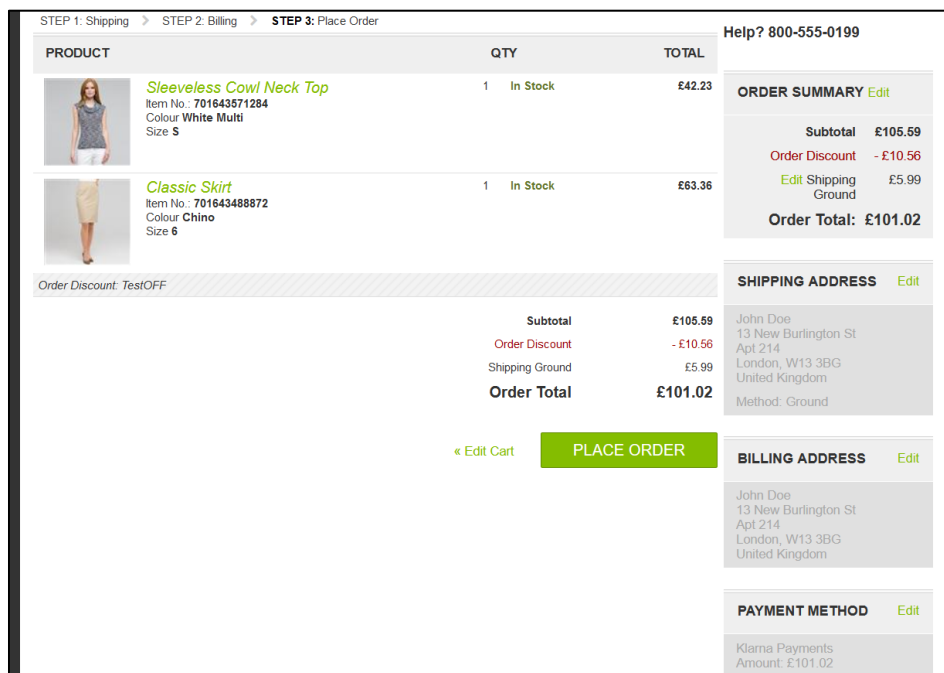


Figure 62 Payment Review Screen

The customer's browser is sent to the "redirect_url" and immediately thereafter shown the Commerce Cloud Order Confirmation page.

If you have questions about your order, we're happy to take your call (800-555-0199) Monday - Friday, 8AM - 8PM

Order Number: 00055607

Order Placed: 4 Mar 2021

PAYMENT METHOD
Klarna Payments
Amount: £101.02

BILLING ADDRESS
John Doe
13 New Burlington St
Apt 214
London, W13 3BG
United Kingdom
Phone: 01895808221

SHIPMENT NO. 1

SHIPPING STATUS:
Not Shipped
METHOD:
Ground

ITEM
Sleeveless Cowl Neck Top
ITEM NO: 701643571284
COLOUR White Mugs
SIZE 5
Classic Skirt
ITEM NO: 701643488872
COLOUR China
SIZE 6

QTY
1
1

PRICE
£42.23
£63.36

PAYMENT TOTAL
Subtotal £105.59
Order Discount - £10.56
Shipping Ground £5.99
Order Total: £101.02

SHIPPING TO
John Doe
13 New Burlington St
Apt 214
London, W13 3BG
United Kingdom
01895808221

CREATE ACCOUNT
Creating an account is easy. Just fill out the form below and enjoy the benefits of being a registered customer.
• First Name
John
• Last Name
Doe
• Email
john@doe.com
• Confirm Email
• Password
8 - 255 characters
• Confirm Password
Create Account

[Return to Shopping](#)

Figure 63 Order Confirmation Page

The newly created order can be inspected in Business Manager:

You're using the new Search service.

This page allows you to search for orders by order number. Select **Advanced** to use more search options. Select **By Number** to search by order number. Entered text is treated as case-sensitive; substring matching isn't supported.

Order Search

Order Number: [Find](#)

Number	Order Date	Site	Created By	Registration Status
00055607	3/4/21 3:00:52 pm Etc/UTC	SiteGenesisGlobal	Customer	Unregistered
00055606	3/4/21 2:58:25 pm Etc/UTC	SiteGenesisGlobal	Customer	Unregistered
00055605	3/4/21 2:52:27 pm Etc/UTC	SiteGenesisGlobal	Customer	Registered
00055604	3/4/21 12:51:34 pm Etc/UTC	SiteGenesisGlobal	Customer	Registered
00055603	3/4/21 12:49:03 pm Etc/UTC	SiteGenesisGlobal	Customer	Registered
00055602	3/4/21 11:08:50 am Etc/UTC	SiteGenesisGlobal	Customer	Unregistered
00055505	3/3/21 5:58:33 pm Etc/UTC	SiteGenesisGlobal	Customer	Unregistered
00055405	3/2/21 8:41:39 am Etc/UTC	SiteGenesisGlobal	Customer	Registered
00055404	3/2/21 8:35:37 am Etc/UTC	SiteGenesisGlobal	Customer	Registered
00055403	3/2/21 7:46:22 am Etc/UTC	SiteGenesisGlobal	Customer	Registered

Showing 1 - 10 of 506 items

Show items

Figure 64 Orders List in BM

Klarna Payments order id can be inspected in the Attributes tab of the order:

Merchant Tools > Ordering > Orders > Order: 00055607(SiteGenesisGlobal)

General **Attributes** Payment Notes History

Attributes for Order '00055607'

On this page you can edit the attributes of the order. Fields with a red asterisk (*) are mandatory. Click **Apply** to save changes. Click **Reset** to

Klarna Payments	
Klarna Payments Order ID:	77aa252d-70fa-2719-957b-b047a82b3343
Is VCN Used:	<input type="checkbox"/>
VCN Card ID:	

<< Back to List

Figure 65 Order Attributes

Payment method details can be inspected on the Payment tab of the order, and it should be Klarna:

Merchant Tools > Ordering > Orders > Order: 00055607(SiteGenesisGlobal)

General Attributes **Payment** Notes History

Payment Information for Order '00055607'

Order Total:	£101.02
Amount Paid:	£0.00
Balance Due:	£101.02

Invoice Number:	00242007
Payment Status:	Paid

Payment Method:	Klarna Processor: KLARNA_PAYMENTS Transaction: 77aa252d-70fa-2719-957b-b047a82b3343 Amount: £101.02 Klarna Payment Category ID: pay_over_time Klarna Payment Category Name: Buy now, pay later Fraud Status: ACCEPTED
------------------------	---

<< Back to List

Figure 66 Order Payment Detail

Order can be further inspected in Klarna Merchant Portal:

- EU: eu.portal.klarna.com
- US: us.portal.klarna.com
- OC: us.portal.klarna.com

#BW6DKM8G Captured

£101.02

Merchant reference 1
00055607
Edit

Merchant reference 2
4f8acec8fc30bebb330570ab69
Edit

Created
Mar 4, 2021, 5:00 PM

Expires
Apr 1, 2021, 3:00 AM

Merchant ID
K500726

Customer

Shipping address




John Doe
13 New Burlington St
Apt 214
London
W13 3BG
GB
Tel
01895808221
Email
john@doe.com
Edit shipping address

Billing address

Additional Info

Order lines (3)

Refund Print packing slip

	Item / Reference	Qty	Unit price	Discount	Tax	Amount
<input type="checkbox"/>	 Sleeveless Cowl Neck Top 701643571284	1	42.23	4.22	20% 6.34	38.01 Captured
<input type="checkbox"/>	 Classic Skirt 701643488872	1	63.36	6.34	20% 9.50	57.02 Captured
<input type="checkbox"/>	 Ground GBP001	1	5.99	0.00	20% 1.00	5.99 Captured

PAYMENT DETAILS

Initial Payment Method
Pay later in parts
VISA 411111*****1111
Resend statement

ORDER TOTAL

Captured

£101.02

Refunded

£0.00

Not Captured

£0.00

CUSTOMER BILLED

£101.02

Activity Log

Mar 4, 2021
7 minutes ago

5:00 PM
Captured: £101.02
Via API

5:00 PM
Order placed: £101.02
By Klarna

Klarna. Copyright © 2005-2021 Klarna Bank AB (publ). All rights reserved

Terms & Conditions

Figure 67 Klarna Portal Order View

7. Known Issues

The LINK Cartridge has no known issues.

8. Release History

Version	Date	Changes
18.1.0		Initial release of Klarna Payments
19.1.0		Added SFRA version
19.1.1		Updated VCN to use the newest API version
19.1.2		Fix auto capture for the pipelines cartridge
19.1.4		New country locales added. Minor bug fixes. Cartridge templates and forms updated for latest SFRA.
19.1.5		Added additional verification for all notifications. Minor fixes around the configuration objects. Added Canadian support. Documentation updates.
19.1.6		New country locales added. Updated VCN to store encrypted card details
21.1.0		Fixes around discounts taxation & VCN error handling. Added VCN improvements, additional On-site Messaging placements, BOPIS support. New IT, CA, FR & NZ country locales. Removed acknowledge call. Documentation updates.

9. Additional Information

9.1. Klarna API Information

The Klarna Payments API is accessible through different endpoint based on the context of the webstore. There are separate endpoints for testing and live and the Klarna merchant identifier (MID) is configured for respective markets in regions (EU, NA, OC) by endpoint.

9.1.1. *Live Environment*

The API for the European production environment can be found at

- <https://api.klarna.com/>

The API for the North America production environment can be found at

- <https://api-na.klarna.com/>

The API for the Oceania production environment can be found at

- <https://api-oc.klarna.com/>

9.1.2. *Testing Environment*

The API for the European Playground/testing environment can be found at

- <https://api.playground.klarna.com/>

The API for the North America Playground/testing environment can be found at

- <https://api-na.playground.klarna.com/>

The API for the Oceania Playground/testing environment can be found at

- <https://api-oc.playground.klarna.com/>

9.2. Generate Key Pair and Key Id for Virtual Card Settlements (VCN)

The recommend RSA keypair size of 4096 bits. This key pair must be associated with a key_id (UUIDv4). The public key must be shared in JWK format with Klarna contact. Note that for production and playground, the key_id and keypair combination shared are different and must be configured prior to testing/go-live of the virtual card product.

To generate an RSA keypair with a 4096-bit private key you can use the following **openssl** command:

```
openssl genpkey -algorithm RSA -out private_key.pem -pkeyopt  
rsa_keygen_bits:4096
```

To extract the public key from an RSA keypair, you can use the following **openssl** command:

```
openssl rsa -pubout -in private_key.pem -out public_key.pem
```

In the folder where you have executed the above commands two new files will be created - **public_key.pem** and **private_key.pem**.

The contents of the files should look something like:

public_key.pem

```
-----BEGIN PUBLIC KEY-----  
MIICIjANBgkqhkiG9w0BAQEFAAOCAg8AMIICGKCAgEAAoNYG7l2G8nZa+22oBYZk  
tV228lw3UE9W04oxfknJtKEdHn84x55ULt8KQTh9NVtdeKC8nTfTgyvMt/GNca18  
xuZV/lGYDftKt85hbV5EjOum+StAIufEXv1BX7nMOMclKyWm9kp2kbqd88mFIX63  
KV94OoNEXcNatRDFYR+qz53+ifadDQtQ1slVNStdroCZDJ1+LxtBy9V+BdmsBK1E  
RLsKh/JLXyWE24FJKV+z00s7TQkdWW/5ET12OGQYZsWo1yqgi9HplNvrisve8vWP  
xaL4m8iZ3I/9yYdg7yANQbTxSJcbbRCgaaagPo30CNxeqU6qafY5g8vY3E52CoXH  
DdO4Us1XlqcuYIDhqaDzey6W+b8m755xLi+rqQyM4PBWL0J0dM3FVid8+4YKILex  
3AKBFciqRCMHSOGaEeyrXKTjlAsghr9RS8PifvQRrL440cHzqw2vX0DvpjSWcmUJ  
tW4wUq5RNSsobrnxnVmoV6fj1z67Q/1P+15Ie+oowdahR5ztVqJl0+2PNoX4I5VDs  
/Pkz3f8wWVc3Mp2oNT244o+/NIiYRfPFaJJx7JAgrcvZt2nFamY4QApXLFJCpgEM  
wYucE4AH4gJKsh3KZbxRERrrO72bL2rxvWqBp/0h7DcMsV9sQs4BvxxIl6CF506F  
ThzmclaKLBAYd5LALiXiPfkCAwEAAQ==  
-----END PUBLIC KEY-----
```

private_key.pem

```
-----BEGIN PRIVATE KEY-----  
MIIEJQIBADANBgkqhkiG9w0BAQEFAASCCSswggknAgEAAoICAQCglgbuXYbydlr7  
bagFhmS1XbbyXdDQT1Y7ijF+Scm0oR0efzjHn1Qu3wpBOH01W114oLydN9ODK8y3  
8Y0JrXzG5lX+UZgN+0q3zmFtXkSM66b5K0Ai58Re+UFfucw4xzUrJab2SnaRup3z  
yYUhfrcpX3g6g0Rdw1q1EMVhH6rPnf6J9p0NC1DWyVU1K12ugJkMnX4vG0HL1X4F  
2awErUREuwqH8ktfJYTbgUkpX7PTSztNCR1Zb/kRPXY4ZBhmxaJXKqCL0emU2+uK  
y97y9Y/FovibyJncj/3Jh2DvIA1BtPFilxttEKBppqA+jfQI3F6pTqpp9jmDy9jc  
TnYKhccN07hSyVfWpy5ggOGpoPN7Lpb5vybvnnEuL6upDIzg8FYvQnR0zcVWJ3z7  
hgogt7HcAoEVyKpEIwdI4ZoR7KtcpOOUcyCGv1FLw+J+9BGsvjjRwfOrDa9fQO+m  
NJZyZQmlbjBSrlE1KyhuvGdWahXp+PXPrtD/U/6Xkh76ijBlqFhN0lWomU77Y82h  
fgjlUOz8+TPd/zBZVzcynag1Pbjij780iLJF88VoknHskCCty9m3acUCZjhAClcs  
UkKmAQzBi5wTgAfiAkqyHcplvFERGus7vZsvavG9aoGn/SHsNwyxX2xCzgG/HEiX  
oIXnToVOHOZyVoosEDJ3ksAuJeI9+QIDAQABAoICACRkaUsUNI22RB3yEPu3DiCP  
pO6v+QAeA4gTW+GUDqR9dCZLaSCZ7bhxVVOuoX4qPzslO6hjUmOyzG6upFgVPk+P  
HNQfyEUZoC148Eib9OziAXUN2URMpv1KbwVm+BO814X8zguai7uru0PHTG1oy677  
4Ct1OknxAxxHQDIaxT6XJFo5SA4EinUfnZ2Bo3/xry/QjxW/mCK0GwDd4PNp9TGM  
FPTv2SgdSDOWzGQlOH5N3owuzMpI8NV6z74wv+i5Ptv41Dzu8WhyXpiYSsk00SRK
```

```
HPC68j2bAzTPghp5aSZ9976SGm2SPonJXyboXdiHbI/osdyqDxeIT3iB9GmrHX/i
kHPGJCh7fRZvqj39Hc+IxYjabwW3rDeDIPB7ab9z1KLF4z1D6AZOKCPyTaDRdQ1Q
eDi7LwDmk7NHEPrmF/nIcguQdqB1bmFO2zEs0TOe6y4uBMndRsbQprTNSMudBkrA
lNaYVSTQ1Z0Y/8DZDpGcyS1OnJv74F15uDjKN6/ov991mZ1JrZ+V2sdS3EDUlmvP
6thQKwI7Ln6h+ApHtWUG1NmvQe5gJE0qAeJ9b45clUzIRUwhVmEp8NoIJh0kAjaN
d4lk7xy9ZRDUY5yekPeYrJPShjsHAYEoktJIjRufI2UUq3uxNjjICoQcOVGFNDIS
YTTTPwpulpmC0C+rh2fgBAoIBAQRultRARvtc2JKhVOUyZk88zd9kvrI6fNiyKmi
HgiWf7qkTPD9xhOQWDw3iWRfQAD+YkgV5MCBO8wp8oO8GESOCI+XZWEExOcPT0Vfj
PZHiQrTFnlfg/+fAO14xLf3j3ED4YQXhHOKI3xoLknQx/EydLoctxgkkgpWlrsA7
DwdSag1/0sBvaHY27ogAfdimHdaKZ5OAE4a9klqP3xVZBuOe8Sd65unBavUJLDuv
ikeNmKSVgWlsm55/729Jir63USHF76It+vE1cdZ+vKg5vYotsQgPzvNBmUO/E8Gj
zMXQRfQfVEdLNXEX0rCupTkw1G6AGTwQc/NPzyr/LTpLe6UBAoIBAQDEUjTiG11V
hf7WjdG3gctrRlr+mYapQHgXdVLx2QSaQUYid+0QXK11YfJlsRB6nwa+OED83RfP0
lIFqxpzudSLPmoDuIBT7D15c/aleyKs/siUusP8QVDXk6OAR84XSytC35sIRV7pE
VMuBL91jfkQ0Lf/PreslK/ki6Yvwwp4qrHK6/f9TgciHclYtf+/oti4ky6GJgfmP
fmuCqjxmUKbXXFPd5RbL2THGOowilb8zDLjf3R1bj1QFqogAk6H9hp2V0VZLiJHp
UWM3z3zxDWeDaqJ08sHuk/rA9QpsVTu8IGTQsxdj8JwluN1Q+YZiOuPiSENBqPzT
V3exexzo3sD5AoIBAGU3qEyPoJz1+9D1SaI8LW2CABzlq4z9g84ABAZOs1xX5q7W
x1PinZyDSQSRXg1B13jt29ZdIR79ygnQlglYOBjcvtgVQHPuafk3R1BQbbCh+vaI
9dn/tUxMGqhnhunKaby1rovJHfdqnPpKwzNAjYUqaGkJ822xhmmke/fEyAanIPa4
stDRvIPEWPTLx5xcOCdx13khpKSnkgRvaLEfpwkVX7Vr7hK/2OSFaYTNmrzXYBQ7
c6D/9d3Oo4nLb/mu+Tq67S19t53Qg/GEgTfkpuRoVPi0KyhUnKKCGWlBMZLTwyIG
S9eTFDKoJ0cSTGipjW7bPua93wZ8eEbRABpf4QECggEANNhQBeEJ0aCdBVHtdrEI
crDaa8X0WlaJi5dol4hYCRajaKsfHAF/QfdgMQVxHwUC5YG4En/Q+DAVWhGWYpXD
RhC3zeFy5FVszyk0sx/fAO1KGvRn5BRW4YRR9GMRzbjsT+RcruBnckdE9ERXGpX9
c/JB3rxZBIt+oIiFM8yfWktMwsrmNKtFuDftvJeok4KejycFF4eWDqsf828xjPT+
xA/FP4CQD1UqkcpmuFSiAwXo6LXVY7NTS0nKMiUnTLkLlTIHtLn09+9jmNapWRP
Tc+hZUuHKLpI8DHFmX2j87LgkFD05eD5lynY4RgZtU1W1C1RdVYwoA72WB7knEaB
uQKCAQAH9s67P/7fFX9dfEans3PHU4nGjD8dJ8eoNQ6DhBMydZpGWI5ZUeEBZDRk
0cBOeRs5BOcS43Em9kETpzawyCwxmnwzl+CzoPzMqCTw9tXomF9HG6RJ9XBdJfGA
ALAwCd4bASxmFM6guSP5GKnZ9aY3tR3tWWDfr7f9z8wOewzzpPclwRh009fPe4TC
NXoEm1MELJVeUieDSLKZgjgCw8WHGqLItONpA0/fwSM2gIcxETVV7qx3aPuJzCVh
LQZ0BLQk3UMKsWDdpzeBdiERE66NagV9k2Xe7SY9EY2vymaq761ilxlvlprT27qp
240LDJawqM0IraKmdCvWjofWSaOU
-----END PRIVATE KEY-----
```

9.3. Decrypt VCN Card Details

To decrypt the virtual card details stored on order level and authorize the credit card processor you can use the following code snippet. You can find more information about the decryption process [here](#).

```
var OrderMgr = require( 'dw/order/OrderMgr' );
var Cipher = require( 'dw/crypto/Cipher' );
var Encoding = require( 'dw/crypto/Encoding' );
var Site = require( 'dw/system/Site' );

var Order = OrderMgr.getOrder( "order_id" );
var VCNPrivateKey = Site.getCurrent().getCustomPreferenceValue(
'vcnPrivateKey' );
var cipher = new Cipher();

var keyEncryptedBase64 = Order.custom.kpVCNAESKey;
var keyEncryptedBytes = Encoding.fromBase64( keyEncryptedBase64 );
var keyDecrypted = cipher.decryptBytes( keyEncryptedBytes, VCNPrivateKey,
"RSA/ECB/PKCS1PADDING", null, 0 );
```

```
var keyDecryptedBase64 = Encoding.toBase64( keyDecrypted );
var cardDataEncryptedBase64 = Order.custom.kpVCNPCIData;
var cardDataEncryptedBytes = Encoding.fromBase64( cardDataEncryptedBase64 );
var cardDecrypted = cipher.decryptBytes( cardDataEncryptedBytes,
keyDecryptedBase64, "AES/CTR/NoPadding", Order.custom.kpVCNIV, 0 );

var cardDecryptedUtf8 = decodeURIComponent( cardDecrypted );
var cardObj = JSON.parse( cardDecryptedUtf8 );
var expiryDateArr = cardObj.expiry_date.split( "/" );

// Retrieve ecnrypted card details
var cardPAN = cardObj.pan, cardCVV = cardObj.cvv,
    cardExpiryMonth = expiryDateArr[0], cardExpiryYear = expiryDateArr[1];
```