Chargent API & Integrations

Chargent Developer API

> Development using Chargent's API requires existing customers to be on Chargent's Platform Edition or above. Full API documentation is available by request for customers who meet those requirements.

Chargent has an extensive, PCI compliant API for web site integrations. These web services are available as an extension to the standard SOAP API that Salesforce offers, so they can be called in the same way that you interact with other parts of the Salesforce API.

You can also call Chargent methods directly from inside Salesforce, without a user initiating them in the normal manner, such as from a customized Visualforce interface.

The Chargent web services API requires a Chargent Platform edition license. Please contact us for complete documentation and details.

Chargent's API allows you to programmatically call the following actions:

- Charge
- Charge Authorized
- Authorize
- Refund
- Void
- Register Token
- Generate Payment Request
- Parse payment result message and status

For complete information regarding Chargents API please check out our online Developer documentation page.

Testing Callouts

When testing the callouts that Chargent makes, you can't use the HttpCalloutMock class in Salesforce because it has to be done from within the Chargent managed package/namespace.

To create sample responses and make sure that your code behaves appropriately, here is a workaround:
Wrap the actually call to Chargent Webservices in a
if(!test.isRunnungTest()){

    code that makes call out here

}else{

  //create a transaction record manually or set whatever values need for your test to continue

}

Uncommitted Work Errors

Purpose

When building a custom visualforce page that will be using Chargent Webservice Methods to process a charge there is a potential that you may receive an error from Salesforce when executing the Chargent Webservice Method as follows:

You have uncommitted work pending. Please commit or rollback before calling out

Background
One of the framework requirements when developing on the Salesforce platform is that you cannot perform any DML prior to making a callout to any external service in the same transaction. Chargent Webservices, when executed, makes a callout to the appropriate gateway to process the charge, thus you cannot perform any DML prior to executing a method from Chargent Webservices.

Solution
Within the lifecycle of a Visualforce page, each execution of a controller method from the Visualforce page itself is a separate transaction. The key is that individual execution of the method must come from the page and not from the controller.

The following is an example of a minimalistic page and controller. It is intended only to show the typical pattern that causes the issue.

Error Example:

Page

```
<apex:page standardController="Orders" extensions="example_Controller">

  <apex:pagemessage id="msgs"/>

  <apex:form>
    <apex:commandbutton action="{!update_and_charge}" rerender="msgs"/>
  <apex:form>

<apex:page>
```

Controller

```
public class example_Controller{

  public Orders o;

  public error_example(ApexPAges.standardController con){}
  }

public void update_and_charge(){
    update o;
    charge();
  }

public void charge(){
    ChargentOrders.TChargentOperations.TChargentResult result = ChargentOrders.tChargentOperations.ChargentOrders_Click(o.id);

  }
}
```

When the command button is clicked it executes the update_and_charge() method. That method then updates the Chargent Order record and then attempts to execute the charge() method. When the charge() method executes the ChargentWebservices Orders_Click method and attempts to make the callout to the gateway you will receive the error message.
The solution is to break up the calls using the oncomplete tag on the command button in conjunction with an apex:actionFunction to call the charge() method. This solution follows:

Working Example

Page

```
<apex:page standardController="Orders" extensions="example_Controller">

  <apex:pagemessage id="msgs"/>

  <apex:form>

    <apex:actionFunction name="charge_it" action="{charge}" rerender="msgs"/>
    <apex:commandbutton action="{!update_and_charge}" onComplete="charge_it();" rerender="msgs"/>

  </apex:form>

</apex:page>
```

Controller

```
public class example_Controller{

  public Orders o;

  public error_example(ApexPAges.standardController con){
    o=con.getRecord();
  }

  public void update_and_charge(){
    update o;
  }

  public void charge(){
    ChargentOrders.TCharegntOperations.TChargentResult result = ChargentOrders.tChargentOperations.ChargentOrders_Click(o.id);

  }
}
```

Notice that on the page we added an apex:actionFunction to call the charge() method in the controller on completion of the update_and_charge() method. Also, we removed the call to the charge() method from the controller that was on line 11 in the error example.

In doing this, on click of the command button the update_and_charge() method is executed which updates the Chargent Order and then the transaction finishes and returns control to the page. The page then calls the actionFunction via the oncomplete tag and then executes the charge() method in the controller in a separate transaction without producing any errors.

Payment Gateway Integrations

Chargent includes ready-to-go, direct integrations to more than 30 payment gateway integrations. Choose from Authorize.net, CyberSource, Stripe, PayTrace, Paypal Payflow Pro, NMI, MerchantE, and more. You can even connect to multiple payment gateways simultaneously with Chargent, to support different business units or currencies.

For more information on Chargent's Payment Gateway integrations, please see our Gateways page.

Chargent for Salesforce Billing

[embed video: How To Take Payments In Salesforce CPQ With Chargent (DEMO), https://www.youtube.com/watch?v=uBs3MUaaH8Q]

Step 1: Package Installation

Prerequisites

Packages

- Salesforce CPQ 222.2, or later must be installed first.
- Salesforce Billing, Version Summer '19 220.7 or later must be installed first.
- Consumption Schedule must be enabled. Contact Salesforce support for enablement if needed.

Install Links

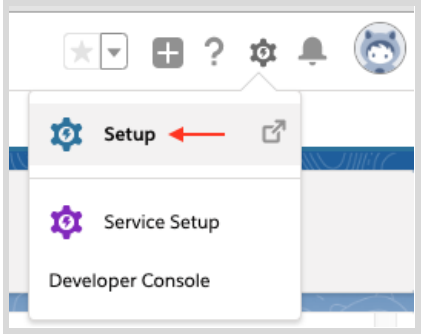
Chargent Gateways Connector for Salesforce Billing

Step 2: Salesforce Organization Setup

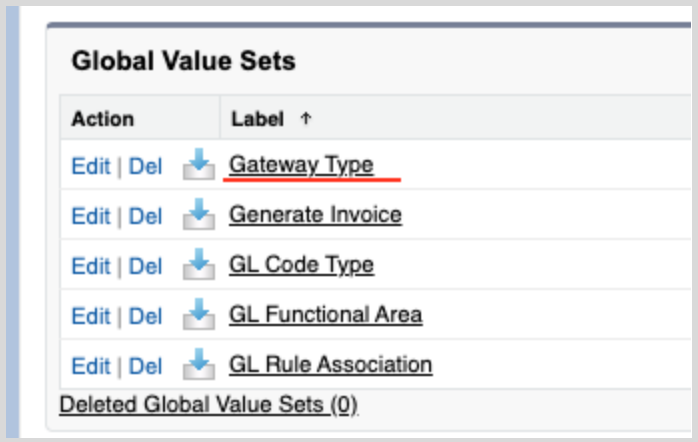
Add Gateway Type

In order to add Chargent gateways to your Salesforce Billing setup, you will first need to create a new Gateway Type in the Global Value set. To do this simply:

1. Navigate to Salesforce Setup



2. In the search box type “Picklist Value Sets”
3. Click on “Picklist Value Sets”
4. Find the “Gateway Type” value set
5. Click On the “Gateway Type” label

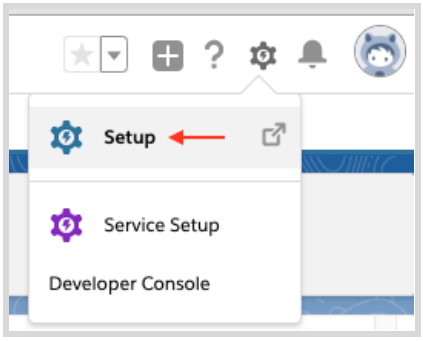


6. Click “New” under values
7. Type “Chargent Gateway” in the text box - exclude the quotes
8. Check the box that reads “Add the new picklist values to all Record Types that use this Global Value Set.”
9. Click the [**Save**] button

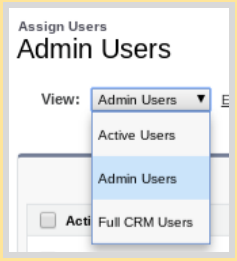
Permission Set Assignment

Assign the “Chargent Gateway Admin” permission set to users who will set up and / or edit the Gateway records. Typically, it is advised to only provide this permission to Admin users.

1. Navigate to Salesforce Setup



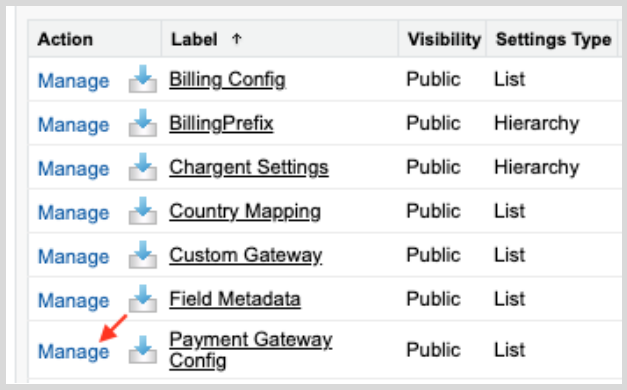
2. In the search box type “Permission Sets”
3. Click on “Permission Sets”
4. Click on “Chargent Gateway Admin”
5. Click the [Manage Assignments] button near the top of the screen
6. Click the [Add Assignments] button

Note: If you wish to limit the users list to admins only, click on “Admin Users” from the “View” picklist.	

7. Check the box next to users you wish to grant access, then click the [Assign] button.

Add Chargent Gateway to Custom Setting

1. Navigate to Salesforce Setup
2. In the search box type “Custom Settings”
3. Click “Custom Settings”
4. Next to **Payment Gateway Config** click “Manage”



5. Click the [New] button
6. Complete the Fields:
    a. Name
        i. Chargent Gateway
    b. Gateway Class Name
        i. ChrgntBllng.ChargentGatewayAPI

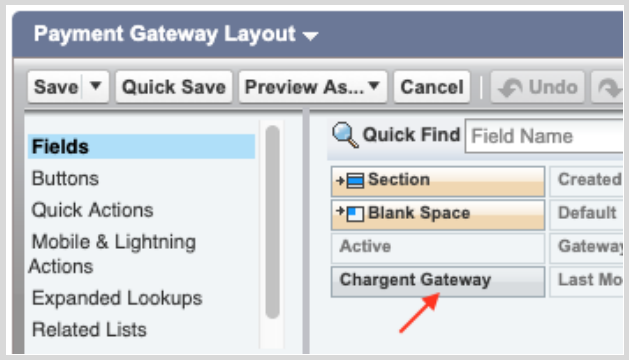
7. Click the [Save] button

Update the Payment Gateway Page Layout

NOTE: Completing this step can be done in two ways; Adding the “Chargent Gateway” field to the default “Payment Gateway” page layout, or assigning the “Chargent Gateway Layout” to the needed profiles. **We recommend the first option**, adding the “Chargent Gateway” field to the default “Payment Gateway” page layout, as it’ll ensure that your “Payment Gateway” layout remains as up-to-date as possible in cases where Salesforce.com updates this Object.

To add the “Chargent Gateway” field to the Payment Gateway Layout, follow these steps:

1. From the Salesforce Setup page, click the “Object Manager” tab
2. Click on “Page Layouts” in the left-side menu
3. Click “Payment Gateway Layout”
4. From the “Fields” section of the palette, drag the “Chargent Gateway” field (ChrgntBllng__Chargent_Gateway__c) onto the Payment Gateway layout

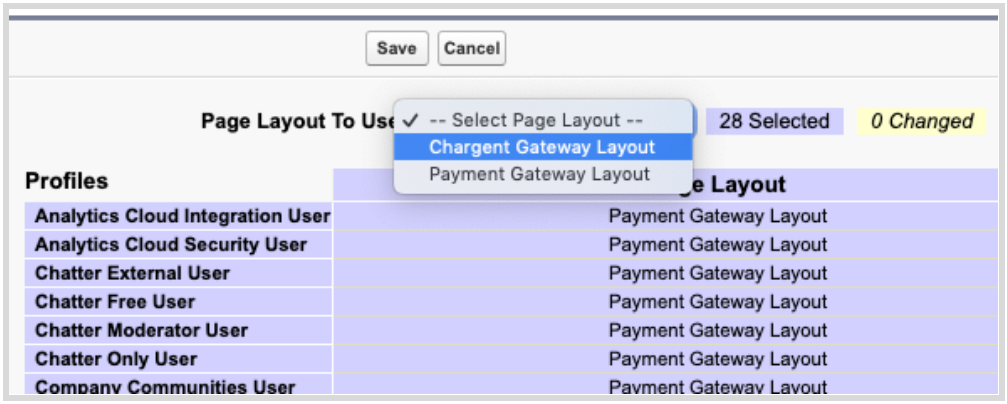


5. Click the [Save] button

If you wish to instead assign the Chargent Gateway Layout to the appropriate profiles, follow these steps:

1. From the Salesforce Setup page, click on the “Object Manager” tab
2. Click on “Payment Gateway”
3. Click on “Page Layouts” in the left-side menu
4. Click the [Page Layout Assignment] button
5. Click the [Edit Assignments] button

6. Select the appropriate profiles (those you wish to use for this feature). Hold the CTRL button while clicking to choose multiple profiles



7. From the “Page Layout to Use” picklist, choose “Chargent Gateway Layout”
8. Click the [Save] button

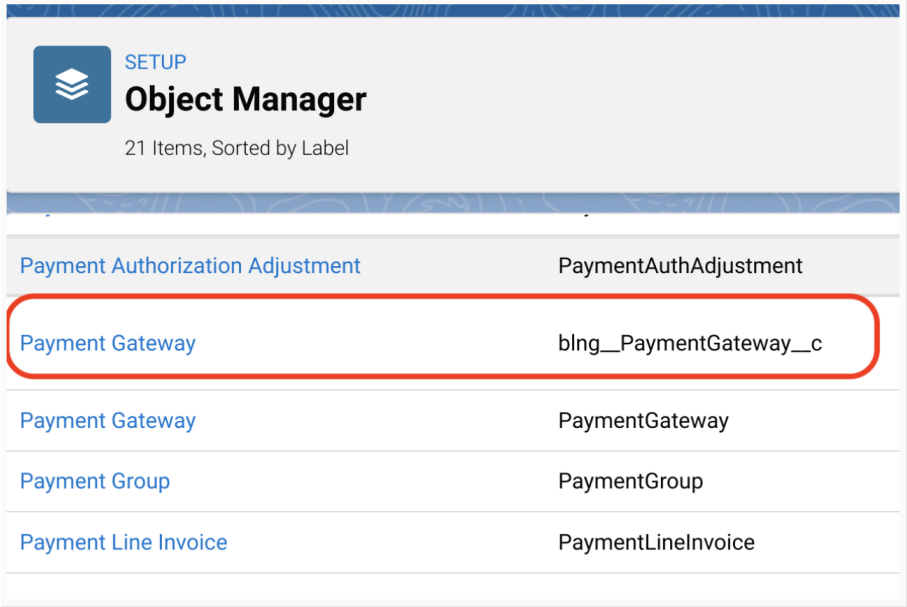
Step 3: Setup Your Chargent Gateway Record

1. Click the App Launcher icon at the top-left of the screen
2. Search for and click on “Chargent Settings”
3. Click the “Chargent Setup Wizard” tab
4. Choose your payment gateway from the picklist
5. Enter your payment gateway credentials, then click the [Sign In] button
6. Choose whether to enable Tokenization. (We highly recommend enabling it in order to protect yours and your customers’ data.)
7. Select the Currency Types you plan to accept, then click the [Continue] button
8. Select the Payment Methods you plan to accept, then click the [Continue] button

Step 4: Salesforce Billing Payment Gateway Setup

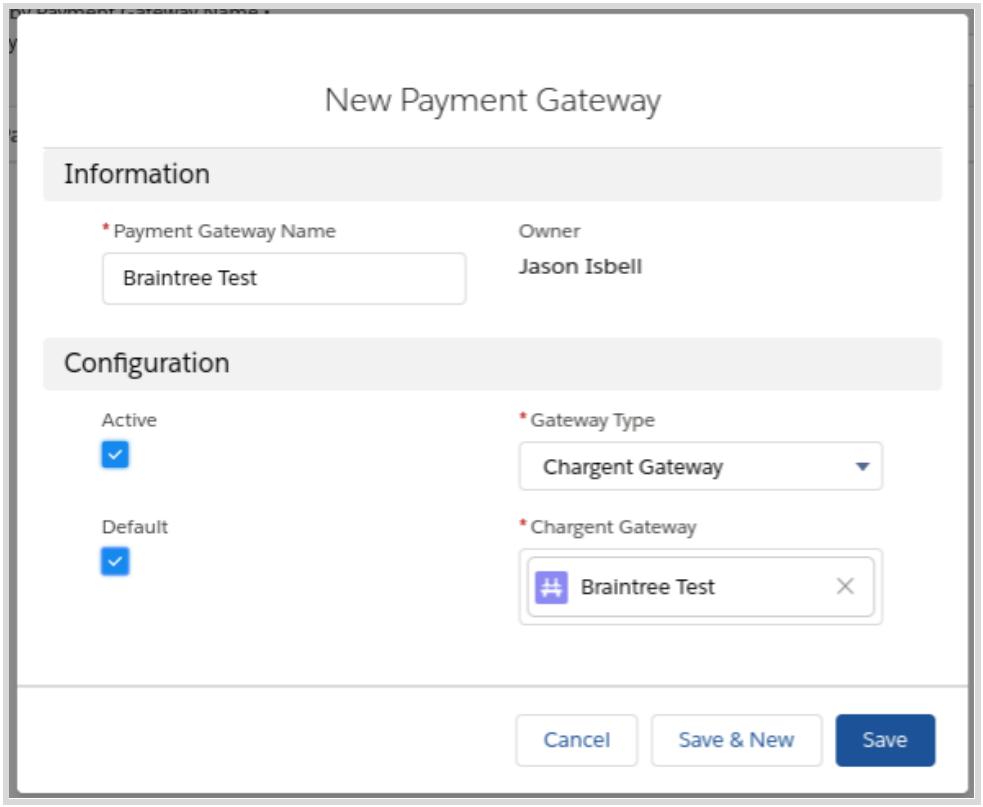
1. Click the App Launcher icon at the top-left of the screen
2. Search for “Payment Gateways,” then click on “Payment Gateways”

Note: There will be two different Payment Gateway objects. You want to select the one that is installed with the Billing Connector package.



3. Click the [New] button
4. Complete the following fields:
    - **Payment Gateway Name**
    - Check the **Active** box
    - Check the **Default** (Based on your requirements)
    - **Gateway Type** should be Chargent Gateway (See organization setup above if not present)
    - **Chargent Gateway** - Lookup to the Chargent Gateway record you created in the previous step

5. Click the [Save] button



Process Payments Using Your Gateway of Choice

Once the above is completed, your gateway can be used according to the documentation for Payment Gateway usage provided by Salesforce Billing. Salesforce provides great documentation for taking payments! Click here to see these docs.

You can use the Payment Virtual Terminal on Invoices, the Payment Scheduler, or any other Salesforce Billing feature where a payment can be taken.

Payment Methods Supported

Cards (Credit/Debit/Prepaid/Procurement)

Chargent Gateways Connector will support all cards supported by your payment gateway and payment processor.

Tokenization

Tokenization can be used for both cards and bank drafts (ACH/eCheck/EFT/Direct Debit).

Bank Account Tokenization

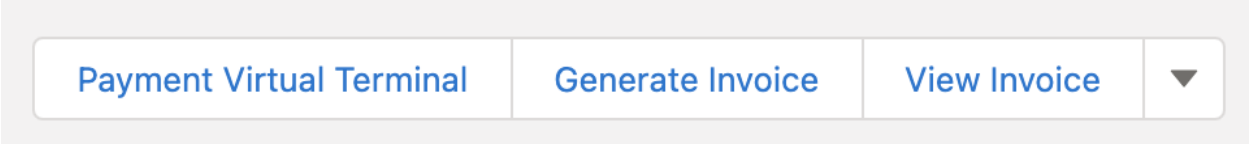
When creating a new bank payment method (direct debit/EFT/ACH) via the “New Payment Method” button in Salesforce Billing, the payment method will not immediately be tokenized. Only once a payment is made using that payment method, will Chargent perform the tokenization, if enabled.

Chargent for Salesforce Billing - Customer User Guide

Overview

Through cloud salesforce billing you can create and automate the invoices, payments, and revenue Salesforce billing is an add-on package that uses the key records and information from Salesforce CPQ. After an order is placed under Salesforce CPQ billing picks up the order and generates a record for an invoice, payment, and revenue.

Chargent for Salesforce Billing allows you to connect to your Gateway to accept payments and apply them to the balance of an open Invoice or an Account. You can use the Payment Virtual Terminal on Invoices, the Payment Scheduler, or any other Salesforce Billing feature where a payment can be taken.



Prerequisites

- Salesforce CPQ 222.2, or later must be installed.
- Salesforce Billing, Version Summer '19 220.7 or later must be installed.
- Consumption Schedule must be enabled. Contact Salesforce support for enablement if needed.
- Chargent for Salesforce Billing must be installed and configured based on the documentation.
- The Chargent Base Package must be installed.
- You have your Chargent Gateway setup using the Chargent Base package and Setup Wizard.
- You have a Salesforce Payment Gateway set up in Salesforce that connects to the Chargent Gateway record.

In Salesforce Billing a few steps need to happen before you can process a payment using the Chargent for Salesforce Billing component.

1. Create a new Opportunity
2. Create a Quote from the Opportunity
3. Create a new Order from the Quote
4. Check the Bill Now checkbox to create an Invoice record (if you are looking to apply payment towards Invoices).

Take Payments Using Your Gateway of Choice

Your gateway can be used according to the documentation for Payment Gateway usage provided by Salesforce Billing. Salesforce provides great documentation for taking payments! Click here to see these docs.

Payment Methods Supported

Cards (Credit/Debit/Prepaid/Procurement)

Chargent for Salesforce Billing will support all cards supported by your payment gateway and payment processor.

Tokenization

Tokenization can be used for both cards and bank drafts (ACH/eCheck/EFT/Direct Debit).

> Note: Tokenization for all payment methods is not supported for every gateway. You should check our documentation specific to your gateway to see if it supports tokenization.

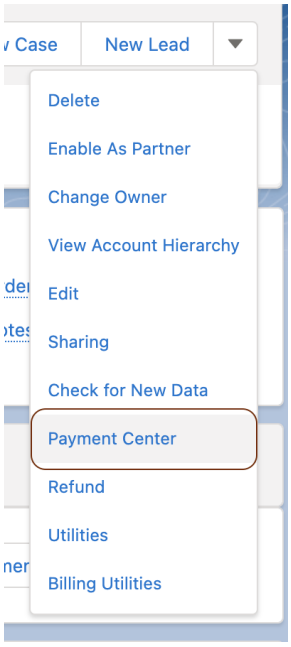
Bank Account Tokenization

When creating a new bank payment method (direct debit/EFT/ACH) via the “New Payment Method” button in Salesforce Billing, the payment method will not immediately be tokenized. Only once a payment is made using that payment method, will Chargent perform the tokenization, if enabled.

Processing Payments using Chargent for Salesforce Billing

Processing Payments Using the Payment Center

From an Account Record click the [Payment Center] button.

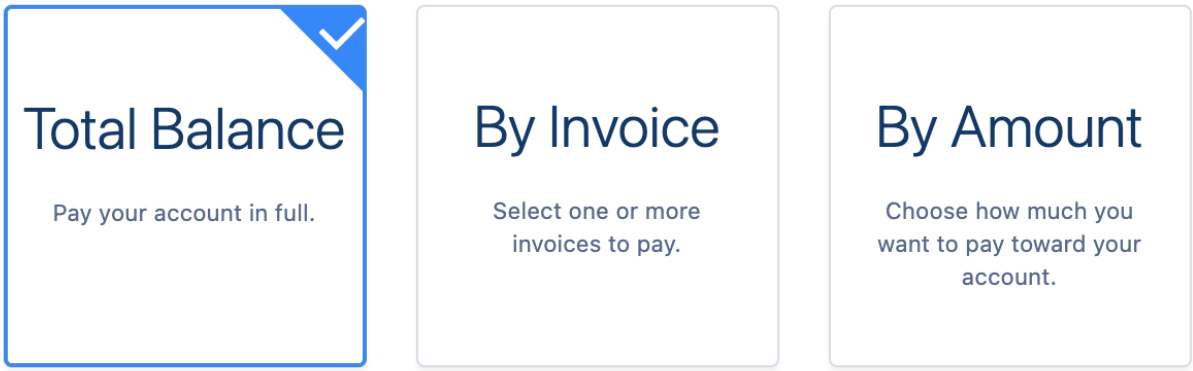


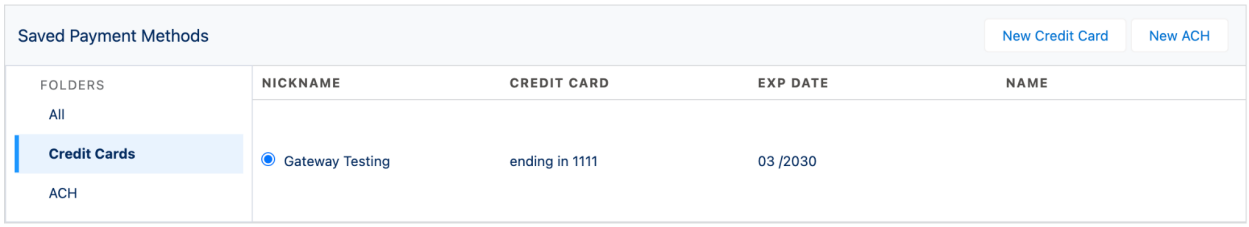
You will have 3 options

- Total Balance - Pay the complete balance due for a specific Account
- By Invoice (Recommended) - Choose an Invoice you want to apply a payment towards. This includes partial payments as well.
- By Amount - Select a specific amount to apply payment towards a specific Account.

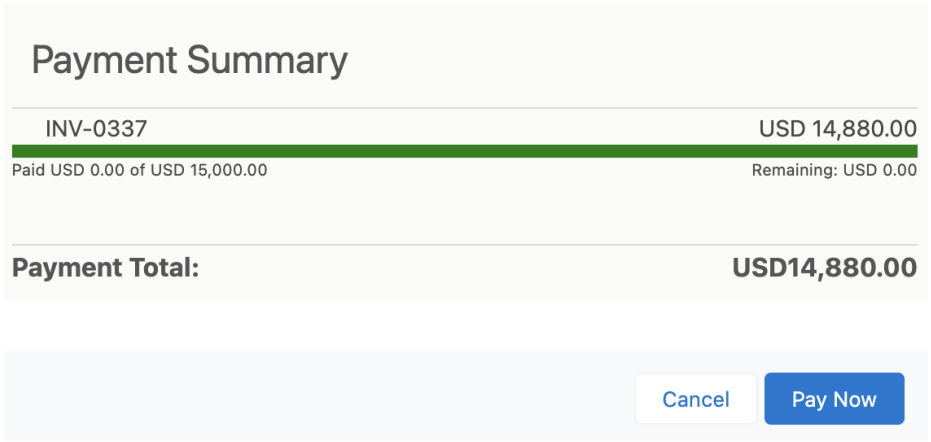
Total Balance

Selecting Total Balance and pressing the [Next] button, it will bring you directly to any stored credit cards or ACH bank information on file. You can then use the existing payment information to process the remaining balance of the Account.





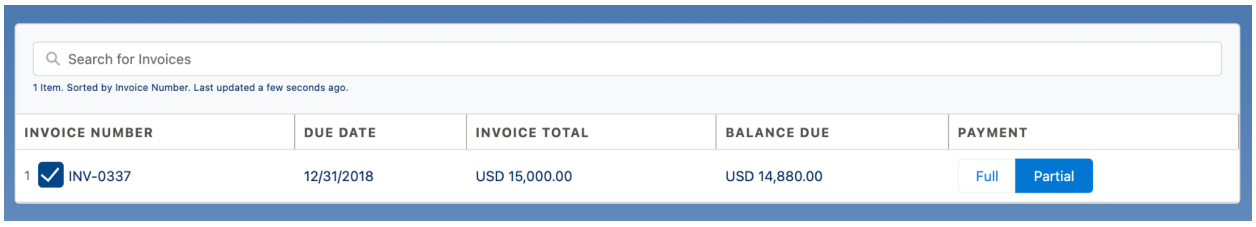
Clicking the [Pay Now] button will allow you to process the full balance.

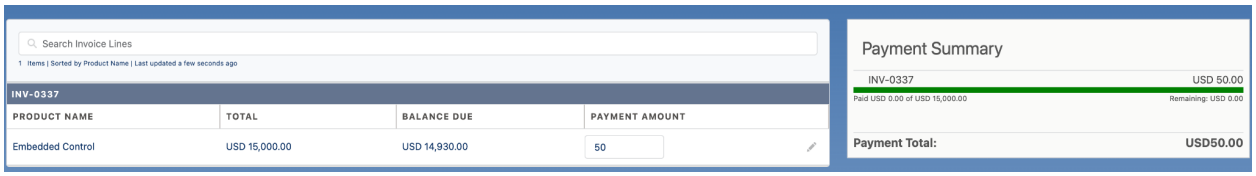


By Invoice (Recommended)

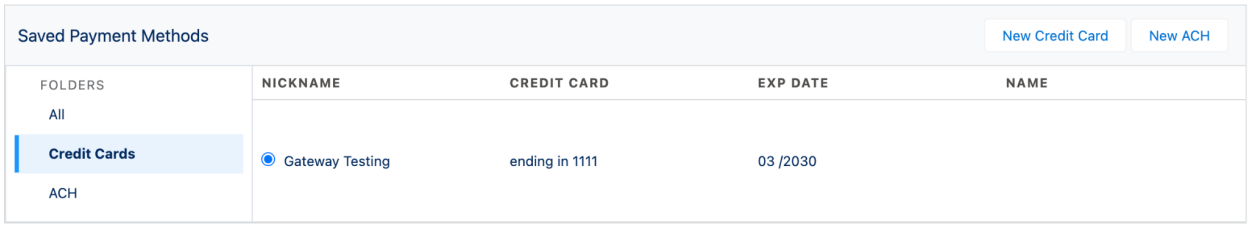
Selecting ‘By Invoice’ will allow you to choose to pay in Full or add a Partial Payment towards an open Invoice. Select the [+] next to the Invoice and either the [Full] or [Partial] option. Click the [Next] button to go to your stored Payment Methods.

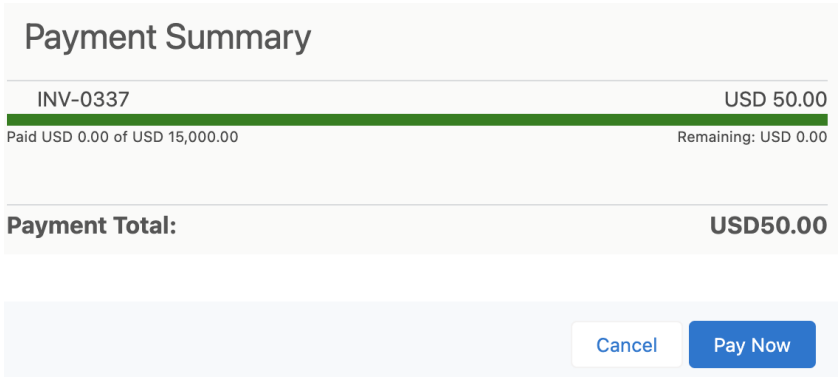
Note: If you choose [Partial] you will need first to edit the amount by clicking the pencil icon prior hitting [Next] and moving to the stored Payment Methods.





Select the Payment Method and click the [Pay Now] button to complete the transaction.





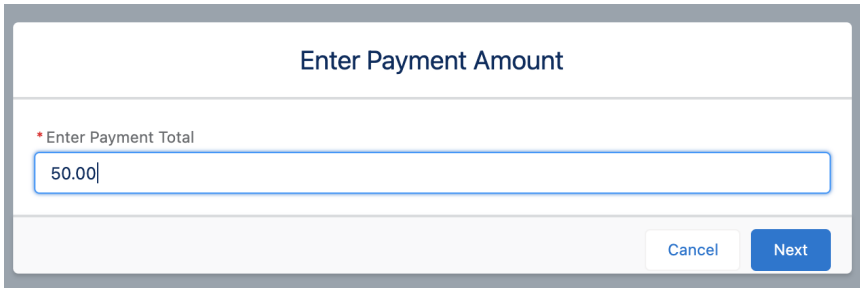
The Invoice will update

- Invoice Status is changed from Draft to Posted
- Invoice Payment Status is changed from Unpaid to Paid

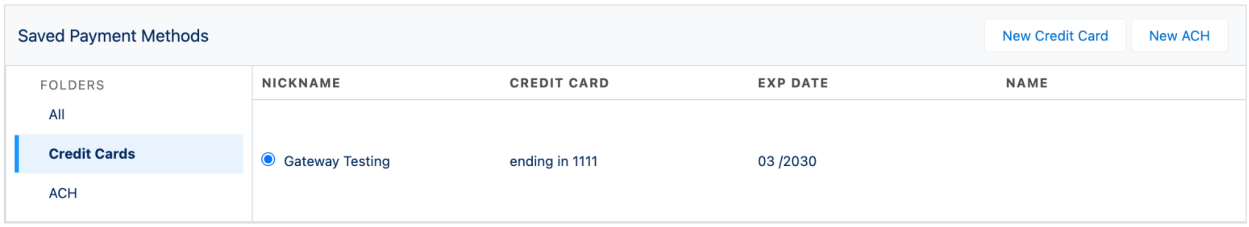
By Amount

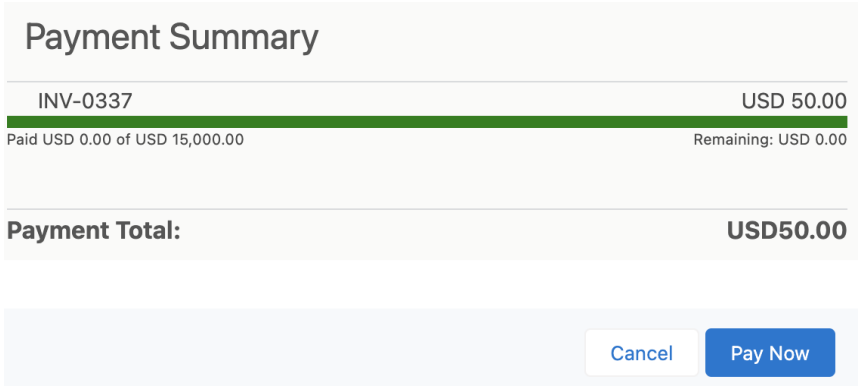
Selecting ‘By Amount’ it will allow you to enter an amount that will be applied to the Account’s balance.

Enter the specific amount and click [Next] to go to the Payment Method Page.

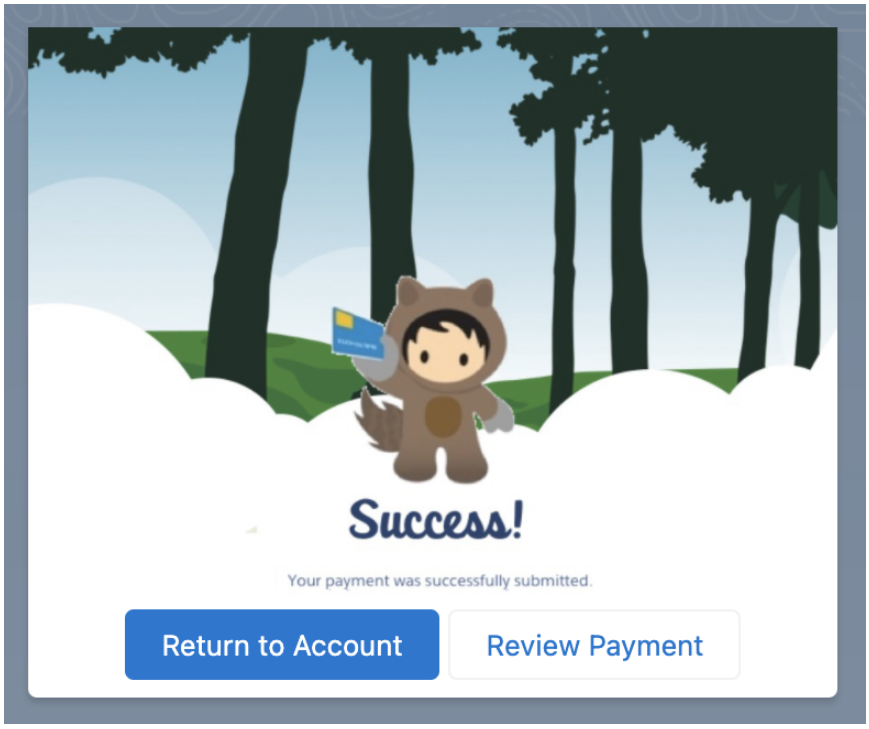


Select the Payment Method and click the [Pay Now] button to complete the transaction.



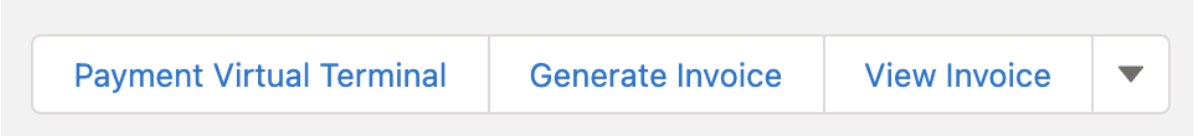


If the transaction is successful you will see the pop-up window showing Success. You will also have the option to Return to the Account or Review the Payment details. A Payment Transaction Record will be created logging the payment in your Salesforce Org.



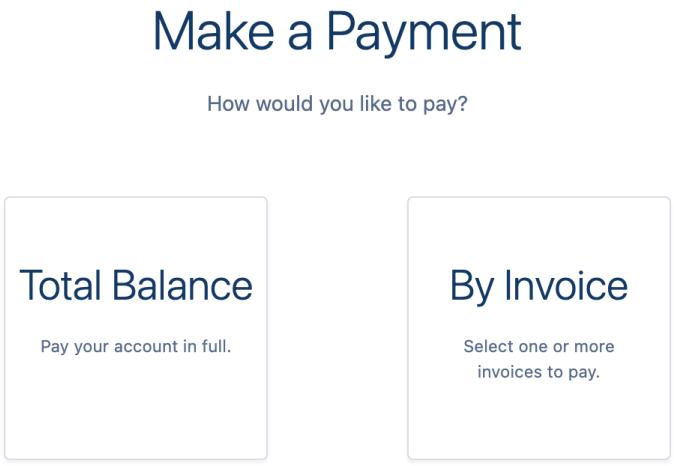
Processing Payments using Payment Virtual Terminal on the Invoice

Adding the Payment Virtual Terminal button to any of the Salesforce objects makes it even easier to process Payments anywhere in Salesforce Billing.



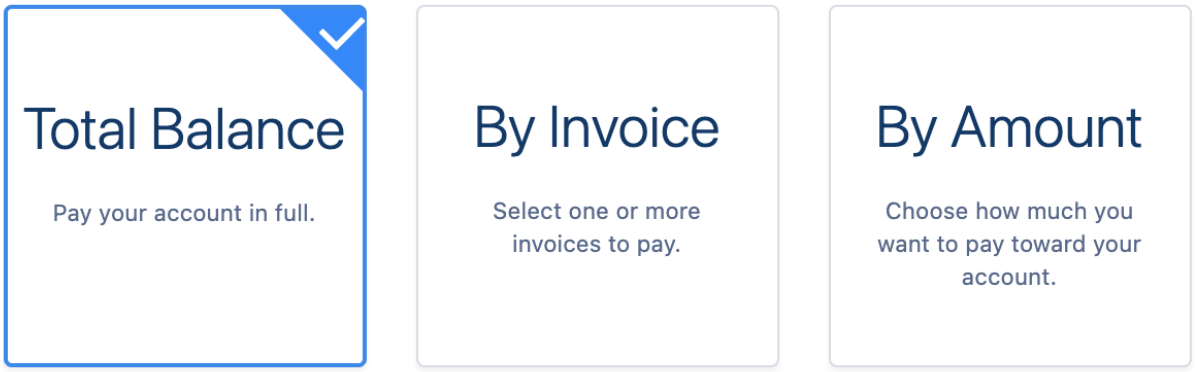
Similar to processing payments from the Payment Center, when you click the Payment Virtual Terminal button you will see the following payment options.

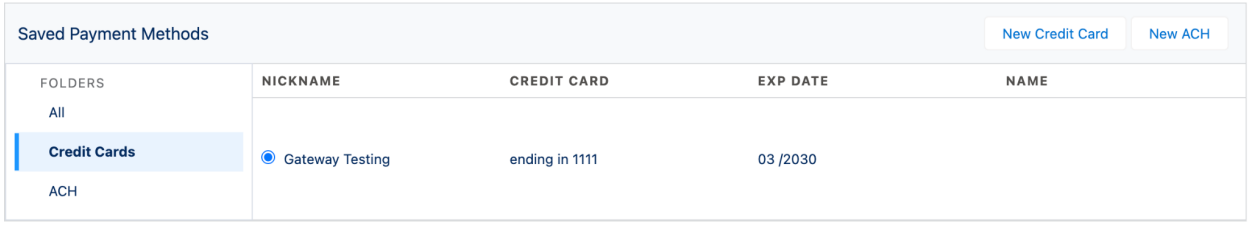
- Total Balance
- By Invoice



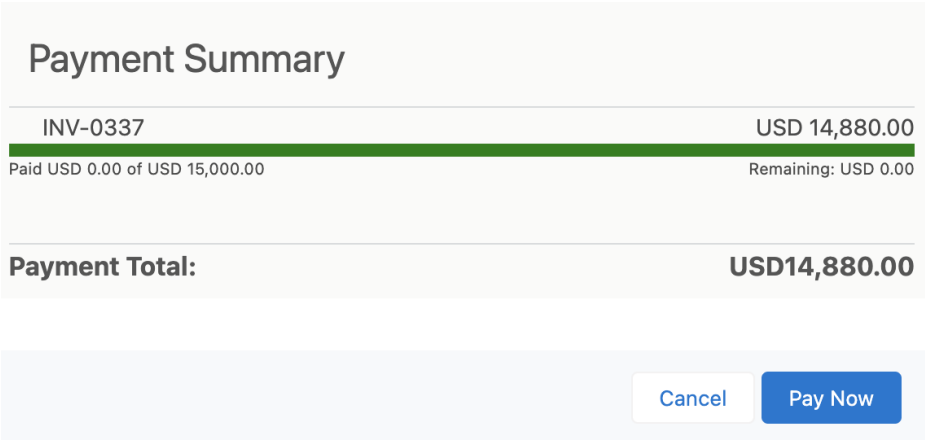
Total Balance

Selecting Total Balance and pressing the [Next] button, it will bring you directly to any stored credit cards or ACH bank information on file. You can then use the existing payment information to process the remaining balance of the Account.





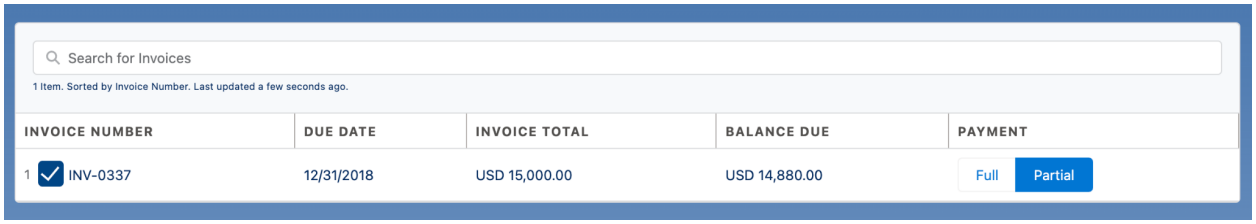
Clicking the [Pay Now] button will allow you to process the full balance.

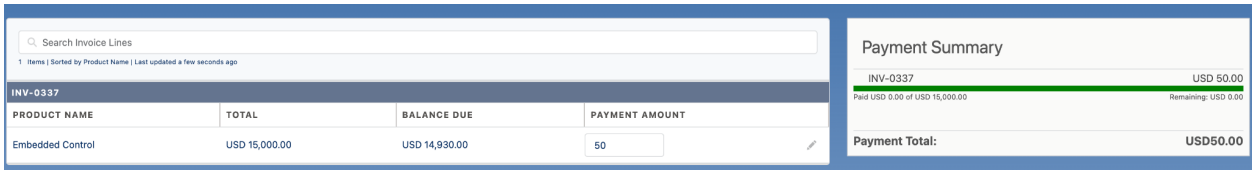


By Invoice (Recommended)

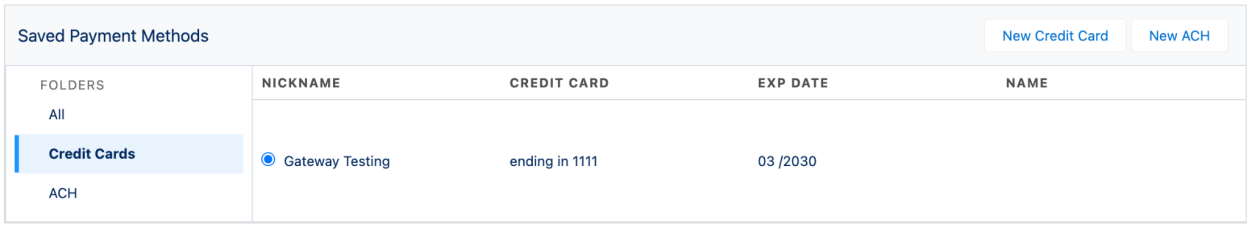
Selecting ‘By Invoice’ will allow you to choose to pay in Full or add a Partial Payment towards an open Invoice. Select the [+] next to the Invoice and either the [Full] or [Partial] option. Click the [Next] button to go to your stored Payment Methods.

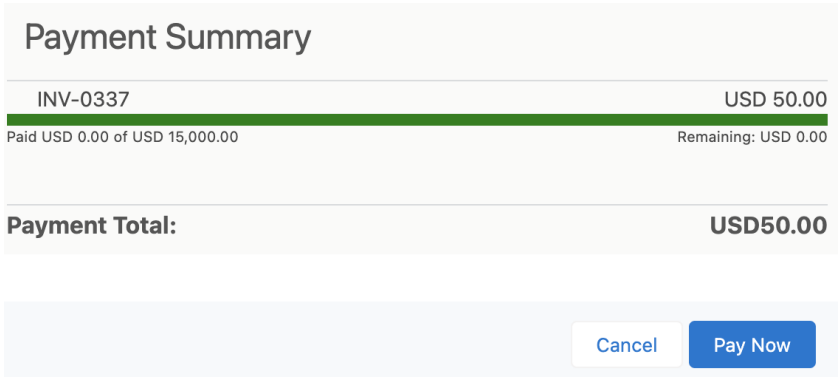
Note: If you choose [Partial] you will need first to edit the amount by clicking the pencil icon prior hitting [Next] and moving to the stored Payment Methods.





Select the Payment Method and click the [Pay Now] button to complete the transaction.

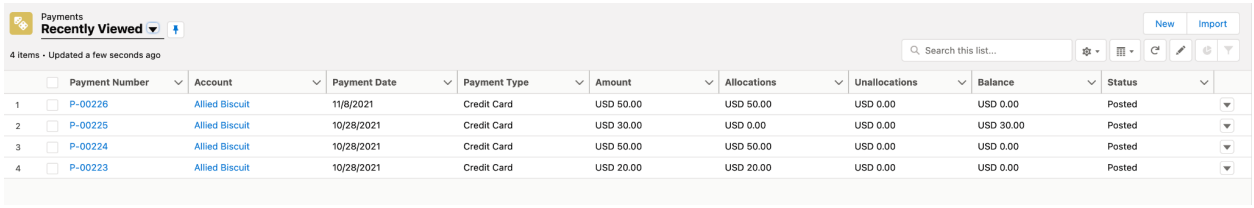




The Invoice will update

- Invoice Status is changed from Draft to Posted
- Invoice Payment Status is changed from Unpaid to Paid

Payment Transaction



Related **Details**

Payment Number
P-00226

Status
Posted

Account
Allied Biscuit

Payment Date
11/8/2021

Amount
USD 50.00

Payment Type
Credit Card

Notes

Payment Method
PM-0130

Invoice
INV-0337

Payment Mode
Electronic

Card Code Response

Paid By

Additional Information

Payment Description

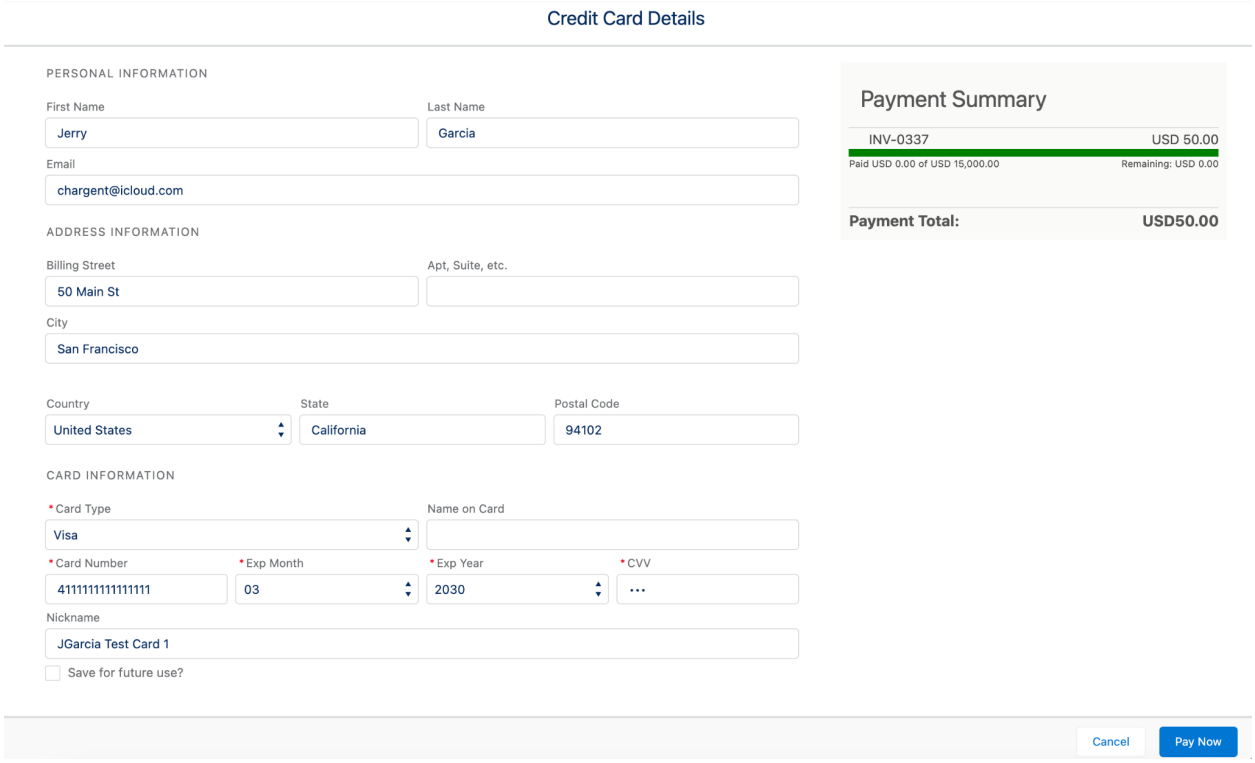
Transaction
TRX-000000269

Allocations / Unallocations

Allocations
USD 50.00

Unallocations
USD 0.00

Enter and Saving New Payment Methods



Salesforce Mapping

Keep in mind Salesforce sends the data to your Payment Gateway. Chargent does not control what fields are sent. You should check with Salesforce for accuracy. Here are some of the fields that will be mapped from your gateway to the Salesforce Payment Transaction record.

	Gateway Fields		Salesforce Billing Fields	Copied to Payment Method Record
Card Details	Card Type	<	Card Type	Payment Method Record
	Card Holder	<	First Name / Last Name	Payment Method Record
	Card Number	<	Card Number	Payment Method Record
	Expiry Date	<	Expiration Month / Expiration Year	Payment Method Record
Shopper Detail	Reference	<	Account Record ID	
	Token	<	Payment Gateway Token	Payment Method Record
	Billing Address	<	Billing Address 1, Billing Address 2, Billing City, Billing State, Billing Zip Code	Payment Method Record

Chargent for Commerce Cloud

Connecting payment gateways to B2B2C Commerce Cloud, B2B Commerce Cloud and the Order Management System is hard. Chargent makes gateway connections easy. Just install and start processing.

For more information check out the Chargent for Commerce Cloud on the AppExchange.

[embed video: Salesforce Commerce Cloud Explained With Chargent, https://www.youtube.com/watch?v=x2w1BhClT9A]

Zero Footprint Tokenization (Developer Lightning Component)

Overview

The Zero Footprint tokenization component is a lightning component that allows the developer to quickly implement tokenization of payment data in a configurable UI anywhere within your salesforce ecosystem. The component will fire an event upon tokenization for you to obtain the necessary results and optionally can update the payee details with the entered data as well as create a corresponding chargent order and chargent transaction.

Placement

The component does not require an underlying SObject and can be placed within a lightning app as well as within your own lightning component. While it does not require an underlying SObject for placement, it does require an SObject that supports the configurable payee detail fields.
For example, the component can be placed within a component in your lightning application and use the SObject Contact to support the payee fields that you wish to display

Usage

The component can simply be used within your markup for your lightning component. There are some parameters that need to be set as well as some optional parameters. These are outlined below

Component

<ChargentBase:zeroFootprint_Tokenization ..../>

Parameters

GatewayId (Required)

This is the salesforce Id of the gateway record to use for the tokenization.

Note: The payment entry screen will respect the settings on this gateway record. If, for example, you have not selected Credit Card as a payment option then it will not be displayed. Another example, would be if you have not selected Visa as an accepted credit card type then it would not allow the user to complete the tokenization with a visa card number.

If the gateway has the “Require CVV in payment console” set to true then the payment entry screen will also require the CVV to be entered.

buttonLabelPayeeSubmit (default: Next)

When in one column mode this will be the label on the button to submit the entered payee details and continue to the next screen to collect the payment information.

buttonLabelTokenize (default: Submit)

This is the label of the button to tokenize the payment information. In one column mode this will be the button on the second screen where the user enters the payment information. In two column mode this will be the only button displayed.

twoColumnLayout (default: false)

This indicates the display mode.

When false, the component will present the payee details first, then the user clicks the payee submit button and is taken to the payment entry screen where they will enter their payment data and click the tokenize button. Useful for small modal windows and the narrow side of a lightning layout.

When true, the component will present both the payee details and payment information side by side on one screen. Then the user enters the appropriate data and then clicks the tokenize button. Useful for medium to large modal windows and the wide side of a lightning layout.

payeeRecordId (Id - not required)

If using an pre-existing record to pre-populate or associate the payee details with set this Id to the desired record Id. This field is not required. If not set and “updatePayeeRecord” is set to true a new payee object will be created.

payeeObjectType (String - Required)

This is the Object type of the payeeRecordId (if set) or the object type of the object where the specified payee fields will be pulled from. This is a required parameter and must be set in order to populate the fields on the payform.

How the component knows how to display each of the specified payee fields according to their type.

payeeRecordFieldList (String[] - Required)

This is a comma separated list of field API names you wish to display from the payee object specified. These are the fields the user will be populating when they use the component.

An example if using the Contact Object Type:
FirstName,LastName,MailingStreet,MailingCity,MailingState,MailingPostalCode

payeeToChargentOrderFields (Object - Required)

While no underlying SObject is required to place the component, it does rely on an abstract version of the Chargent Order. In order to properly associate the fields you wish to display with the appropriate data to send to the gateway this mapping will need to be provided.

The mapping is a javascript object with key value pairs wrapped in {}. The key is the payee SObject Field API Name and the value is the corresponding Chargent Order field API name

Key: Payee SObject Field API name

Value: Corresponding Chargent Order field API name

An example mapping using the contact fields specified above:

{'FirstName':'ChargentOrders__Billing_First_Name__c','LastName':'ChargentOrders__Billing_Last_Name__c','MailingStreet':'ChargentOrders__Billing_Address__c','MailingCity':'ChargentOrders__Billing_City__c','MailingState':'ChargentOrders__Billing_State__c','MailingPostalCode':'ChargentOrders__Billing_Zip_Postal__c'}

This is passed in as one long string. In order to see it a bit more clearly we have prettified it a bit below.

Do not pass the information in the below format

{

"FirstName": "ChargentOrders__Billing_First_Name__c", "LastName": "ChargentOrders__Billing_Last_Name__c", "MailingStreet": "ChargentOrders__Billing_Address__c", "MailingCity": "ChargentOrders__Billing_City__c", "MailingState": "ChargentOrders__Billing_State__c", "MailingPostalCode": "ChargentOrders__Billing_Zip_Postal__c"

}

In the above examples you can see it has been specified that the FirstName be associated with the ChargentOrders__Billing_First_Name__c field and so on. This is dynamic and will depend on your configuration for the payee fields and SObject type

orderStaticValueMap (Object - Not required)

This parameter is used to pass specific static information to the gateway during tokenization but do not wish to display that information to the user or obtain their input. The fields that can be passed are limited to:

- ChargentOrders__Payment_Method__c - ChargentOrders__OrderSource__c

- ChargentOrders__Order_Information__c - ChargentOrders__Description__c

- ChargentOrders__Currency__c

- ChargentOrders__Customer_IP__c

This is similar to the format of the payeeToChargentOrderFields parameter except the key is the Chargent Order Field API Name and the value is the value to be sent.

Key: Corresponding Chargent Order field API name Value: value to be passed through to the gateway

An example using order information is: {'ChargentOrders__Order_Information__c':'An example Information'}

createChargentOrder (Default: false)

This parameter indicates if a Chargent Order and related chargent transaction record should be created as a result of the tokenization.

updatePayeeRecord (Default: false)

This parameter indicates if you wish to have the specified payee record updated or in the case of no payeeRecordId parameter created. If this is not set to true the entered information will be passed on to the gateway but the underlying payee record will not be altered or created.

Tokenization Event

When the tokenization is complete an event named ChargentBase:tokenizationEvent is fired. Components including the Zero Footprint tokenization component can handle this event and perform actions based on the results of the tokenization event.
Event Parameters

Parameter	Type	Description
success	Boolean	Tokenization successful
accountName	String	Credit card or bank account holder name
accountFirstName	String	Credit card or bank account holder first name
accountLastName	String	Credit card or bank account holder last name
accountStreet	String	Account holder billing street
accountCity	String	Account holder billing city
accountState	String	Account holder billing state
accountPostalCode	String	Account holder billing postal code
cardExpirationMonth	String	Credit card expiration month
cardExpirationMonth	String	Credit card expiration month
cardExpirationYear	String	Credit card expiration year
accountLast4	String	Last four of the credit card or bank account number
bankRoutingNumber	String	Bank account routing number
paymentMethod	String	Credit Card or eCheck

paymentType	String	The card type or bank account type
gatewayStatus	String	The status returned by the gateway. If there was a platform error this will be set to “ERROR”
gatewayMessage	String	Either the gateway message or Exception message
gatewayReasonText	String	The returned reason text from the gateway
gatewayReasonCode	String	The returned reason code from the gateway
gatewayId	String	The id assigned to this transaction by the gateway
gatewayResponse	String	Full response from gateway
paymentToken	String	The returned payment token, if any
chargentTransactionId	String	The salesforce Id of the created transaction if createChargentOrder is set to true
payeeRecordId	String	The id of the payee record that was created or updates during this transaction. Wil be null if the updatePayeeRecord parameter is false

Example

An example implementation of the component placed on the opportunity object embedding the Zero Footprint tokenization component within it.
This example can be used to quickly create a lightning component and begin playing with the various options.

Design

```
<design:component>
	<sfdc:objects>
		<sfdc:object>Opportunity</sfdc:object>
	</sfdc:objects>
	<design:attribute description="Gateway Id" name="gatewayId"
label="Gateway Id"/>
	<design:attribute description="Create or Update Payee" name="updatePayeeRecord"
label="Update Payee Record" default="false"/>
	<design:attribute description="Label for payee submit button"
name="buttonLabelPayeeSubmit" label="Label for Payee Submit button" default="Next"/>
	<design:attribute description="Label for tokenization button"
name="buttonLabelTokenize" label="Label for tokenization button" default="Submit"/>
	<design:attribute name="twoColumnLayout" default="false" label= "Two Column
```

```
Layout" description="show payee details and payment entry side by side in two columns"/>
        <design:attribute name="contactLookupAPIName" l abel="Billing Contact Lookup API
Name" description="The API name of the billing contact lookup on Opportunity"/>
        </design:component>
```

Component

```
<aura:handler name="init" value="{!this}" action="{ !c.doInit}"/>
<aura:attribute name="gatewayId" type="String" required="true" access="global"/>
<aura:attribute name="updatePayeeRecord" type="Boolean" access="global"/>
<aura:attribute name="buttonLabelPayeeSubmit" type=" String" default="Next"
access="global"/>
<aura:attribute name="buttonLabelTokenize" type="String" default="Submit"
access="global"/>
<aura:attribute name="contactLookupAPIName" type="String" access="global"
required="true"/>
<aura:attribute name="twoColumnLayout" type="Boolean" default=" false" access="global"/>
<aura:attribute name="opportunityFields" type="String[]" access="private"/>
        <aura:attribute name="opportunityRecordFields" type="Object" description="The fields
object for reference in markup"
access="private"/>
<aura:attribute name="opportunityRecord" type="Object"
description="The opportunity record" access="private"/> <aura:attribute
name="opportunityRecordError" type=" String"
access="private" description="Error messages associated with lightning data service for the
opportunity record"/>
<aura:handler name="tokenEvent" event="c:tokenizationEvent"
action="{!c.handleTokenization}"/ >
<force:recordData aura:id="opportunityRecord" layoutType="FULL"
recordId="{!v.recordId}" targetRecord="{!v.opportunityRecord}"
targetFields="{!v.opportunityRecordFields}" fields="{!v.opportunityFields}"
targetError="{!v.opportunityRecordError}" mode="VIEW"/>
<lightning:card title="Payment Tokenization"> <div class="slds-p-around_medium">
<div aura:id="tokenComponent"> <aura:if
isTrue="{!not(empty(v.opportunityRecordError))}"> <div
class="{!if(empty(v.opportunityRecordError),'slds-hide','recordError')}">
severity="error" closable="true">
<ui:message title="Error"
{!v.opportunityRecordError}
</ui:message>
</div>
<aura:set attribute="else">
<ChargentBase:zeroFootprint_Tokenization twoColumnLayout="{!v.twoColumnLayout}"
buttonLabelTokenize="{!v.buttonLabelTokenize}"
```

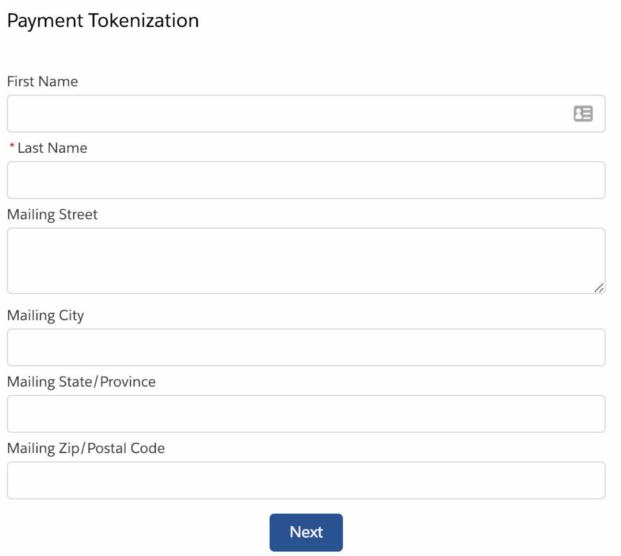
```
buttonLabelPayeeSubmit="{!v.buttonLabelPayeeSubmit}"
payeeRecordFieldList="FirstName,LastName,
MailingStreet,MailingCity,MailingState,MailingPostalCode"
updatePayeeRecord="{!v.updatePayeeRecord}" payeeObjectType="Contact"
GatewayId="{!v.gatewayId}"
payeeRecordId="{!v.opportunityRecordFields.ChargentBase__Billing_Contact__c }"
payeeToChargentOrderFields="{'FirstName':'ChargentOrders__Billing_First_Nam
e__c','LastName':'ChargentOrders__Billing_Last_Name__c','MailingStreet':'Ch
argentOrders__Billing_Address__c','MailingCity':'ChargentOrders__Billing_Ci
ty__c','MailingState':'ChargentOrders__Billing_State__c','MailingPostalCode
':'ChargentOrders__Billing_Zip_Postal__c'}"
orderStaticValueMap="{'ChargentOrders__Order_Information__c':'An example Information'}
/> </aura:set>
</aura:if> </div>
<div aura:id="result-div" class="slds-grid slds-size_1-of-1" id="result-div"></div>
</div>
</lightning:card> </aura:component>
```

Controller

```
({
doInit: function(component,event,helper){
component.set("v.opportunityFields",component.get("v.contactLookupAPIName") );
},
handleTokenization: function(component,event,handler){
} })
var result = JSON.stringify(event.getParams(),null,2);
$A.util.addClass(component.find("tokenComponent"),'slds-hide');
component.find("result-div").getElement().innerText = result;
```

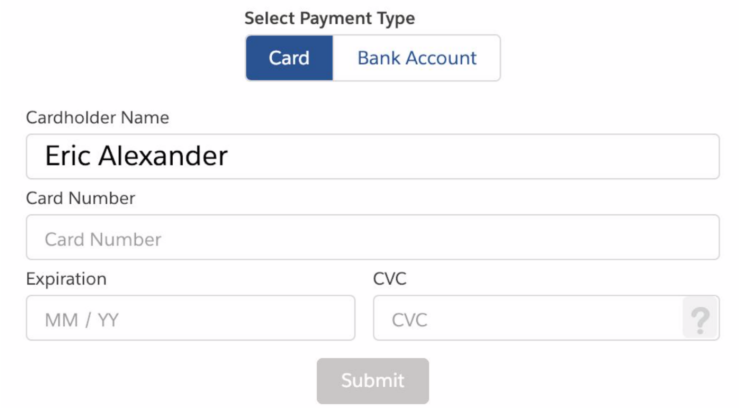
Images of Results

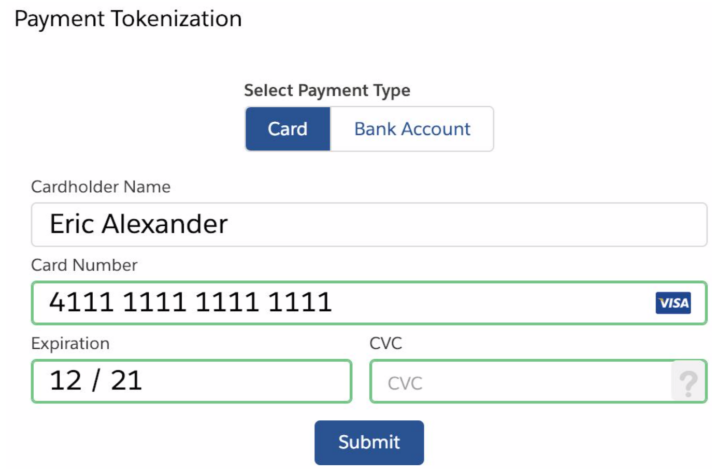
One column Layout
Payee Details



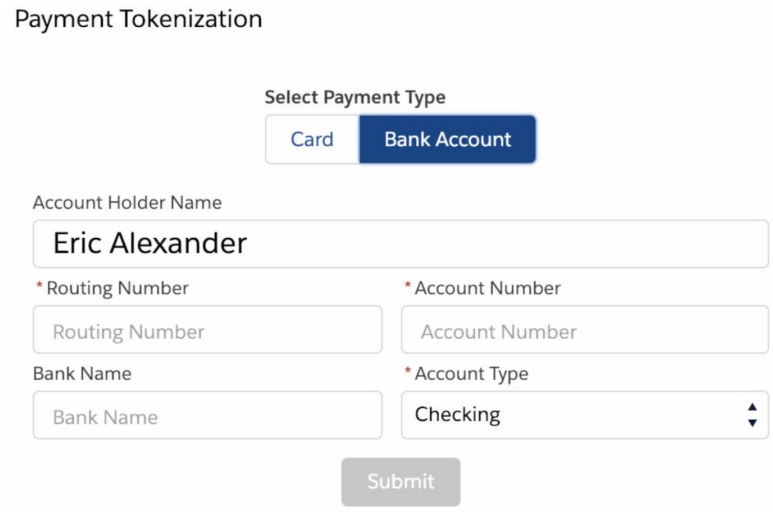
Payment Entry Screen

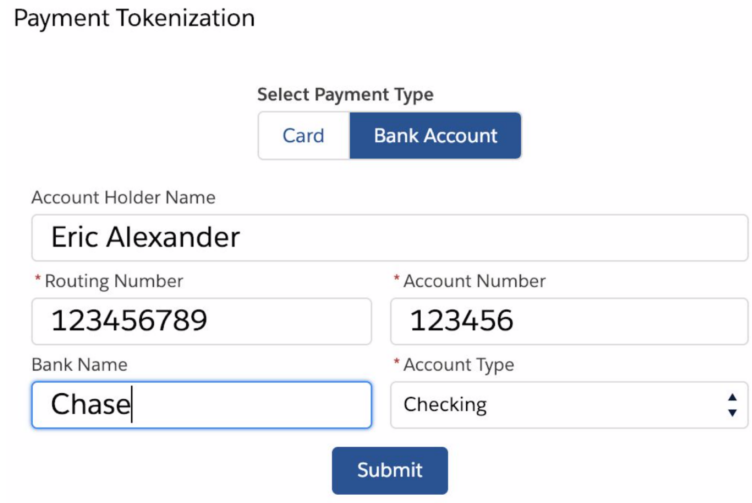
Credit Card





Bank Account



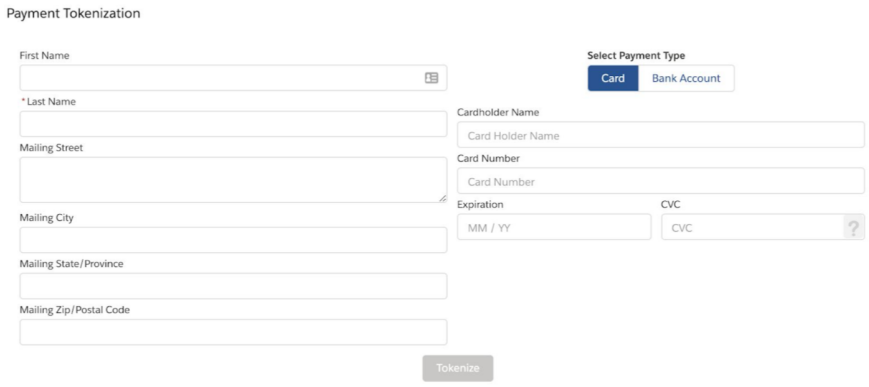


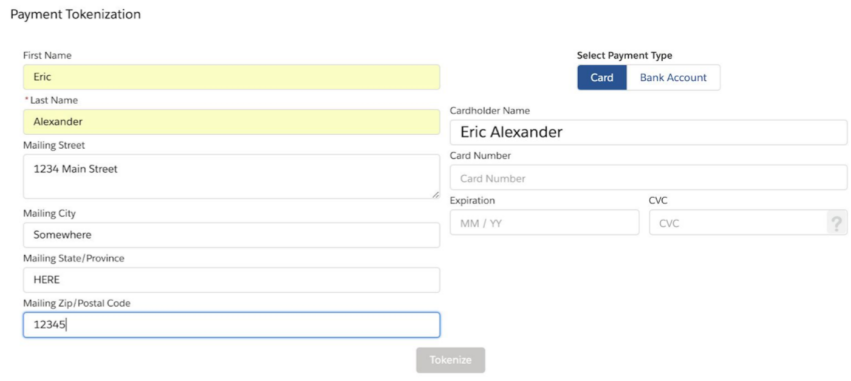
Results

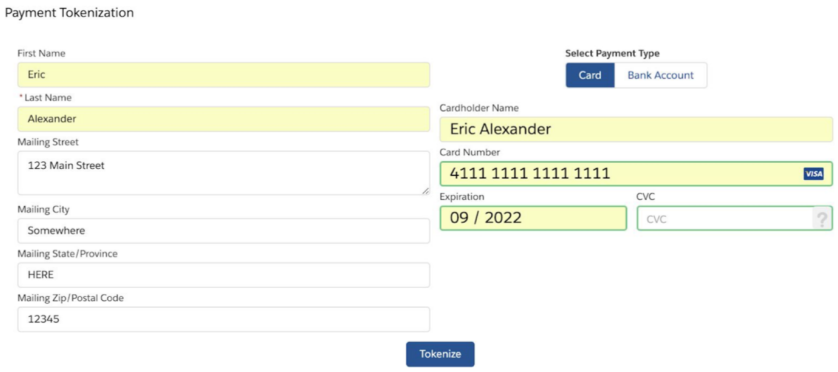
Payment Tokenization

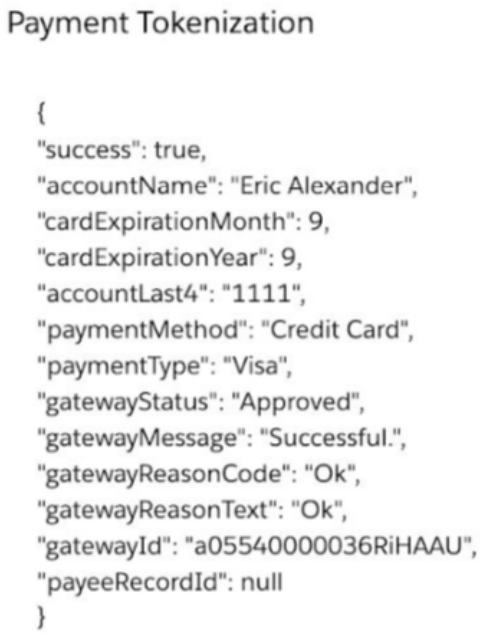

```
{
"success": true,
"accountName": "Eric Alexander",
"cardExpirationMonth": 12,
"cardExpirationYear": 12,
"accountLast4": "1111",
"paymentMethod": "Credit Card",
"paymentType": "Visa",
"gatewayStatus": "Approved",
"gatewayMessage": "This transaction has been approved.",
"gatewayReasonCode": "1",
"gatewayReasonText": "This transaction has been approved.",
"gatewayId": "a05540000036RiHAAU",
"payeeRecordId": null
}
```


Two Column Layout







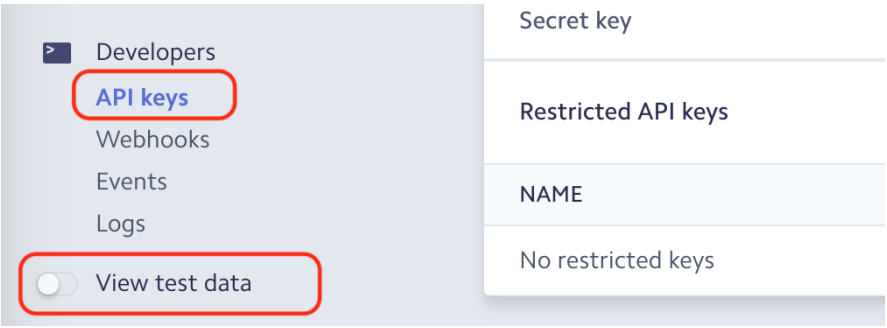


Accept Card Payments

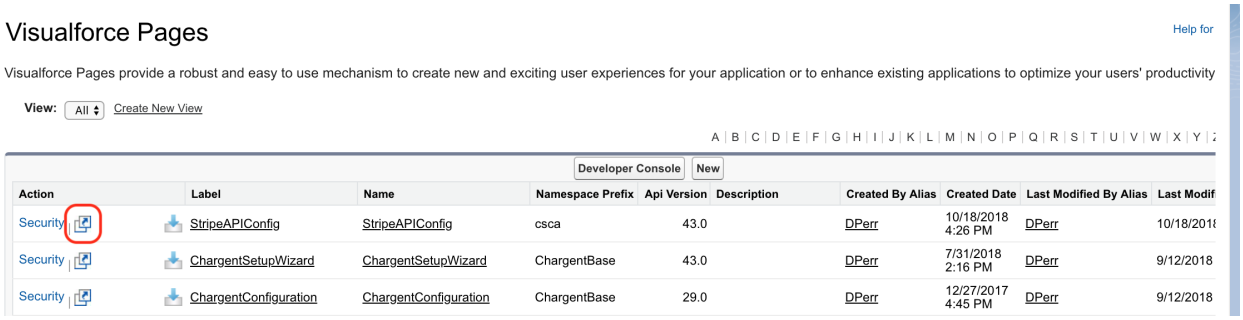
1. Download the package from the AppExchange
2. Create a Stripe account if you don't already have one.

Getting your Stripe API Secret Key

1. Login to Stripe and Copy the API Secret Key
    a. Go under Developer > API keys > Reveal live/test Secret Key (for testing you should enable the View Test Data switch in Stripe.



2. In Salesforce you want to configure the Visualforce Page
    a. Click the gear icon in the top right and choose Setup
    b. Navigate to Custom Code > Visualforce Pages
    c. Click the Preview in a New Window icon next to the StripeAPIConfig page.



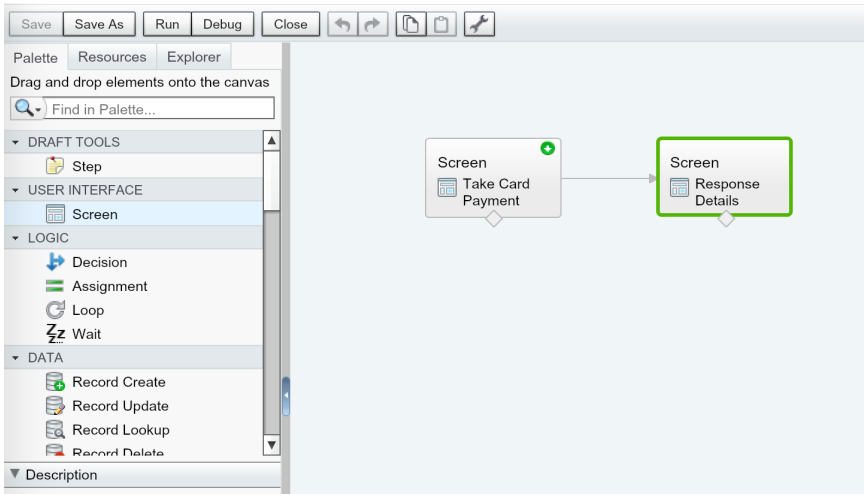
3. Copy your API Key and click Store Key (Note: If you've already stored an API key it will prompt you to Update API Key)

Create the Flow in Salesforce

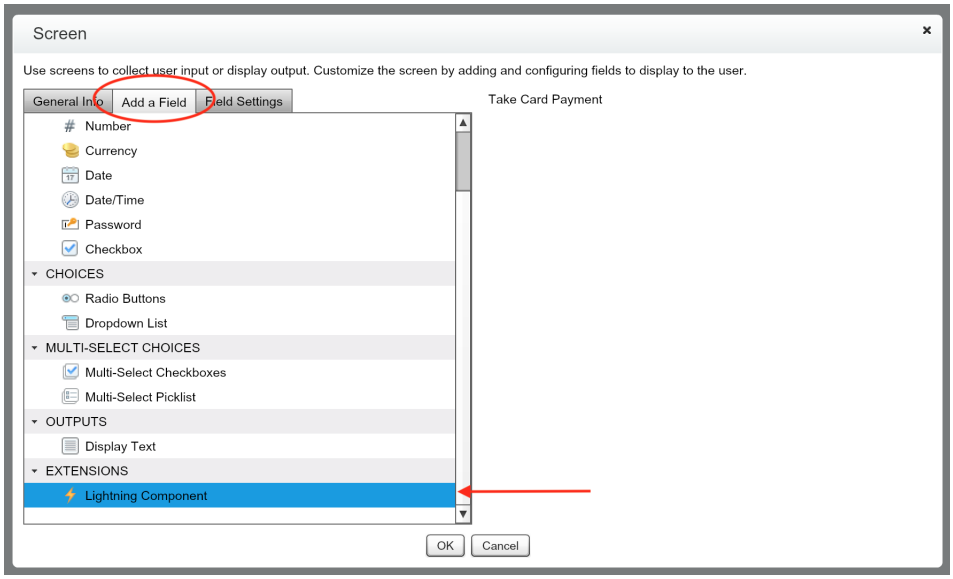
1. Click the gear icon in the top right and choose Setup
2. Under Process Automation select Flows

Step 1 Create the First Screen

1. Click New Flow
2. From the Pallette drag Screen into the Flow



3. Name your Screen (Ex: Take Card Payment)
4. Under the Add a Field tab select Lightning Component



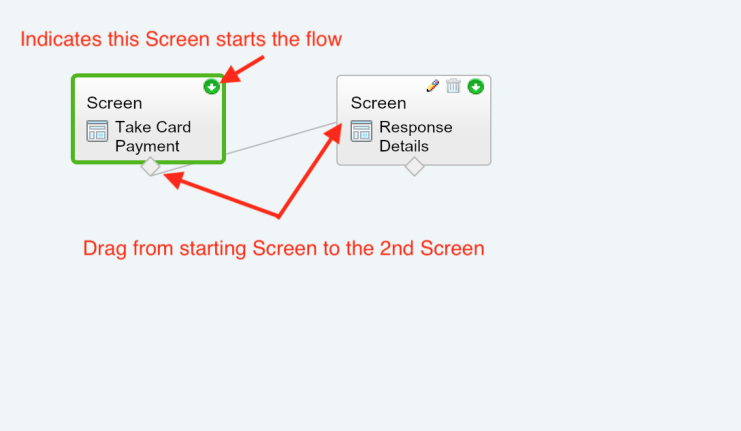
5. From the Field Settings tab
    a. Add a Unique Name
    b. Choose the Lightning Component csca:FlowStripeCharger
    c. Input Status (the default text that you would like in your flow)
        i. Billing Name; (choose a default billing name for your form, ex: John Wayne or Mickey Mouse)
        ii. Charge Amount: (choose the default amount for your form)
    d. Output
        i. Attribute = Response Status
        ii. Variable = {!ResponseStatus}

Step 2: Create the Second Screen

1. From the Pallette drag a new Screen into the Flow
2. Name your Screen Response Details
3. Under the Field tab select Display Text
4. Under the Add a Field tab choose Display Text
5. Under Field Settings Name the Field Response
6. Select the Resource {!ResponseStatus}

Connecting your Screens

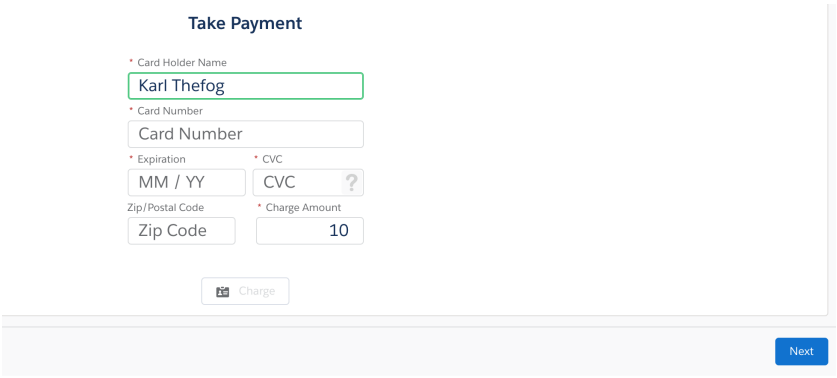
1. Run your mouse over the first Screen and click the green arrow pointing down. This indicates that this is the first Screen and the start of the flow.
2. Click and drag the diamond at the bottom of your first Screen to the 2nd Screen. This will create an arrow to connect the two. It determines the path the flow will take.



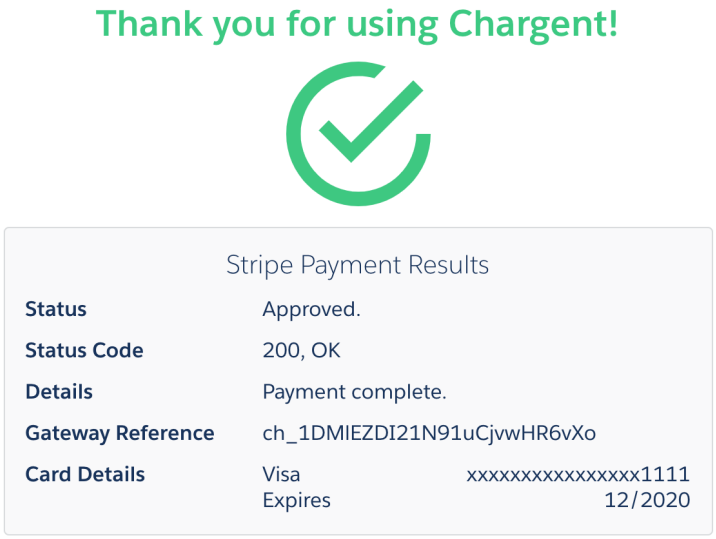
3. Click Save and name the Flow

Testing your Flow

1. Click Run to test the flow.



2. Use Credit Card Number 4111 1111 1111 1111 with any other information to receive an approved status.



[embed video: How To Accept Credit Card Payments In Salesforce, https://www.youtube.com/watch?v=iyt1ROkdPvI]