



Version 21.0.0



int_sezzle_sg

Table of Contents

1.	Summary	3
2.	Component Overview	3
2.1	Functional Overview	3
2.2	Use Cases	3
2.3	Limitations, Constraints	3
2.4	Compatibility	3
2.5	Privacy, Payment	3
3.	Implementation Guide	4
3.1	Setup	4
3.1.1	Deploying cartridge to a sandbox	4
3.1.2	Sandbox setup	
3.2	Configuration	4
3.2.1	Site Preferences configuration	5
3.2.2	Sezzle Job configuration	
3.3	Custom Code (Controllers)	5
3.5	External Interfaces	9
3.6	Firewall Requirements	11
3.7	Testing	11
4.	Operations, Maintenance	12
4.1	Data Storage	12
4.2	Support	12
5.	User Guide	13
5.1	Roles, Responsibilities	13
5.2	Business Manager	13
5.3	Storefront Functionality	13
6.	Known Issues	13
7.	Release History	13

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 21.2
- SiteGenesis version 105.1.0
- Compatibility mode 21.2
- 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 -> Sites -> Manage Sites -> Storefront Sites**. Select correct site, then select **Settings** tab. In cartridge path at the end write the following:
`int_sezzle:int_sezzle_sg:`

and, at the starting:
`sezzle_sg_changes:`
2. Go to **Business Manager -> Sites -> Manage Sites -> Business Manager Site**. Click on **Business Manager**, then select **Settings** tab. In cartridge path at the beginning write the following:
`bm_sezzle:int_sezzle:`
3. 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

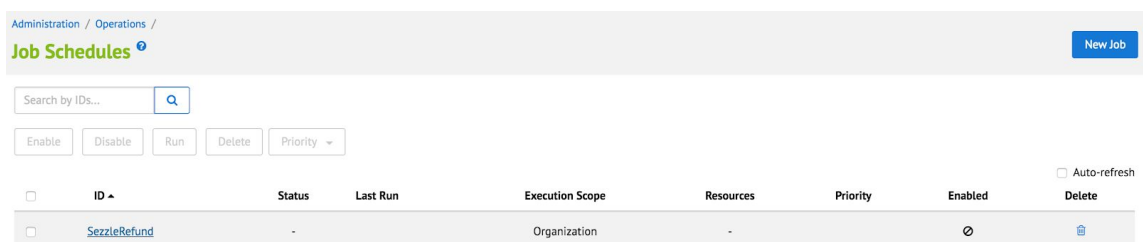
This section describes configuration of the sandbox.

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 the status (enable/disable) of Sezzle integration.
3. Select a value from dropdown **Sezzle Mode**. This attribute defines which mode cartridge will work. Allowable values are “**Sandbox**” and “**Production**”.
4. Add site preference attribute - **Merchant UUID** with provided Merchant UUID from Sezzle.
5. Add site preference attribute - **Sezzle Public Key** with provided public key from Sezzle.
6. Add site preference attribute - **Sezzle Private Key** with provided private key from Sezzle.
7. Add site preference attribute - **Payment Action**. This attribute defines whether you want to capture the payment instantly after order creation or do in a later period after the order is successfully created and payment is authorized. If you choose to go with the first way, select **AUTHORIZATION + CAPTURE** else **AUTHORIZATION**.
8. Add site preference attribute - **Tokenize Customer?**. By selecting Yes, you will allow Sezzle to tokenize the authenticated customer’s account. So, if the customer gets tokenized, the next time while checking out, he/she will not have to redirect to Sezzle for completing the order. And, payment action will be processed as per the selection you have made in the Payment Action field.
9. Add site preference attribute - **Allow Widget in PDP Page?**. By selecting Yes, you will allow Sezzle Widget to render in PDP. Please refer to Storefront functionality for detailed information on this.
10. Add site preference attribute - **Allow Widget in Cart Page?**. By selecting Yes, you will allow Sezzle Widget to render in the Cart Page. Please refer to Storefront functionality for detailed information on this.

1.1.2. Sezzle Job Configuration

1. Verify that each imported job was created.



The screenshot shows the 'Job Schedules' page in a web application. At the top, there's a breadcrumb 'Administration / Operations / Job Schedules' and a 'New Job' button. Below is a search bar 'Search by IDs...' and a row of action buttons: 'Enable', 'Disable', 'Run', 'Delete', and a 'Priority' dropdown. A table follows with columns: ID, Status, Last Run, Execution Scope, Resources, Priority, Enabled, and Delete. One job is listed: 'SezzleRefund' with status '-', last run '-', execution scope 'Organization', resources '-', priority '0', and enabled status '0'. An 'Auto-refresh' checkbox is on the right.

<input type="checkbox"/>	ID	Status	Last Run	Execution Scope	Resources	Priority	Enabled	Delete
<input type="checkbox"/>	SezzleRefund	-	-	Organization	-	0	0	

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/templates/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.equals(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_CREDIT_CARD));

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

cart.removePaymentInstruments(cart.getPaymentInstruments(PaymentInstrument.METHOD_CREDIT_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. Replace the contents with the below code:

```

var redirectStatus =
require('*/cartridge/controllers/Sezzle').Redirect();
if (!redirectStatus.isSezzle) {
  var placeOrderResult = app.getController('COPlaceOrder').Start();
  if (placeOrderResult.error) {
    start({
      PlaceOrderError: placeOrderResult.PlaceOrderError
    });
  } else if (placeOrderResult.order_created) {
    showConfirmation(placeOrderResult.Order);
  }
  return;
}

if (!redirectStatus.error) {
  return;
}

start({
  PlaceOrderError: redirectStatus.RedirectError
});

```

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:

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

5. Add the below snippet at the beginning of the file - storefront/templates/default/product/productcontent.isml for showing widget in PDP

```

<script>
    var sezzleData = require('*/cartridge/scripts/data/sezzleData');
</script>
<isif condition="{sezzleData.isSezzleEnabled() && sezzleData.getMerchantUUID() && sezzleData.isWidgetScriptAllowedInPDP()}">
    <script src="{Resource.msg('sezzle.widget.url', 'sezzle', null)}?uuid={sezzleData.getMerchantUUID()}"
    type="text/javascript"></script>
</isif>

```

6. Add the below snippet after line no 10 in the file - storefront/templates/default/checkout/cart/cart.isml for showing widget in Cart Page

```

<script>
    var sezzleData = require('*/cartridge/scripts/data/sezzleData');
</script>
<isif condition="{sezzleData.isSezzleEnabled() && sezzleData.getMerchantUUID() && sezzleData.isWidgetScriptAllowedInPDP()}">
    <script src="{Resource.msg('sezzle.widget.url', 'sezzle', null)}?uuid={sezzleData.getMerchantUUID()}"
    type="text/javascript"></script>
</isif>

```

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/>

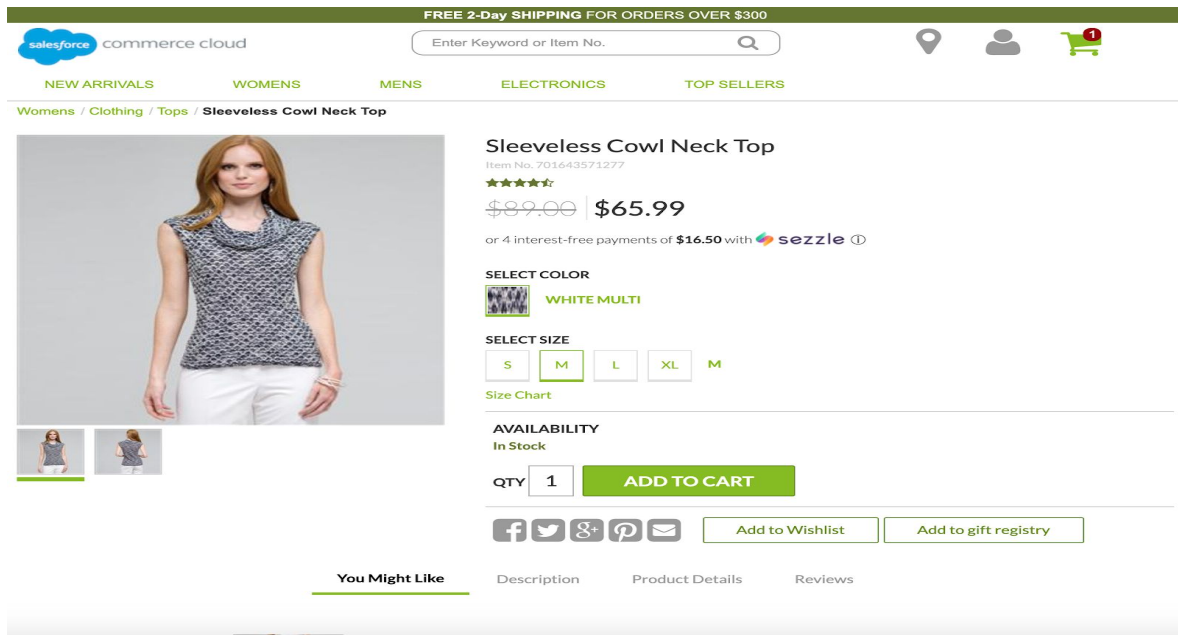
1.1.1. Sezzle Widget Integration

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

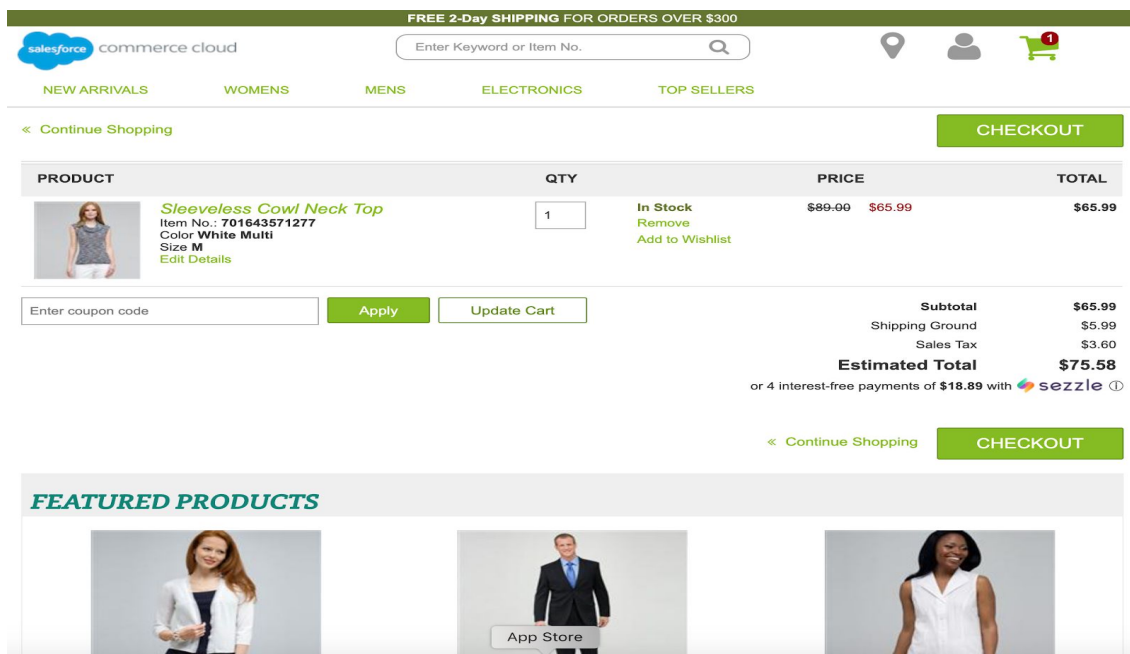
Link for integrating widgets - <https://docs.sezzle.com/sezzle-pay/#sezzlejs>

Here are some snapshots after integrating Sezzle's widgets.

Product Page



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

Sezzle Transactions module will be enabled here if you have successfully configured the Sezzle business manager modules. From this section, you will be able to perform various operations on Sezzle order. Operations include :-

1. Full Capture
2. Partial Capture
3. Full Refund
4. Partial Refund
5. Full Release
6. Partial Release

 Sandbox - sezzle01
RefArch

Merchant Tools Administration Storefront Toolkit

  (Arijit De) 

Ordering > Sezzle Transactions

Sezzle Transactions

This page allows you to search for orders which have Sezzle transactions by order number. Also there is possibility to search Sezzle transaction by reference id.

Search Order

By Order Number By Reference ID

Order Number:

Number	Order Date	Created By	Registration Status	Customer	Customer Email	Order Total	Sezzle Amount	Sezzle Auth Expiration	Action
00026901	6/01/20 12:51 pm	Customer	Unregistered	Arijit De		\$57.74	USD 57.74	6/01/20 2:21 pm	Change
00026702	5/29/20 7:10 pm	Customer	Registered	R B	rajanstillaive@gmail.com	\$57.74	USD 57.74	5/29/20 8:40 pm	Change
00026701	5/29/20 7:07 pm	Customer	Registered	R B	rajanstillaive@gmail.com	\$57.74	USD 57.74	5/29/20 8:37 pm	Change
00026601	5/29/20 6:42 pm	Customer	Registered	R B	rajanstillaive@gmail.com	\$57.74	USD 50.00	6/05/20 6:42 pm	Change
00026504	5/29/20 6:21 pm	Customer	Registered	R B	rajanstillaive@gmail.com	\$57.74	USD 50.00	6/05/20 6:21 pm	Change
00026503	5/29/20 6:17 pm	Customer	Registered	R B	rajanstillaive@gmail.com	\$325.48	USD 200.00	6/05/20 6:17 pm	Change
00026502	5/29/20 6:13 pm	Customer	Registered	R B	rajanstillaive@gmail.com	\$31.49	USD 20.00	6/05/20 6:13 pm	Change
00026501	5/29/20 6:05 pm	Customer	Registered	R B	rajanstillaive@gmail.com	\$57.74	USD 57.74	6/05/20 6:05 pm	Change
00026405	5/29/20 5:58 pm	Customer	Registered	R B	rajanstillaive@gmail.com	\$57.74	USD 50.00	5/29/20 7:28 pm	Change
00026404	5/29/20 5:56 pm	Customer	Registered	R B	rajanstillaive@gmail.com	\$57.74	USD 57.74	5/29/20 7:26 pm	Change

Showing 1 - 10 of 249 items.

Show 50 100 All items

[1](#) [2](#) [3](#) [4](#) [5](#) [6](#) ... [25](#) [Next](#) >>

<< Back to Ordering

Note :-

5. As long as the Authorization is valid, you can full/partial capture the payment as well as release amount.
6. After the Authorization is expired and by that time, if no amount is captured, the payment will be released and become invalid and you will not be able to do any more operations.
7. After the Authorization is expired and by that time, if some amount is captured, then you will be able to perform operation on that captured amount and the remaining amount will be automatically released.

7.1. 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.

SELECT PAYMENT METHOD • REQUIRED

☒ Credit Card



You will be redirected to Sezzle to complete your purchase securely.

8. 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

20.2.0	31/12/2020	<ul style="list-style-type: none"> ● Failing order and restoring cart if capture fails. ● Order marking as completed and payment as paid only if capture succeeds. ● Failing order if basket changes after completion of payment at Sezzle. ● Sezzle widget added for PDP and Cart. ● Rate Limiting and Circuit Breaker disabled.
21.0.0		<ul style="list-style-type: none"> ● Sezzle V2 API Support ● Manual Capture ● Full/Partial Capture ● Full/Partial Refund ● Full/Partial Release ● Customer Tokenization