Version 20.1.0

int_sezzle_sg

Table of Contents

1. Summary

This document describes how to implement Sezzle cartridge in SiteGenesis site. This cartridge can be configured in the Business Manager and contains all elements necessary to perform a successful best practices implementation of Sezzle.

2. Component Overview

2.1. Functional Overview

Sezzle is an alternative payment platform that increases sales and basket sizes by enabling interest-free installment plans at online stores. Consumers pay over time, but our merchant partners are paid upfront, eliminating risk of fraud or non-payment.

When you pay with Sezzle, your purchase is split into four interest-free installments automatically scheduled over the next six weeks. It's a financially responsible way to pay over time and build credit.

1.1. Use Cases

Customers can use Sezzle payment method to pay their purchases.

1.2. Limitations, Constraints

The Sezzle cartridge does not have support for partial refunds. These can be placed in the Sezzle Merchant Dashboard. Sezzle's product does not allow for greater than the amount of the original purchase.

1.3. Compatibility

Cartridge is designed and developed for:

- Salesforce platform version 17.1 to 20.8
- SiteGenesis version 105.1.0
- Compatibility mode 19.10
- Locales - en_US

1.4. Privacy

Sezzle's privacy agreement can be found on our legal website at https://legal.sezzle.com/privacy

2. Implementation Guide

2.1. Setup

Section describes steps that should be completed before cartridge configuration in Business Manager.

2.1.1. Deploying cartridge to a sandbox

1. Download cartridge source code.
2. Import Sezzle cartridge to a workspace in Salesforce UX Studio.
3. Add Sezzle cartridge to Project Reference of Server Connection.
4. Wait until Studio completes workspace built and uploading source codes to a sandbox.

1.1.2. Sandbox setup

1. Go to Business Manager -> Site -> Manage Sites. Select correct site, then select Settings tab. In cartridge path at the end write the following:

   :int_sezzle:int_sezzle_sg

   At the starting, add **sezzle_sg_changes: -** This will add the overriding Sitegenesis files.

1. Download cartridge source code.
2. Upload and import metadata from the **site_import** folder. To do so, go to **Business Manager > Administration > Site Development > Site Import & Export**. Then apply the standard procedure for uploading metadata into the Commerce Cloud site. You can compress the **site_import,** after renaming the site name inside it to your site, to a .zip archive or upload it via XML files and import it separately.

1.2. Configuration

This section describes configuration of the sandbox.

1.1.1. Site Preferences configuration

1. Go to Merchant Tools > Site Preferences > Custom Site Preferences > Sezzle
2. Enable attribute - Sezzle Online Status. This attribute defines status (enable/disable) of Sezzle integration.

1. Select a value from dropdown Sezzle Mode. This attribute defines in which mode cartridge will work. Allowable values are “Sandbox” and “Production”.

1. Add site preference attribute - Sezzle Public Key with provided public key from Sezzle.
2. Add site preference attribute - Sezzle Private Key with provided private key from Sezzle.

1.1.2. Sezzle Job Configuration

1. Verify that each imported job was created.

2. Change execution scope of every job to your site.

1.3. Custom Code (Controllers)

This section describes changes that should be made to a merchant storefront cartridge.

1. Find and open template – storefront/default/checkout/billing/paymentmethods.isml. Find the following code:

```
18            <div class="form-row label-inline">
19                <isset name="radioID" value="${paymentMethodType.value}" scope="page"/>
20                <div class="field-wrapper">
21                    <input id="is-${radioID}" type="radio" class="input-radio" name="${pdict
22                </div>
23                <label for="is-${radioID}"><isprint value="${Resource.msg(paymentMethodType.
24            </div>
```

Replace it with the next code.

```
<isif condition="${paymentMethodType.value.equals('Sezzle')}">
        <isinclude template="sezzle/paymentMethodInput" />
<iselse/>
        <div class="form-row label-inline">
                <isset name="radioID" value="${paymentMethodType.value}"
  scope="page"/>
        <div class="field-wrapper">
        <input id="is-${radioID}" type="radio" class="input-radio"
  name="${pdict.CurrentForms.billing.paymentMethods.selectedPaymentMethodID
  .htmlName}" value="${paymentMethodType.htmlValue}" <isif
  condition="${paymentMethodType.value ==
  pdict.CurrentForms.billing.paymentMethods.selectedPaymentMethodID.htmlVal
  ue}">checked="checked"</isif> />
        </div>
        <label for="is-${radioID}"><isprint
  value="${Resource.msg(paymentMethodType.label,'forms',null)}"/></label>
        </div>
</isif>
```

Find the following code:

```
<iscomment>
    Custom processor
    ------------------------------------------------------------------
</iscomment>
```

Paste code after:

```
<isinclude template="sezzle/sezzlePaymentMethod" />
```

2. Find and open controller:

storefront/cartridge/controllers/COBilling.js. Find the following code (lines 139-145):

```
countryCode = Countries.getCurrent({
    CurrentRequest: {
        locale: request.locale
    }
}).countryCode;

applicablePaymentMethods = PaymentMgr.getApplicablePaymentMethods(customer, countryCode, paymentAmount.value);
```

Paste code after:

```
applicablePaymentMethods = require('*/cartridge/controllers/Sezzle').Init(cart,
        applicablePaymentMethods);
```

Find the following code (lines 174-179):

```
// Initializes all forms of the billing page including: - address form - email address - coupon form
initAddressForm(cart);
initEmailAddress(cart);

var creditCardList = initCreditCardList(cart);
var applicablePaymentMethods = creditCardList.ApplicablePaymentMethods;
```

Replace the last line with the following code:

var applicablePaymentMethods = require('*/cartridge/controllers/Sezzle').Init(cart, creditCardList.ApplicablePaymentMethods);

Find the following code (lines 351-372):

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

Replace the above code with the following code:

```
var status = Transaction.wrap(function () {
  if (app.getForm('billing').object.paymentMethods.selectedPaymentMethodID.value.equals('PayPal')) {
      app.getForm('billing').object.paymentMethods.creditCard.clearFormElement();

app.getForm('billing').object.paymentMethods.bml.clearFormElement();
      cart.removePaymentInstruments(cart.getPaymentInstruments(PaymentInstrument.METHOD_CREDIT_CARD));

cart.removePaymentInstruments(cart.getPaymentInstruments(PaymentInstrument.METHOD_BML));

cart.removePaymentInstruments(cart.getPaymentInstruments('Sezzle'));
} else if (app.getForm('billing').object.paymentMethods.selectedPaymentMethodID.value.equals(PaymentInstrument.METHOD_CREDIT_CARD)) {

app.getForm('billing').object.paymentMethods.bml.clearFormElement();
```

```
cart.removePaymentInstruments(cart.getPaymentInstruments(PaymentInstrument.METHOD_BML))
;

cart.removePaymentInstruments(cart.getPaymentInstruments('PayPal'));

cart.removePaymentInstruments(cart.getPaymentInstruments('Sezzle'));
}
else if
                (app.getForm('billing').object.paymentMethods.selectedPaymentMethodID.value.eq
                uals(PaymentInstrument.METHOD_BML)) {

app.getForm('billing').object.paymentMethods.creditCard.clearFormElement();
                        if (!app.getForm('billing').object.paymentMethods.bml.ssn.valid) {
                                        return false;
                                }

cart.removePaymentInstruments(cart.getPaymentInstruments(PaymentInstrument.METHOD_CREDI
T_CARD));
                         cart.removePaymentInstruments(cart.getPaymentInstruments('PayPal'));
                         cart.removePaymentInstruments(cart.getPaymentInstruments('Sezzle'));
                 }
                else if
        (app.getForm('billing').object.paymentMethods.selectedPaymentMethodID.value.equals('Se
        zzle')) {

cart.removePaymentInstruments(cart.getPaymentInstruments(PaymentInstrument.METHOD_CREDI
T_CARD));

cart.removePaymentInstruments(cart.getPaymentInstruments(PaymentInstrument.METHOD_BML))
;
                cart.removePaymentInstruments(cart.getPaymentInstruments('PayPal'));
                }
                return true;
                });

                return status;
```

3. Find and open controller – storefront/cartridge/controllers/COSummary.js. Find "submit" function. Add the below code at the beginning of the function:

```
var redirected = require('*/cartridge/controllers/Sezzle').Redirect();
if (redirected) {
        return;
}
```

4. Find and open property file - int_sezzle/cartridge/templates/resources/sezzle.properties. Find property “sezzle.controllers.cartridge”. Set it as name of your controllers cartridge (with script files app.js, guard.js), for example:

```
22 sezzle.controllers.cartridge=app_storefront_controllers
```

1.4. External Interfaces

Commutation between Sezzle service and Salesforce Commerce Cloud Platform is based on Sezzle REST API. All outside traffic from Salesforce Commerce Cloud instance is handled by HTTPS protocol.

In the Salesforce Commerce Cloud Platform each API call to Sezzle is wrapped into Service that is handling monitoring and logging functionality.

Sezzle integration documentation - https://docs.sezzle.com/sezzle-pay/

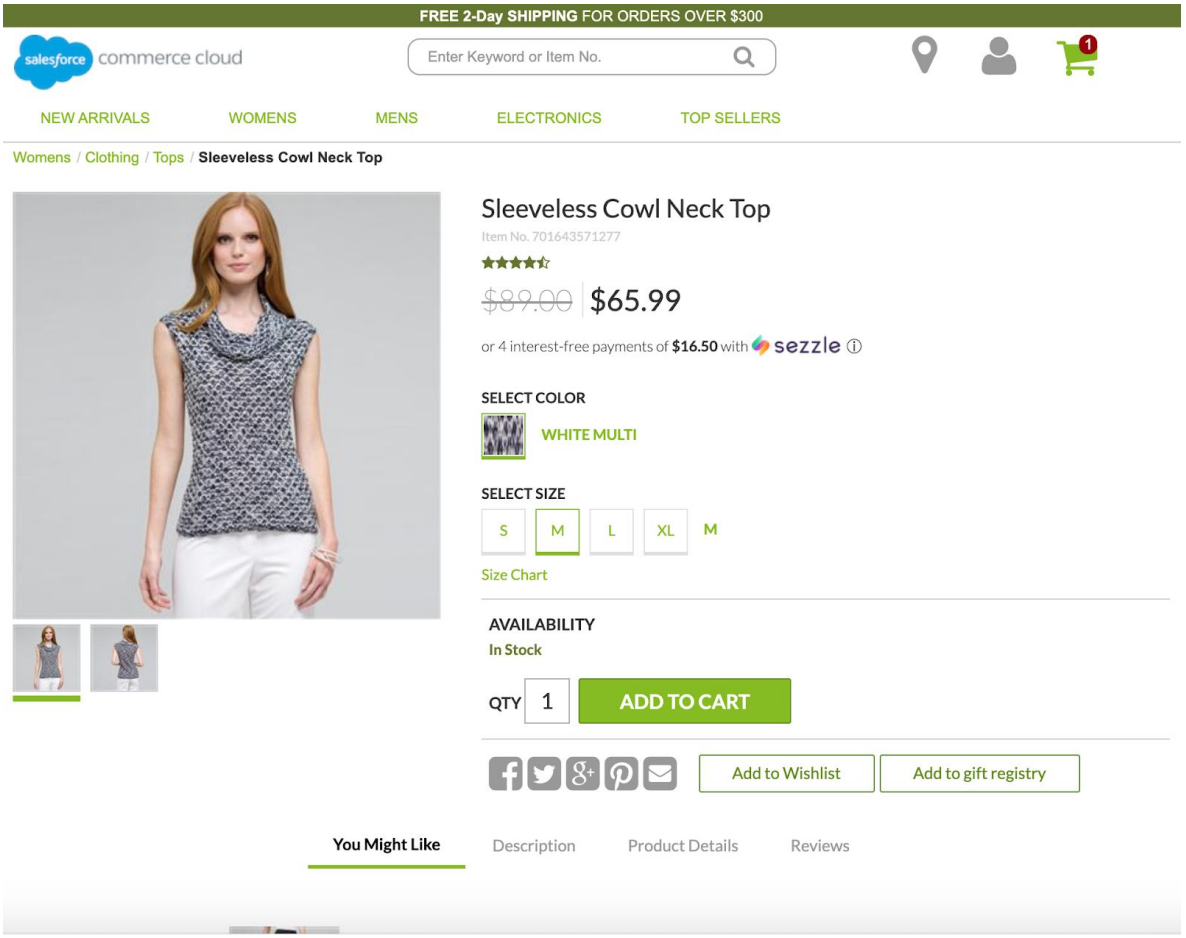
1.1.1. Sezzle Widget Integration

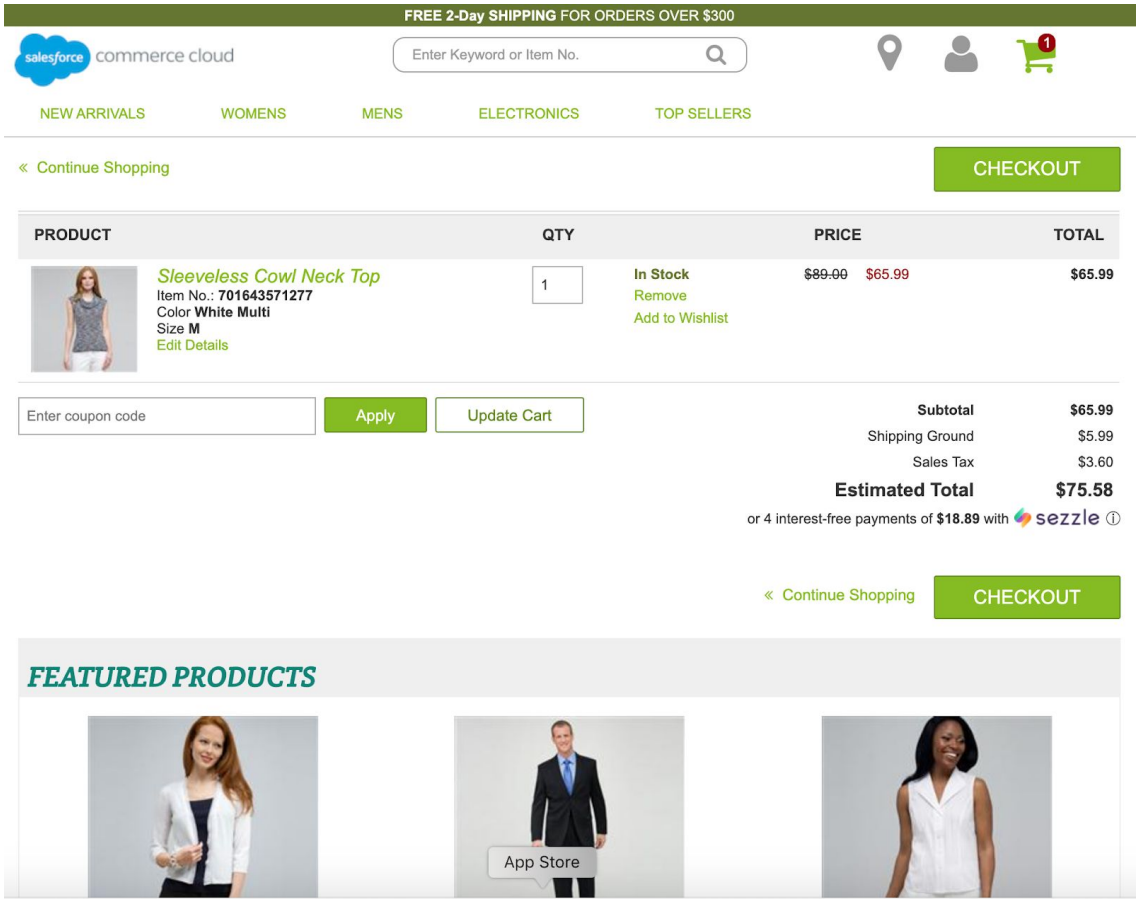
Merchants can integrate Sezzle’s widgets to display Sezzle’s service on their product and cart pages.

Link for integrating widgets - https://docs.sezzle.com/sezzle-pay/#sezzlejs

Here are some snapshots after integrating Sezzle’s widgets.

Product Page


FREE 2-Day SHIPPING FOR ORDERS OVER $300
salesforce commerce cloud
Enter Keyword or Item No.
NEW ARRIVALS
WOMENS
MENS
ELECTRONICS
TOP SELLERS
Womens / Clothing / Tops / Sleeveless Cowl Neck Top
Sleeveless Cowl Neck Top
Item No. 701643571277
$89.00
$65.99
or 4 interest-free payments of $16.50 with sezzle
SELECT COLOR
WHITE MULTI
SELECT SIZE
S
M
L
XL
M
Size Chart
AVAILABILITY
In Stock
QTY
1
ADD TO CART
Add to Wishlist
Add to gift registry
You Might Like
Description
Product Details
Reviews


FREE 2-Day SHIPPING FOR ORDERS OVER $300
salesforce commerce cloud
Enter Keyword or Item No.
NEW ARRIVALS
WOMENS
MENS
ELECTRONICS
TOP SELLERS
« Continue Shopping
CHECKOUT
PRODUCT
QTY
PRICE
TOTAL
Sleeveless Cowl Neck Top
Item No.: 701643571277
Color White Multi
Size M
Edit Details
1
In Stock
Remove
Add to Wishlist
$89.00
$65.99
$65.99
Enter coupon code
Apply
Update Cart
Subtotal
$65.99
Shipping Ground
$5.99
Sales Tax
$3.60
Estimated Total
$75.58
or 4 interest-free payments of $18.89 with sezzle
« Continue Shopping
CHECKOUT
FEATURED PRODUCTS
App Store

Cart Page

2.2. Firewall Requirements

No specific Firewall requirements.

2.3. Testing

Sezzle has a sandbox that can be used for testing. In Business Manager, navigate to the SiteGenesis Site -> Site Preferences->Custom Preferences. A custom site preference group with the ID SEZZLE_PAYMENT is available. Please select it and locate 'Sezzle Mode'. Select 'Sandbox' as the mode for testing and 'Save' it. Please use "123123", when prompted for OTPs while using Sandbox mode.

3. Operations, Maintenance

3.1. Data Storage

Sezzle cartridge is extending Salesforce Commerce Cloud system objects to store related Sezzle data for request. Following objects that were extended: Order, Product, Category, SitePreference.

3.2. Availability

The Sezzle's payment gateway guarantees an uptime of almost 100%. However, in case the system does not respond, customers will not be able to use Sezzle to checkout and will have to use a different payment method.

Customers are able to logout at all times from Sezzle's checkout method, thus enabling a flexible checkout process.

3.3. Support

In case of any availability issues, Sezzle support can be reached via email at merchantsupport@sezzle.com.

4. User Guide

4.1. Roles, Responsibilities

Integration of this cartridge will typically be done by a SFCC developer. Sezzle will provide access keys for be used with the API.

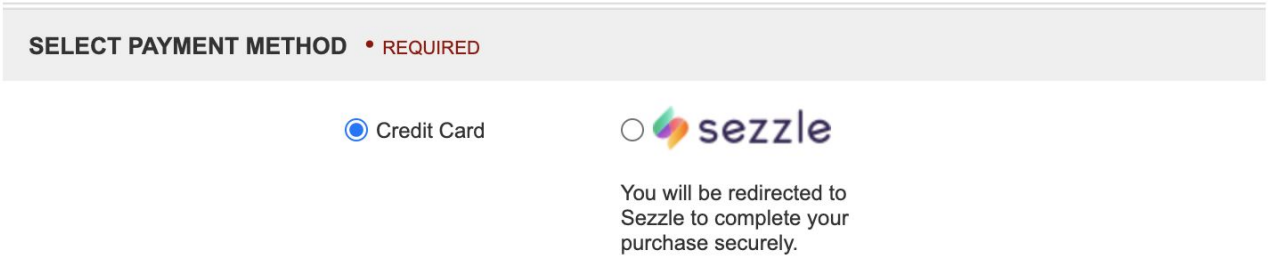
4.2. Business Manager

The cartridge doesn't extend Business Manager.

4.3. Storefront Functionality

Once enabled, the Sezzle Link integration will add a new functionality to your Salesforce Commerce Cloud store.

The Sezzle payment method will show as an option on the billing page.



5. Known Issues

No known Issues.

7. Release History

Version	Date	Changes
18.1.0	25/09/2018	Initial release
19.1.0		SFRA Support and Support for new Guidelines
20.1.0	03/08/2020	Recertification with few test cases and cartridge naming change