INT\_KLARNA\_CHECKOUT

*22.1.0*



Table of Contents

[1. Summary 3](#_Toc24108247)

[2. Component Overview 4](#_Toc24108248)

[Functional Overview 4](#_Toc24108249)

[Use Cases 4](#_Toc24108250)

[Limitations, Constraints 19](#_Toc24108251)

[Compatibility 19](#_Toc24108252)

[Privacy, Payment 19](#_Toc24108253)

[3. Implementation Guide 20](#_Toc24108254)

[Setup of Business Manager 20](#_Toc24108255)

[Configuration 20](#_Toc24108256)

[Custom Code 22](#_Toc24108257)

[Template modification 22](#_Toc24108258)

[Pipeline modifications 24](#_Toc24108259)

[Controller modifications 28](#_Toc24108260)

[Klarna Checkout Locale 30](#_Toc24108261)

[External Interfaces 30](#_Toc24108262)

[4. Testing 30](#_Toc24108263)

[5. Operations, Maintenance 31](#_Toc24108264)

[Data Storage 31](#_Toc24108265)

[Availability 31](#_Toc24108266)

[Support 31](#_Toc24108267)

[6. User Guide 31](#_Toc24108268)

[Business Manager 31](#_Toc24108269)

[Site Preferences 31](#_Toc24108270)

[Storefront Functionality 33](#_Toc24108271)

[7. Known Issues 35](#_Toc24108272)

[8. Release History 35](#_Toc24108273)

# Summary

Klarna is one of leading providers of invoice-based payment solutions for eCommerce. This option allows Consumers to shop using only top-of-mind information and pay after they’ve received the goods. The hassle is taken out of buying on mobile devices creating a simple option and increasing mobile conversion rates. Klarna maximizes the conversion rate by making it safer and simpler for Customers to shop. All potential risk for both the Customer and the merchant is assumed by Klarna, so there is always a guarantee of payment. The int\_klarna\_checkout cartridge makes it easy to integrate Klarna Checkout with the Commerce Cloud storefront.

# **Component Overview**

## Functional Overview

The int\_klarna\_checkout cartridge provides a number of tools that can be used to accelerate the integration of Klarna Checkout with a Commerce Cloud site. This cartridge uses Site current locale and Custom Objects to determine which checkout and modules to display to the customer.

Pipelines, templates and scripts are provided to help invoke the Klarna Checkout UI and use Klarna’s API to communicate the customer’s SCC cart back to Klarna.

Klarna Checkout Overview - <https://developers.klarna.com/en/us/kco-v2/klarna-checkout-overview-v1/>

## Use Cases

This cartridge can be used to take customers through the Klarna Checkout flow on their storefront. It can be used on the desktop storefront and mobile responsive storefront. These are examples of use cases that allow you to customize the buying experience with Klarna Checkout for your needs.

1. Virtual Card Network as an alternative to standard order management

The VCN payment option can be enabled through the site preference as shown below:



This option is disabled by default. However, if standard order management is not a reasonable option for a Klarna integration, instead Klarna’s VCN, or virtual card network, can be enabled. When a customer places an order, the order is first booked in SFCC. If the 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 ACCEPTED. Once an order has been accepted by Klarna, the merchant platform creates a VCN settlement, per the merchant card services (MCS) API.

Once a settlement has been created, the merchant platform has up to 3 hours to authorize the virtual credit card. Then, once the order has been fulfilled, the card funds should be captured. While Klarna is the original payment method of the order, the order will be settled with a credit card, not directly by Klarna.

* In order to use the VCN option the merchant should:
* Enable VCN option in Site Preferences as shown above
* Enter the value of ‘kcVCNkeyId’ in custom site preferences. This is unique identifier of the public key that will be used for encryption of the data. It is provided by Klarna.
* Generate a 4096 bit RSA key pair. 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). As shown below:



**How to generate a 4096 bit RSA key pair:**

In order 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***

In order 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-----

MIICIjANBgkqhkiG9w0BAQEFAAOCAg8AMIICCgKCAgEAoNYG7l2G8nZa+22oBYZk

tV228lw3UE9WO4oxfknJtKEdHn84x55ULt8KQTh9NVtdeKC8nTfTgyvMt/GNCa18

xuZV/lGYDftKt85hbV5EjOum+StAIufEXvlBX7nMOMc1KyWm9kp2kbqd88mFIX63

KV94OoNEXcNatRDFYR+qz53+ifadDQtQ1slVNStdroCZDJ1+LxtBy9V+BdmsBK1E

RLsKh/JLXyWE24FJKV+z00s7TQkdWW/5ET12OGQYZsWo1yqgi9HplNvrisve8vWP

xaL4m8iZ3I/9yYdg7yANQbTxSJcbbRCgaaagPo30CNxeqU6qafY5g8vY3E52CoXH

DdO4UslX1qcuYIDhqaDzey6W+b8m755xLi+rqQyM4PBWL0J0dM3FVid8+4YKILex

3AKBFciqRCMHSOGaEeyrXKTjlAsghr9RS8PifvQRrL440cHzqw2vX0DvpjSWcmUJ

tW4wUq5RNSsobrxnVmoV6fj1z67Q/1P+l5Ie+oowdahR5ztVqJlO+2PNoX4I5VDs

/Pkz3f8wWVc3Mp2oNT244o+/NIiyRfPFaJJx7JAgrcvZt2nFAmY4QApXLFJCpgEM

wYucE4AH4gJKsh3KZbxRERrrO72bL2rxvWqBp/0h7DcMsV9sQs4BvxxIl6CF506F

ThzmclaKLBAyd5LALiXiPfkCAwEAAQ==

-----END PUBLIC KEY-----

**private\_key.pem**

-----BEGIN PRIVATE KEY-----

MIIJQQIBADANBgkqhkiG9w0BAQEFAASCCSswggknAgEAAoICAQCg1gbuXYbydlr7

bagFhmS1XbbyXDdQT1Y7ijF+Scm0oR0efzjHnlQu3wpBOH01W114oLydN9ODK8y3

8Y0JrXzG5lX+UZgN+0q3zmFtXkSM66b5K0Ai58Re+UFfucw4xzUrJab2SnaRup3z

yYUhfrcpX3g6g0Rdw1q1EMVhH6rPnf6J9p0NC1DWyVU1K12ugJkMnX4vG0HL1X4F

2awErUREuwqH8ktfJYTbgUkpX7PTSztNCR1Zb/kRPXY4ZBhmxajXKqCL0emU2+uK

y97y9Y/FovibyJncj/3Jh2DvIA1BtPFIlxttEKBppqA+jfQI3F6pTqpp9jmDy9jc

TnYKhccN07hSyVfWpy5ggOGpoPN7Lpb5vybvnnEuL6upDIzg8FYvQnR0zcVWJ3z7

hgogt7HcAoEVyKpEIwdI4ZoR7KtcpOOUCyCGv1FLw+J+9BGsvjjRwfOrDa9fQO+m

NJZyZQm1bjBSrlE1KyhuvGdWahXp+PXPrtD/U/6Xkh76ijB1qFHnO1WomU77Y82h

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/nIcguQdqbIbmFO2zEs0TOe6y4uBMndRsbQprTNSMUdBkrA

lNaYVSTQ1Z0Y/8DZDpGcyS1OnJv74F15uDjKN6/ov991mZ1JrZ+V2sdS3EDUlmvP

6thQKwI7Ln6h+ApHtWUG1NmvQe5gJE0qAeJ9b45clUzIRUwhVmEp8NoIJh0kAjaN

d4lk7xy9ZRDUY5yekPeYrJPShjsHAyEoktJIjRufI2UUq3uxNjjICoQcOVGfNDIS

YTTPwpu1pmC0C+rh2fgBAoIBAQDRultRArvtc2JKhVOUyZk88zd9kvrI6fNiyKmi

HgiWf7qkTPD9xhOQWDw3iwRFQAD+YkgV5MCBO8wp8oO8GEsOCI+XZWExOcPT0Vfj

PZHiQrTFnlfG/+fAO14xLf3j3ED4YQXdHOKI3xoLknQx/EydLoctxgkkpgWLrsA7

DwdSAg1/0sBvaHY27ogAfdimHdaKZ5OAe4a9k1qP3xVZBuOe8Sd65unBavUJLDuv

ikeNmkSVgW1sm55/729JIr63USHF76It+vE1cdZ+vKg5vYotsQgPzvNBmUO/E8Gj

zMXQRfqfvEDlNXEX0rCupTkw1G6AGTwQc/NPzyr/LTpLe6UBAoIBAQDEUjTiG11V

hf7WjdG3gctRlr+mYapQHgXdVLx2QSaqUYid+0QXK11YfJlsRB6nwa+OED83RfP0

lIFqxpzudSLPmoDuIBT7Dl5c/aleyKs/siUusP8QVDXk6OAR84XSytC35sIRV7pE

VMuBL91jfkQ0Lf/PreslK/kI6Yvwwp4qrHK6/f9TgciHclYtf+/oti4ky6GJgfmP

fmuCqjxmUKbXXFPd5RbL2THGOowilb8zDLjf3RlbjlQFqogAk6H9hp2V0VZLiJHp

UWM3z3zxDWeDaqJ08sHuk/rA9QpsVTu8IGTQsxdj8JwluN1Q+YZiOuPiSENBqPzT

V3exexzo3sD5AoIBAGU3qEyPojz1+9D1SaI8LW2CABzlq4z9g84ABAZOslxX5q7W

x1PinZyDSQSRXg1B13jt29ZdIR79ygnQlg1YOBjcvtgVQHPuafk3RlBQbbCh+vaI

9dn/tUxMGqhnhunKaby1rovJHfdqnPpKwzNAjYUqaGkJ822xhmmke/fEyAanIPa4

stDRvIPEWPTLx5xcOCdx13khpKSnkgRvaLEfpwkVX7Vr7hK/2OSFaYTNmrzXYBQ7

c6D/9d3Oo4nLb/mu+Tq67S19t53Qg/GEgTfkpuRoVPi0KyhUnKKCGWlBMZLTwyIG

S9eTFDKoJ0cSTGipjW7bPua93wZ8eEbRABpf4QECggEANNhQBeEJ0aCdBVHtdrEI

crDaa8X0W1aJi5dol4hYCRajaKsfHAF/QfdgMQVxHwUC5YG4En/Q+DAVWhGWYpXD

RhC3zeFy5FVszyk0sx/fAOlKGvRn5BRW4YRR9GMRzbjsT+RcruBnckdE9ERXGpX9

c/JB3rxZBIt+oIiFM8yfWKtMwsrmNKtFuDftvJeok4KejycFF4eWDqsf828xjPT+

xA/FP4CQD1UqkcpmuFSIwAwXo6LXVY7NTS0nKMiUnTLkLlTIHtLnO9+9jmNapWRP

Tc+hZUuHKlpI8DHFmX2j87LgkFD05eD5lynY4RgZtU1W1C1RdVYwoA72WB7knEaB

uQKCAQAH9s67P/7fFX9dfEans3PHU4nGjD8dJ8eoNQ6DhBMydZpGWI5ZUeEBZDRk

0cBOeRs5BOcS43Em9kETpzawyCwxmnwzl+CzoPzMQcTw9tXomF9HG6RJ9XBdJfGA

ALAwCd4bASxmFM6guSP5GKnZ9aY3tR3tWWDfr7f9z8wOewzzpPclwRh009fPe4TC

NXoEm1MELJVeUieDSLKZgjgCw8WHGqLItONpA0/fwSM2gIcxETVV7qx3aPuJzCVh

LQZoBLQk3UMKsWDdpzeBdiERe66NAgVk92Xe7SY9EY2vymaq761i1x1vlprT27qp

240LDJawqM0IraKmdCvWjofWSaOU

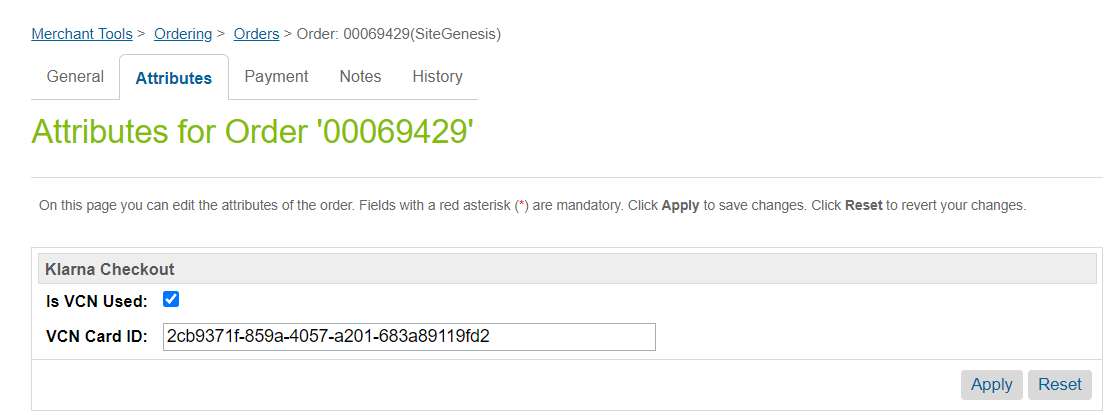
-----END PRIVATE KEY-----

* Finally, you need to send the generated public key to Klarna. It will be used to encrypt the VCN pan and csc on our side. After confirmation from Klarna side that the key has been successfully added to your profile you would be able to use VCN option.
* Authorize the virtual credit card

The virtual credit card details are stored as order level custom attributes, and can be accessed through the Order object as follows:

* Order.custom.kcVCNBrand – virtual credit card brand;
* Order.custom.kcVCNHolder - virtual credit card holder;
* Order.custom.kcVCNCardID- Virtual credit card ID (unique UUIDv4 value);
* Order.custom.kcVCNPCIData – Virtual credit card encrypted data;
* Order.custom.kcVCNIV – Virtual credit card initialization vector;
* Order.custom.kcVCNAESKey – Virtual credit card AES key;
* Order.custom.kcIsVCN – Boolean property indicating whether VCN is used.

These attributes can be inspected in the BM order details in the attributes tab as shown below. Out of the box only the card ID is visible as shown below.



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.

Graphical user interface, text, application, email

Description automatically generated

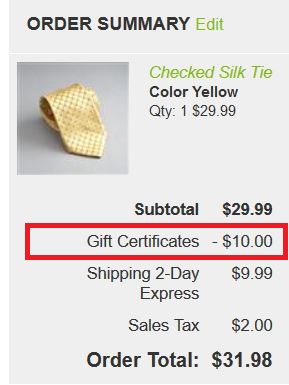
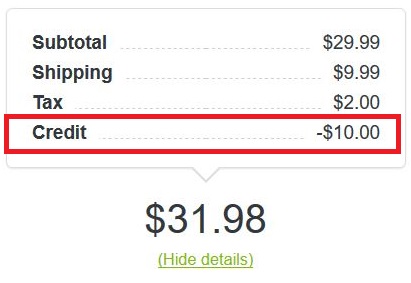
It is up to the merchant to use the credit card details as described above to authorize the virtual credit card. As an example, authorization is done with calling BASIC\_CREDIT-Authorize as below



You need to add your own logic in KLARNA\_CHECKOUT-Authorize in the place of BASIC\_CREDIT-Authorize call node.

2. Paying with Gift Certificates

The cartridge support split payment with GC within the checkout. However as the Klarna Checkout iFrame displays as order total the transaction amount only for the Klarna Checkout payment instrument and this can be misleading for the customers, we suggest some template modification on order totals modules, so we can display the amount paid with GC on a separate row as negative value as is displayed in the Klarna Checkout.

To do this, you need add the following code in

**default/components/order/ordertotals.isml**

<isscript>

var giftDiscount = new dw.value.Money(0, LineItemCtnr.currencyCode);

for each(var gcp in LineItemCtnr.getGiftCertificatePaymentInstruments()) {

giftDiscount = giftDiscount.add(gcp.paymentTransaction.amount);

}

</isscript>

<isif condition="${!empty(giftDiscount) && giftDiscount.value > 0.0}">

<tr class="discount">

<td>${Resource.msg('giftcertpurchase.giftcertificates','checkout',null)}

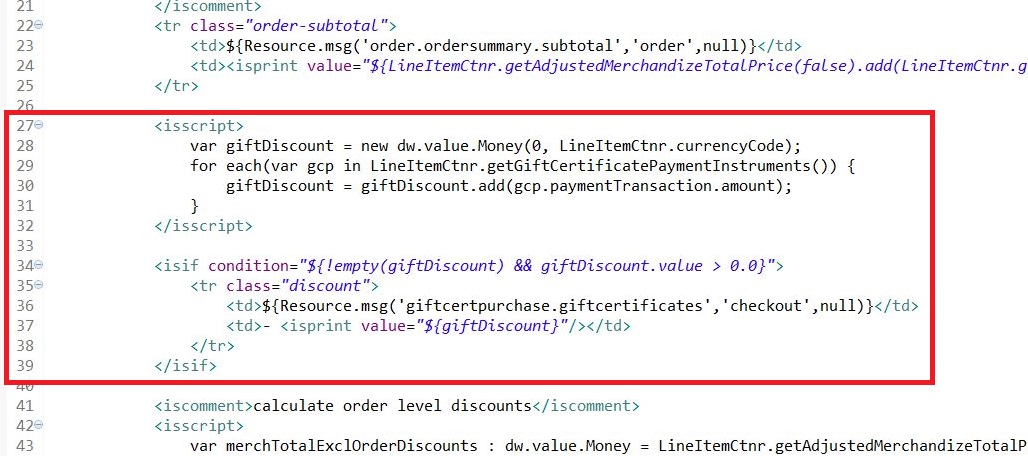
</td>

<td>- <isprint value="${giftDiscount}"/></td>

</tr>

</isif>

After the order subtotal line.



In order to be consistent through the site we can also change the order totals display in My account order history and order confirmation mail. To do so change the following:

**default/account/orderhistory/orders.isml**

After <isif condition="${shipmentloopstate.first}">

Change:

<span class="value"><isprint value="${order.object.totalGrossPrice}"/></span>

With:

<isscript>importScript("checkout/Utils.ds"); </isscript>

<span class="value"><isprint value="${calculateNonGiftCertificateAmount(order.object)}"/></span>



**default/components/order/ordertotalsemail.isml**

Add

<isscript>

var giftDiscount = new dw.value.Money(0, LineItemCtnr.currencyCode);

for each(var gcp in LineItemCtnr.getGiftCertificatePaymentInstruments()) {

giftDiscount = giftDiscount.add(gcp.paymentTransaction.amount);

}

</isscript>

<isif condition="${!empty(giftDiscount) && giftDiscount.value > 0.0}">

<tr>

<td style="font-size:12px;font-family:arial;padding:20px 10px;vertical-align:top; color:#990000; padding:2px 0;">${Resource.msg('giftcertpurchase.giftcertificates','checkout',null)}</td>

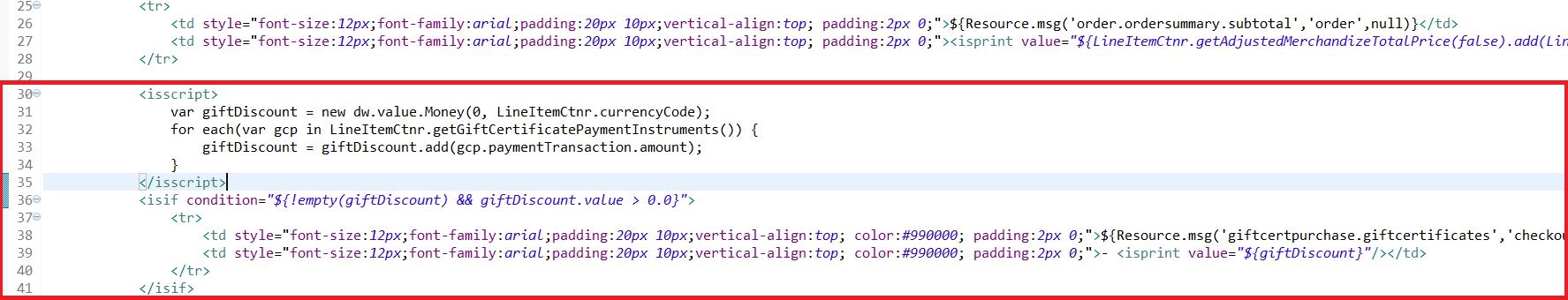
<td style="font-size:12px;font-family:arial;padding:20px 10px;vertical-align:top; color:#990000; padding:2px 0;">- <isprint value="${giftDiscount}"/></td>

</tr>

</isif>

After

<td style="font-size:12px;font-family:arial;padding:20px 10px;vertical-align:top; padding:2px 0;"><isprint value="${LineItemCtnr.getAdjustedMerchandizeTotalPrice(false).add(LineItemCtnr.giftCertificateTotalPrice)}"/></td></tr>



3. Additional payment methods in Klarna Checkout

This feature enables you to use Klarna Checkout for additional payment methods that are currently not offered by Klarna. This means that you will be able to offer one great checkout experience, regardless of how the consumer wants to pay.

**Note**: Integrating new payment methods requires additional integration work and since this payment method will be processed and finalized outside Klarna checkout you will also need to cater for some of the payment flows, both frontend and backend. See examples of additional integration steps below:

To use this feature, you will need an agreement that defines what payment methods that you will use.

Since external payments take place outside Klarna checkout, no user data will be shared for this purchase. You/the external payment method will need to cater for that so all relevant user data to finalize the purchase (e.g. address, name, etc.) is collected.

You will need to host your own “thank you page” since this purchase will take place outside Klarna checkout.

New backend integration towards the external payment method will be required.

Given that this is an external payment, Klarna's buyer protection will not apply for your consumers.

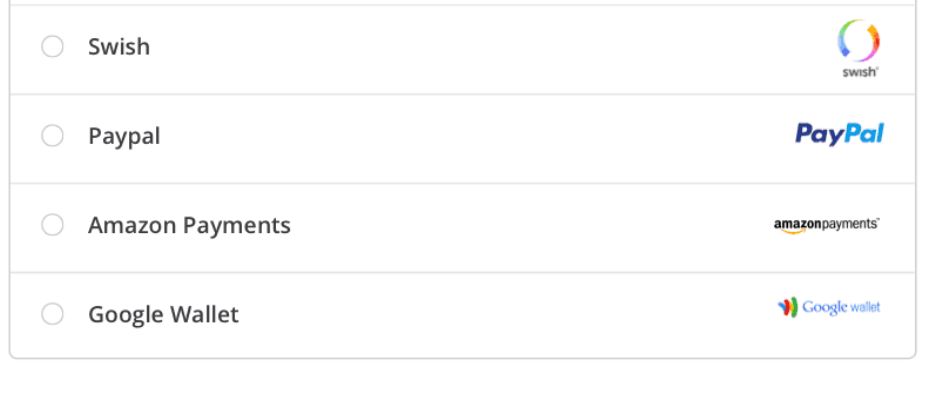
The upside of using external payment methods in Klarna checkout is that you will be able to offer a smoother user experience regardless of payment alternative.

**How does it work?**

When the Checkout is loaded, you have the option to load additional payment methods. "Wallets" are shown as buttons in the bottom of KCO and available from the moment KCO loads. Clicking the button will make the customer leave KCO and be redirected to an external payment window to complete the purchase, e.g. to Paypal.

All payment methods including wallets will be shown in the payment selection list together with our standard offering.

Please note that when a consumer is using a payment method not offered by default, no purchase order is created by Klarna. Klarna doesn’t know whether the consumer eventually finishes the purchase after leaving the checkout. Klarna doesn’t send any kind of confirmation emails or handle any money. All processing of the order is the merchant’s responsibility.



**How to use**

There are **External Payment Methods**  and **External Checkouts** Site Preferences, through which you can use this feature.

The data there is entered in the following format:

[

{

"name": "Cash on delivery", // Mandatory

"redirect\_uri": "https://…", // Mandatory. HTTPS. Page for completing the purchase.

"image\_uri": "https://…", // Optional. HTTPS. Exact size: 69x24px

"fee": 1450 // Optional fee added to the order.

"description": "Lorem.." // Optional. 500 character limit. Links can be set with the Markdown syntax [Text](URL)

}

]

The payment methods are classified as either External Payment Methods or External Checkouts. For External Payment Methods, the user will enter in their consumer data via Klarna Checkout in order to checkout with an external payment method. For External Checkouts, you don’t need Klarna Checkout to collect data about the consumer (e.g. Paypal).

Note: For PayPal you have both external payment method or external checkout to make sure that they appear in all payment flows. You are responsible for the completion of the purchase flow including the e.g. address collection and confirmation page.

**List of External Payment Methods supported in KCO**

We support a wide range of external payment methods in Klarna Checkout. For the list of supported methods in your region, please contact your integration sales representative.

4. Extra merchant data

In some cases, Klarna requires additional information regarding the customer and the purchase in order to make a correct risk assessment.

Examples:

Airline Tickets - when creating an order to sell an airline ticket, information about the passenger and itinerary to be booked should be provided to Klarna.

Event sales merchant - when creating an order for selling a ticket to an event, information about the event should be provided to Klarna.

Recurring orders - when creating an order for subscriptions or recurring payments, subscription and customer account information should be provided to Klarna.

**Create the extra merchant data**

*The cartridge supports sending additional information on the customers past purchase history when turned on for a merchant. This is controlled with the* ***kcAttachments*** *site preference. When turned ON, the integration supports capturing customer and other data. The default setting is OFF.*

The data send to Klarna is customizable through a hook. An example hook contents are provided in int\_klarna\_checkout\_sfra/scripts/EMD/KlarnaCheckoutBuildEMD\_hooks.js. The merchant can customize the contents of the BuildEMD function. Current Basket or Order object can be accessed through args. LineItemCtnr. 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](https://developers.klarna.com/api/" \l "payments-api__create-a-new-orderattachment__body)

*If the example KlarnaCheckouitBuildEMD\_hooks.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}$"

}

}

}

}

}

}

Example data:

{

"extra\_merchant\_data": {

"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"

}]

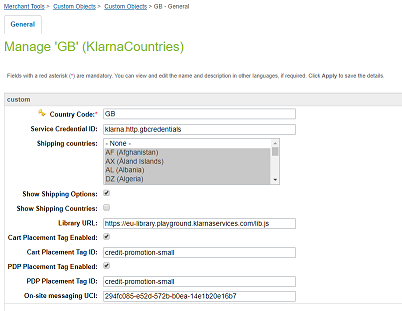
}

}

}

4. Configure On-Site Messaging

Klarna On-Site Messaging is configured per-market via the **KlarnaCountries** custom object. To configure the OSM settings for a market, you must visit “*Merchant Tools – Custom Object Editor*” and search for KlarnaCountries custom object. Then, select one of the available markets for which you need the OSM configured.



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

In *On-site messaging UCI,* please input the On-Site Messaging Client Identifier.

If You want the Placement tag enabled for the Cart Page, the “*Cart Placement Tag ID*” must be filled in with a valid placement tag id and the “*Cart Placement Tag Enabled*” must be set to Yes.

If You want the Placement tag enabled for the PDP Page, the “*PDP Placement Tag ID*” must be filled in with a valid placement tag id and the “*PDP Placement Tag Enabled*” must be set to Yes.



Klarna On-Site Messaging Enabled on Cart Page



Klarna On-Site Messaging Enabled on PDP Page

For more information, please refer to the Klarna Resources Portal:

<https://developers.klarna.com/resources/on-site-messaging>

## Limitations, Constraints

As of this release, Klarna Checkout is currently only offered in Sweden, Norway, Finland, Germany, Netherlands, Austria, the UK and the United States.

The int\_klarna\_checkout cartridge doesn’t support the multi-shipping functionality in SIteGenesis.

In order for the functionality to work well, make sure you provide country and language for all the site’s locales.

## Compatibility

Based on Commerce Cloud 19.10 and SiteGenesis 103.1.3

## Privacy, Payment

If a customer is logged in, their profile data is accessed. This includes their first name, last name, gender (Germany only), email, and preferred address. No other personal data or payment data is accessed.

The cartridge supports sending additional information on the customers past purchase history when turned on for a merchant. This is controlled with the kcAttachments site preference. When turned ON, the integration supports capturing customer and other data. The default setting is OFF.

**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 shop environment. This maybe 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 their validity date

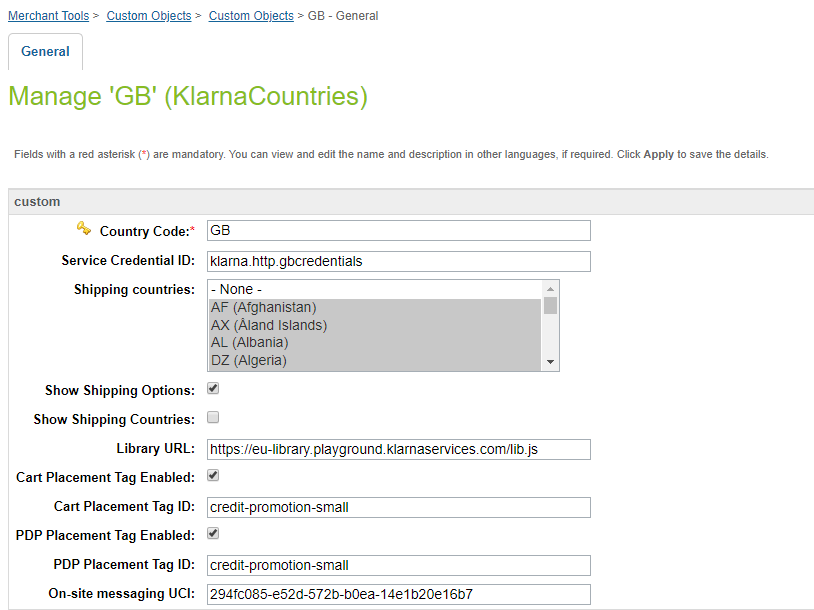
# Implementation Guide

## Setup of Business Manager

* Import the int\_klarna\_checkout\_core 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\_checkout\_core" cartridge.
  + Click Finish.
  + Click OK when prompted to link the cartridge to the sandbox.
* Import the int\_klarna\_checkout 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\_checkout" cartridge.
  + Click Finish.
  + Click OK when prompted to link the cartridge to the sandbox.
* Append 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.
  + Append ": int\_klarna\_checkout\_core:int\_klarna\_checkout" to the "Cartridges" field.
  + Click Apply
* Import metadata
  + 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’ll 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”

## 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).
  + Select the desired Shipping Countries for the current Klarna country.
  + Whether the selected Shipping Countries to be displayed in the Klarna Checkout.
  + Whether shipping options should be rendered within the Klarna Checkout.
  + Choose if cart placement tag is enabled for this market.
  + Input cart Klarna OSM tag id.
  + Choose if product detail page placement tag is enabled for this market.
  + Input product detail page Klarna OSM tag id.
  + Input full URL to the On-Site Messaging JavaScript Library.
  + Input On-Site Messaging Client Identifier.
  + Leave the other settings as is.
  + Repeat for the other countries.



* Configure Klarna Checkout 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 -> KlarnaCheckout
  + Fill out the settings as desired. Descriptions of the site preferences are in the [Site Preferences](#_Klarna_Checkout_Site) section.
* Configure Klarna Checkout 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 Klarna supported [countries custom objects](#_Configuration). Click on the one you want to edit.
  + Enter the Merchant ID (EID) and Shared Secret you received from Klarna.
  + Edit URL field if Production environment. Klarna API URLs information - [https://developers.klarna.com/api/#api-urls](https://developers.klarna.com/api/" \l "api-urls).



* How to disable Klarna Checkout.

Removing int\_klarna\_checkout from the effective cartridge path, will stop the redirects to the Karna Checkout entry point and the loading of any additional Klarna assets.

* + Log into the SCC Business Manager.
  + Click Administration -> Sites -> Manage Sites
  + Select the desired site
  + Click on the Settings tab.
  + Remove ":int\_klarna\_checkout" to the "Cartridges" field.
  + Click Apply

## Custom Code

Integration may vary based on the customer storefront. Site Genesis version 2 is used to demonstrate Klarna Checkout integration.

### Template modification

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

**default/components/footer/footer\_UI.isml**

Add Code:

<isif condition="${!empty(dw.web.Resource.msg('klarna.checkout.enabled', 'klarna', ''))}">

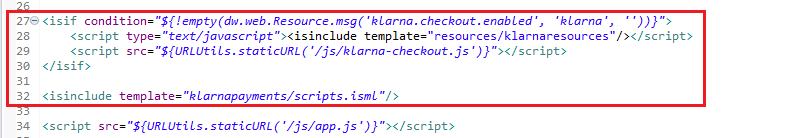
<script type="text/javascript"><isinclude template="resources/klarnaresources"/></script>

<script src="${URLUtils.staticURL('/js/klarna-checkout.js')}"></script>

</isif>

<isinclude template="kosm/scripts.isml"/>

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



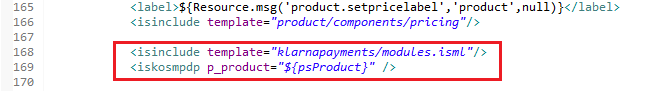
**default/product/producttopcontentPS.isml**

Add the following code:

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

<iskosmpdp p\_product="${psProduct}" />

right under the pricing template include as shown below:



**default/product/productcontent.isml**

Add the following code:

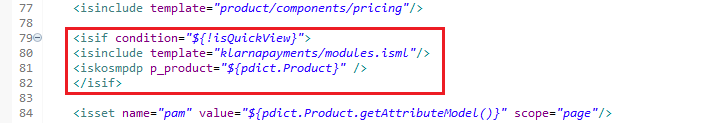
<isif condition="${!isQuickView}">

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

<iskosmpdp p\_product="${pdict.Product}" />

</isif>

right under the pricing template include as shown below:



**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

A screenshot of a cell phone

Description automatically generated

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

Add the following code:

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

<iskosmcart p\_lineitemctnr="${pdict.Basket}" />

right after the <isordertotals>:

A screenshot of a social media post

Description automatically generated

### Controller modifications

To use Klarna Checkout cartridge you will need to add the code bellow, to the following SiteGenesis controllers.

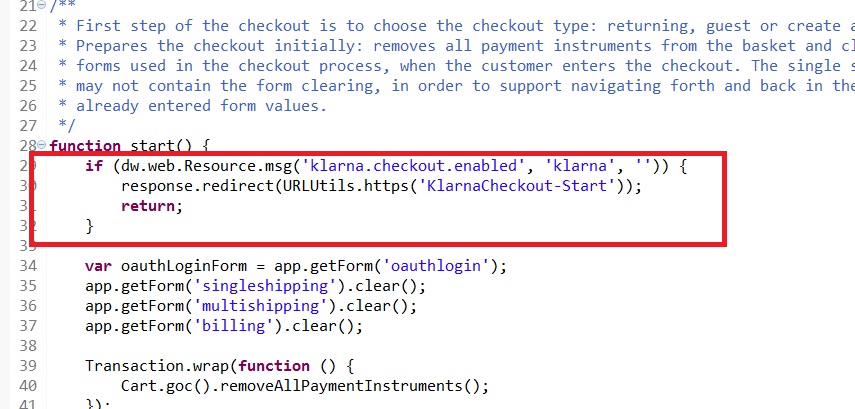
if (dw.web.Resource.msg('klarna.checkout.enabled', 'klarna', '')) {

response.redirect(URLUtils.https('KlarnaCheckout-Start'));

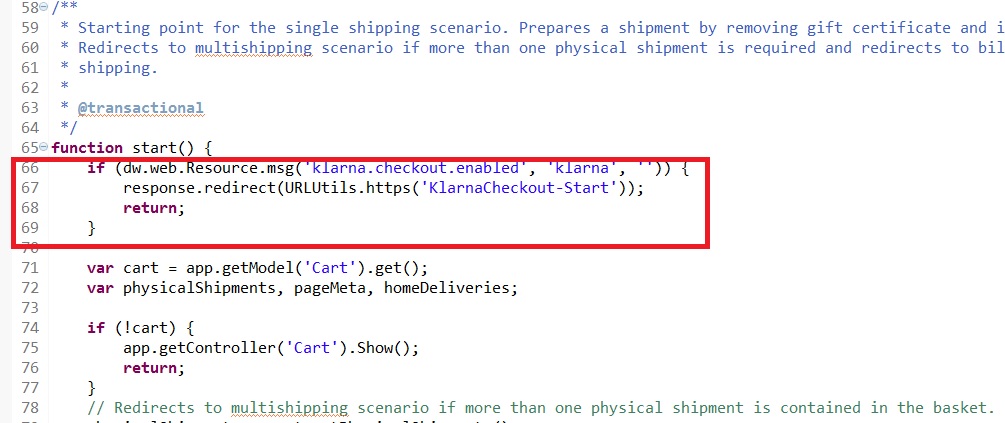
return;

}

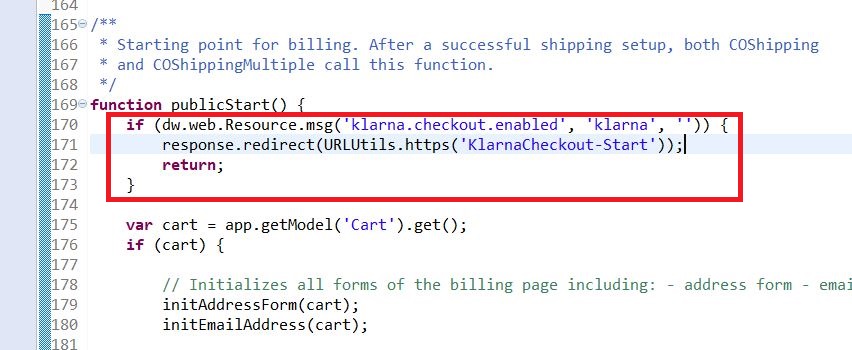
**COCustomer.js**



**COShipping.js**



**COBilling.js**



### Klarna Checkout Locale

This cartridge uses request locale to determine which version of Klarna Checkout to display. This happen through the core cartridge **countries.json** file and KlarnaCountries custom object. If the set up locales do not match any of the defined in the countries.json file, add you your country object in countries.json in the format as the already defined countries there or modify **scripts/util/KlarnaHelper.js,** “getLocaleObject” method so it pulls the correct Klarna market.

## VCN Decryption

In order 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](https://developers.klarna.com/documentation/merchant-card-service/decryption/).

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];

## 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, can be found on their developer site - <https://developers.klarna.com/documentation/klarna-checkout/>

# Testing

You can sign up for Klarna test account and found all needed test triggers and sample data here - https://developers.klarna.com/documentation/testing-environment. Please see “Test Cases.pdf” in the documentation folder for the available test cases.

# Operations, Maintenance

## Data Storage

The only Custom Objects stored in SCC are the ones created during the initial import. These are described in the [Configuration section](#_Configuration).

## 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/>

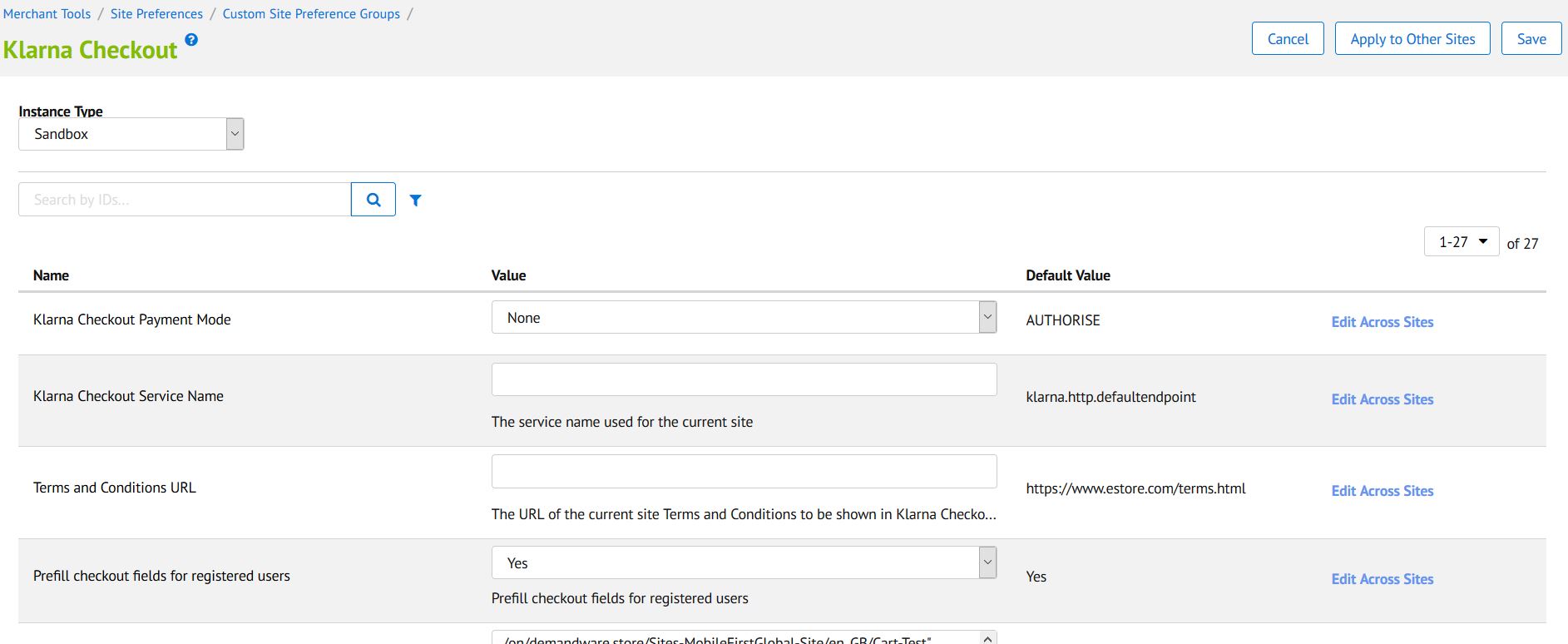
## Support

Choose your market at [https://www.klarna.com](https://www.klarna.com/) and then go to Business > Merchant Support

# User Guide

## Business Manager

### Klarna Checkout Site Preferences



**Klarna Checkout Payment Mode** – Option to turn on/off Auto capture, default is AUTHORISE, meaning the order will be AUTHORISED, but not captured.

**Klarna Checkout Service Name** – The service name used for the current site. Default is klarna.http.defaultendpoint.

**Terms and Conditions URL** – Klarna requires a URL to the storefront Terms and Conditions when rendering the checkout page. Enter the URL of current site Terms and Conditions. The URL will be sent to Klarna with the initial API request.

**Cancellation Terms URL** – URL of merchant cancellation terms. Available for DE & AT markets.

**Prefill checkout fields for registered users** – Whether to prefill checkout fields for registered users. Default is Yes.

**Allow separate shipping and billing addresses** – If Yes, the consumer can enter different billing and shipping addresses in Klarna Checkout. Default is No.

**Require validate callback success** – If Yes, validate callback must get a positive response to not stop purchase. Default is No.

**Button color preference** – CSS hex color to be used in Klarna. Default is #84bd00.

**Button text color preference** – CSS hex color to be used in Klarna. Default is #ffffff.

**Checkbox text color preference** – CSS hex color to be used in Klarna Checkout iFrame. Default is #84bd00.

**Checkbox checkmark color preference** – CSS hex color to be used in Klarna. Default is #ffffff.

**Header color preference** – CSS hex color to be used in Klarna. Default is #ffffff.

**Radius border** – Border radius to be used in Klarna Checkout iFrame. Default is 2.

**Link color preference** – CSS hex color to be used in Klarna. Default is #84bd00.

**External Payment Methods** – List of external payment methods.

**External Checkouts** – List of external checkouts.

**Date of birth mandatory** – If Yes, the consumer cannot skip date of birth. Not available for country US. Default is No.

**Additional checkbox** – Additional merchant defined checkbox. e.g. for Newsletter opt-in.

**Title mandatory** – If specified to No, title becomes optional. Only available for orders for country GB. Default is No.

**Shipping details message** – A message that will be presented on the confirmation page under the headline "Delivery".

**Show Subtotal Detail** – If Yes, the Order detail subtotals view is expanded. Default is No.

**Send Product and Image URLs** – If Yes, Product URL and Product image URL will be send that can be later embedded in communications between Klarna and the customer. Default is Yes.

**Disable Autofocus** – If Yes, Email input will not be focused when you enter the checkout. Default is No.

**Minimal Confirmation** – If Yes, displays minimal confirmation snippet. Default is No.

**Attachments** - Flag to swicth on/off the using of attachments when creating a session.

**Virtual Card Network Enabled** - If set to true SFCC will create Virtual Card Network settlement from every Klarna order. Default is false.

**VCN Private Key** - Your 4096 bit RSA Private Key.

**VCN Public Key** - Your 4096 bit RSA Public Key.

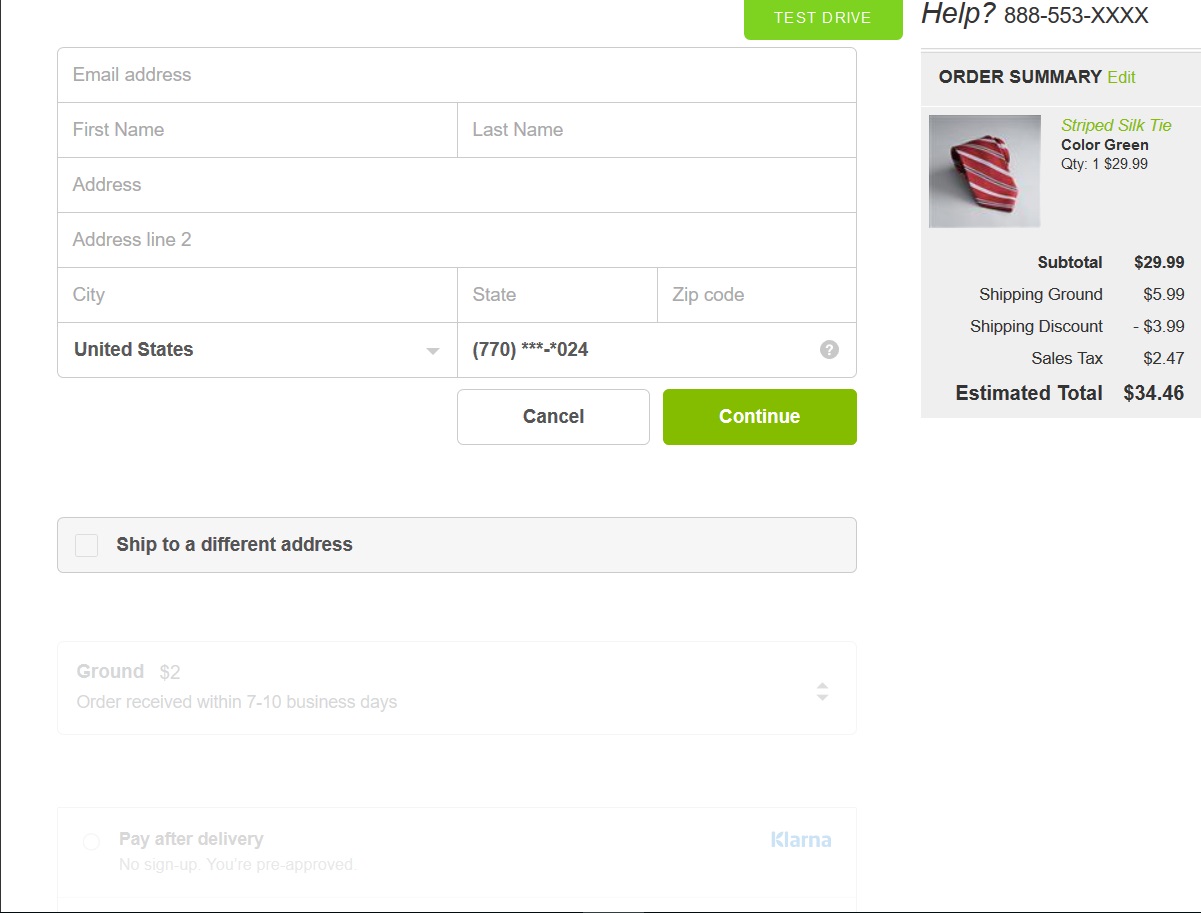
**VCN Key Id** - Unique identifier for the public key used for encryption of the card data.

Additionally, in KlarnaCountries Custom Objects, you can configure display of shipping countries and shipping options based on current Klarna country as shown in the [Configuration](#_Configuration) section.

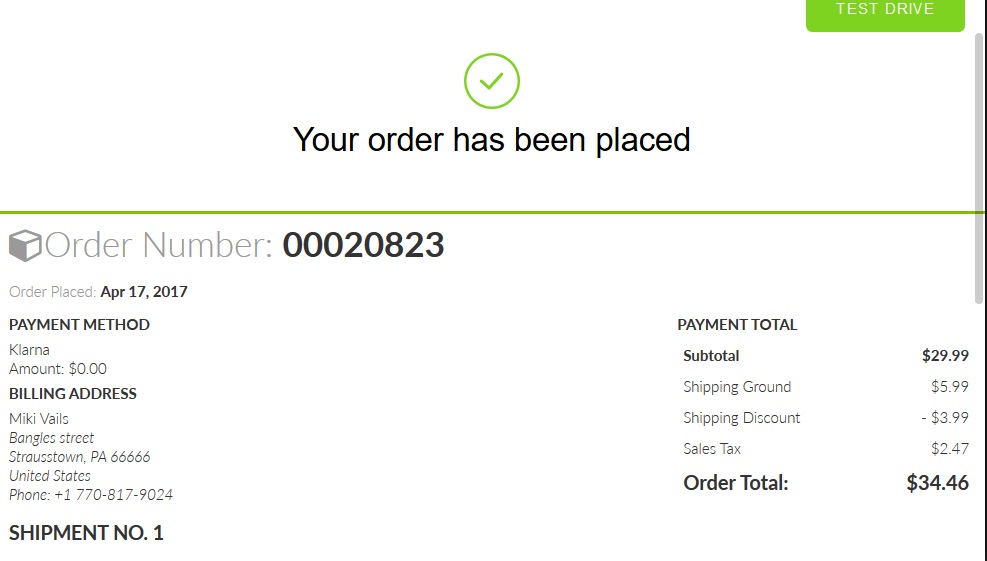
## Storefront Functionality

When a customer opts to use Klarna Checkout, they are directed to the single-page checkout shown below. Order summary at the right and an iframe with Klarna checkout at the left. If **showShippingOptions** in the KlarnaCountries CO is enabled, the applicable shipping options section will be shown above the KCO iFrame.

If a shipping method or shipping address change is made, the order summary and Klarna Checkout iframe will automatically update to reflect the changes.



Once the customer completes the checkout form and submits the order, Klarna will perform a POST request to the provided validate URL. This checkout function will validate the information provided by the consumer in the Klarna Checkout iframe before the order is created in the SCC system. Then the customer will be redirected to the order confirmation page shown below.



If the validation callback success is mandatory (which is recommended), SCC order will be created here. After the confirmation snippet html is got successfully from Klarna, the order is ‘placed’ in SCC as well and the standard confirmation template is rendered along with the Klarna confirmation snippet.

Klarna will send a push notification to the SCC storefront to confirm that the order has been created and placed. This push notification will be sent every two hours for a total of 48 hours or until it has been confirmed that the order has been received. A final attempt for order creation and placing in SCC can be made here (confirmation page redirect may fail for many reasons). If the SCC order is in the correct status, the Klarna will be ‘**acknowledged’.**

Payment results are sent asynchronously to a 'notification URL' at the Merchant (SCC) site. Payments can change status over the time (Accepted, Rejected and Pending) on Klarna Checkout side, and these statuses are updated automatically on the Commerce Cloud side through sending these notifications.

In case order is ACCEPTED order creation in SCC proceeds as usual, in case of REJECTED status, order is failed and in case of PENDING status the 'two phase ordering' approach is used, where the order is first created but only 'placed' after notification for ACCEPTED status is asynchronously received. If neither ACCEPTED, nor REJECTED status is received the order will stay in status ‘created’ until an action is taken by the merchant.

# Known Issues

Due to the way the Klarna API is creating the order in SFCC (in a new session, different than the customer one), All customer group based promotions that require the customer to be authenticated in the system, will not be applied in the traditional way. Instead, a custom price adjustment will be applied for the same amount as the initially applied promotion together will all other applicable promotions for the given product or order.

# Release History

|  |  |  |
| --- | --- | --- |
| 17.1.0 |  | Initial release |
| 17.1.1 |  | Documentation update |
| 18.1.0 |  | SFRA release |
| 20.1.0 |  | Klarna OSM. VCN update. |
| 20.1.1 |  | Updated VCN to store encrypted card details  Minor fixes around the CSS/JS location |
| 22.1.0 |  | Cartridge re-certification |