Salesforce Billing Developer Guide

Version 60.0, Spring ’24

@salesforcedocs

Last updated: December 15, 2023

CONTENTS

CHAPTER 1 Salesforce Billing Developer Guides & Lightning Components

In this chapter ...

Want to get started developing with Salesforce Billing? Check out developer guides, plugins, and Lightning components.

Salesforce Billing Payment Gateway Developer Guide

Use Salesforce Billing gateway API to enable communication between Salesforce Billing and external payment gateways. Salesforce Billing can integrate with payment gateways that process credit card and ACH transactions. While we provide several default integration options, you can also use our payment gateway API to support custom integrations.

Get Started with Salesforce Billing Payment Gateway API

Use Salesforce Billing gateway API to enable communication between Salesforce Billing and external payment gateways.

Salesforce Billing Global Payment API

Salesforce Billing exposes payment transactions as global API. When you're working with our transaction API in your payment gateway, review important guidelines.

Salesforce Billing acts as a pass-through layer for transactions. It routes the processing of input parameters to a payment gateway package and then returns the result of the transaction to the API caller.

Important: Salesforce Billing doesn't perform any implicit operations as a result of calling the transaction API. Our global methods provide only a layer to interact with a payment gateway to process a payment transaction. For example, calling the `chargeTransaction` global API on its own doesn't create a payment transaction object or a payment object.

All global methods in the `blng` namespace use the following input parameter structure.

```
Input - Map<String, TransactionParameter> mapOfTransactionParameterById
```

The map represents a unique string, such as an invoice number. The value is an instance of the `TransactionParameter` class, which contains the information needed to build a payment request.

All global methods in the `blng` namespace use the following output parameter structure.

```
Output - List<TransactionResult>
```

When a payment gateway sends a response following a transaction request, your payment gateway package evaluates the response and stores the information in the `TransactionResult` class.

Salesforce Billing global payment API includes the following methods.

API Method Name	Description
`generateToken`	Generates a token for a given input.

API Method Name	Description
`authorizeTransaction`	Authorizes a transaction.
`captureTransaction`	Captures an unauthorized transaction.
`voidRefundTransaction`	Voids a refund transaction.
`chargeTransaction`	Charges a transaction.
`voidTransaction`	Voids a payment transaction (charge or capture).
`refundTransaction`	Refunds a payment transaction for a payment that was made previously in the same gateway.
`voidTokenTransaction`	Voids a token.
`getPaymentStatus`	Identifies the status of a payment transaction.
`getRefundStatus`	Identifies the status of a refund transaction.
`nonReferredRefund`	Performs a nonreference refund.

Sending Transaction Information with TransactionParameter

The transaction parameter class represents a list of transaction field values that Salesforce Billing sends to the payment gateway package. You can set the class's attributes through Salesforce Billing global API, the Payment Center, or a payment run.

For complete reference information on the `TransactionParameter` class's methods and parameters, review TransactionParameter Class.

This table shows how to set `TransactionParameter`'s attributes for transactions made through Salesforce Billing Global API, the Payment Center, or payment runs. Billing Global API attributes are optional unless otherwise indicated.

TransactionParameter Attribute	Source: Billing Global API	Source: Payment Center	Source: Payment Run
PaymentMethod	`setPaymentMethod(paymentMethod)`	Set by the Billing package.	Set by the Billing package.
Invoice	`setInvoice(invoice)` Required for sending L2 and L3 parameters when making a payment against an invoice.	Set by the Billing package for payments made against an invoice.	Set by the Billing package for all payments.

TransactionParameter Attribute	Source: Billing Global API	Source: Payment Center	Source: Payment Run
InvoiceLine	`setInvoiceLine(line)` Sets a list of invoice lines. Required for sending L2 and L3 parameters when making a payment against one or more invoice lines.	Set by the Billing package for payments made against an invoice line.	Not set.
Account	`setAccount(accountInstance)`	Set by the Billing package.	Set by the Billing package.
RequestBody	`setRequestBody(requestBody)`	Set by the payment gateway package. Contains the request sent to the gateway.	Set by the payment gateway package. Contains the request sent to the gateway.
FirstName	`setFirstName(firstName)`	Set by the Billing package.	Set by the Billing packag.e
LastName	`setLastName(lastName)`	Set by the Billing package.	Set by the Billing package.
EmailId	`setEmailId(emailId)`	Set by the Billing package.	Set by the Billing package.
Street	`setStreet(street)`	Set by the Billing package.	Set by the Billing package.
City	`setCity(city)`	Set by the Billing package.	Set by the Billing package.
State	`setState(state)`	Set by the Billing package.	Set by the Billing package.
ZIP code	`setZipCode(zipCode)`	Set by the Billing package.	Set by the Billing package.
Country	`setCountry(country)`	Set by the Billing package.	Set by the Billing package.
Phone	`setPhone(phone)`	Set by the Billing package.	Set by the Billing package.
Amount	`setAmount(amount)`	Set by the Billing package.	Set by the Billing package.
Transaction	Not set	Not set	Set by the Billing package.
RequestingInvoiceId	`setRequestingInvoiceId(requestingInvoiceId)`	Set by the Billing package for L2 and L3 parameters.	Set by the Billing package for L2 and L3 parameters.
CardCodeResponse	`setCardCodeResponse` Required only for reference refund transactions made through Payeezy.	Set by the Billing package for reference refunds made through Payeezy.	Not set
MerchantId	`setMerchantId()`	Set by the Billing package for reference refunds made through Payeezy.	Not set
GatewayId	`setGateway(gateway)` Sets the transaction ID for the payment. Required only for reference refund transactions.	Set by the Billing package as Gateway reference ID for payment. Used in reference refund transactions.	Not set

TransactionParameter Attribute	Source: Billing Global API	Source: Payment Center	Source: Payment Run
	Required to set by caller for Reference Refund transaction. It is transaction id for the payment.		
CurrencyId	`setCurrencyId(currencyId)` Set only for transactions made through Payeezy.	Set only for transactions made through Payeezy.	Set only for transactions made through Payeezy.
Gateway	`setGateway(gateway)` Required. Contains the ID of the Salesforce Billing payment gateway record used in the transaction.	Set by the Billing package.	Set by the Billing package.
ResponseValueByKey	Not set	Not set.	NA
Payment	`setPayment(paymentInstance)` Required to set when calling `refundTransaction`. Must query all fields from the payment instance except system fields.	Set by the Billing package when `RefundTransaction` is called. Must query all fields from the payment instance except system fields.	NA

Example:

```
global class TransactionParameter
{

    global Map<string, string> mapOfResponseValueByKey = new Map<string, string>();

    // Transaction Result
    global TransactionResult transactionResult = new TransactionResult();

    global void setTransactionUseCase(TransactionUseCase transactionUseCase) {}

    global TransactionUseCase getTransactionUseCase() {}

    global void setTransactionFrequencyType(TransactionFrequency
transactionFrequencyType) {}

    global TransactionFrequency getTransactionFrequencyType() {}

    global void setIsCredentialsOnFile(Boolean isCredentialsOnFile) {}

    global Boolean getIsCredentialsOnFile() {}

    //Set RequestingInvoiceId
```

```
    global void setRequestingInvoiceId(Id requestingInvoiceId) {}

    //Get RequestingInvoiceId
    global Id getRequestingInvoiceId() {}

    //Set cardCodeResponse
    global void setCardCodeResponse(String cardCodeResponse) {}

    // Get cardCodeResponse
    global String getCardCodeResponse() {}

    // Set requestingCreditCardNumber
    global void setCardNumber(String requestingCreditCardNumber) {}

    // Get requestingCreditCardNumber
    global String getCardNumber() {}

    // Set Payment
    global void setPayment(Payment__c paymentInstance) {}

    // Get Payment
    global Payment__c getPayment() {}

    // Get merchantRefId
    global String getMerchantId() {}

    //Set MerchantId
    global void setMerchantId(String merchantId) {}

    //set Gateway ID
    global void setGateWayId(String gatewayId) {}

    // Get Gateway ID
    global String getGateWayId() {}

    // Get ResponseValueByKey
    global Map<string, string> getResponseValueByKey() {}

    // Set Gateway
    global void setGateWay(PaymentGateway__c gateway) {}

    // Get Gateway
    global PaymentGateway__c getGateWay() {}

    // Set PaymentMethod
    global void setPaymentMethod(PaymentMethod__c paymentMethod) {}

    // Get PaymentMethod
    global PaymentMethod__c getPaymentMethod() {}

    // Set Invoice
    global void setInvoice(Invoice__c invoice) {}

    // Get Invoice
```

```
    global Invoice__c getInvoice() {}

    // Set listOfInvoiceLine
    global void setInvoiceLine(InvoiceLine__c line) {}

    // Get listOfInvoiceLine
    global List<InvoiceLine__c> getInvoiceLine() {}

    // Set Transaction
    global void setTransaction(PaymentTransaction__c transactionInstance) {}

    // Get Transaction
    global PaymentTransaction__c getTransaction() {}

    // Set Account
    global void setAccount(Account accountInstance) {}

    // Get Account
    global Account getAccount() {}

    // Set RequestBody
    global void setRequestBody(String requestBody) {}

    // Get RequestBody
    global String getRequestBody() {}

    // Set FirstName
    global void setFirstName(String firstName) {}

    // Get FirstName
    global String getFirstName() {}

    // Set LastName
    global void setLastName(String lastName) {}

    // Get LastName
    global String getLastName() {}

    // Set EmailId
    global void setEmailId(String emailId) {}

    // Get EmailId
    global String getEmailId() {}

    // Set Street
    global void setStreet(String street) {}

    // Get Street
    global String getStreet() {}

    // Set City
    global void setCity(String city) {}

    // Get City
```

```
    global String getCity() {}

    // Set State
    global void setState(String state) {}

    // Get State
    global String getState() {}

    // Set ZipCode
    global void setZipCode(String zipCode) {}

    // Get ZipCode
    global String getZipCode() {}

    // Set Country
    global void setCountry(String country) {}

    // Get Country
    global String getCountry() {}

    // Set Phone
    global void setPhone(String phone) {}

    // Get Phone
    global String getPhone() {}

    // Set Currency Id
    global void setCurrencyId(String currencyId) {}

    // Get Currency Id
    global String getCurrencyId() {}

    // Set Amount
    global void setAmount(String amount) {}

    // Get Amount
    global String getAmount() {}

}
```

Mapping Gateway Responses to Payment Transactions with TransactionResult

The `TransactionResult` class contains a list of information that the payment gateway sends in response to a transaction request. For charge transactions, your payment gateway package evaluates this information and maps it to a Payment Transaction record in Salesforce Billing.

For complete reference information on the `TransactionResult` class's methods and parameters, review TransactionResult Class.

`TransactionResult` methods map their output to specific fields on the payment transaction object.

Note: The payment gateway package doesn't create a payment transaction for tokenization and refund transactions.

Table 1: TransactionResult to Payment Transaction Field Mapping

Field Label	API Name	TransactionResult Method
Card Code Response	blng__CardCodeResponse__c	`setCardCodeResponse()`
Gateway Status	blng__GatewayStatus__c	`setGatewayStatus()`
Request-Type of Payment	blng__RequestTypeOfPayment__c	Internal
Response	blng__Response__c	Internal
Response Code	blng__ResponseCode__c	`setResponseCode()`
Response Gateway ID	blng__GatewayID__c	`setGatewayId()`
Response Message	blng__ResponseMessage__c	`setResponseMessage()`
Response Status	blng__ResponseStatus__c	Internal
Status	blng__Status__c	Internal
Type	blng__Type__c	Charge, Refund, Void

Integrating a Payment Gateway Package

Configure Salesforce Billing and your payment gateway package to communicate with an external payment gateway.

Configure Payment Gateways in Salesforce

Install a payment gateway package and configure your payment gateway entities to communicate with your payment gateway provider.

Salesforce Billing Payment Gateway Interfaces

A payment gateway package requires three interfaces to communicate with Salesforce Billing. Each interface contains methods that the user-defined PaymentGatewayAPI class implements to process different types of transaction actions.

Configuring Salesforce Billing to Access Your PaymentGatewayAPI Class

To integrate with a payment gateway package, Salesforce Billing must know which gateway package class contains the methods required to process a payment request and convert the results to data that Salesforce Billing can read. The Salesforce Billing custom setting Payment Gateway Config defines the class that Salesforce Billing calls from the gateway package. You must also add code that installs a record of the class in PaymentGatewayConfig.

Mapping Gateway Response Codes to Salesforce Billing Gateway Statuses

Payment gateways have many response codes for payment operations. Salesforce Billing categorizes these codes into one of seven Gateway Status field values on your payment transaction record. The gateway status lets you know the result of a gateway call and whether you must correct anything. When you configure a payment gateway package, create metadata that maps your chosen gateway's response codes to one of the statuses.

Capture Externally Authorized Payments

Authorize payments in an external system, then capture the payment in Salesforce Billing using our Capture Transaction API methods. You can then use Salesforce Billing to manage the payment lifecycle and settle invoices.

Merchant Credential Fields

Each payment gateway package installs merchant credential fields on the Salesforce Billing payment gateway object. You can use these fields to provide merchant information when making a transaction request to the payment gateway.

Required Gateway Classes

All payment gateway and payment method records require several fields to communicate with the payment gateway, regardless of the gateway type. Specific gateway types may also require additional fields.

Configure a Payment Gateway in Salesforce Billing

Configure Salesforce Billing to send data to your chosen payment gateway.

GitHub Repository for Sample Payment Gateway Implementation

We've created a GitHub repository containing code samples for a sample Cybersource payment gateway implementation. Review the sample code if you need help with configuring your payment gateway implementation.

Configure Payment Gateways in Salesforce

Install a payment gateway package and configure your payment gateway entities to communicate with your payment gateway provider.

Important: If a custom payment gateway doesn't send the paymentGatewayToken, Salesforce Billing payment gateway API submits an extra tokenization call, which creates a new payment method.

1. Install your payment gateway package.

   Salesforce Billing provides payment gateway packages for AuthorizeDotNet, Cybersource, Payeezy, and PayFlowPro. You can also use packages for other gateways.

2. In Salesforce Billing, create a payment gateway record and set its Gateway Type field to your payment gateway. If your gateway name doesn't appear as a Gateway Type value, add it to the picklist.
3. Review the merchant credential fields on your payment gateway and enter the necessary merchant information.

   Your payment gateway record requires merchant credential fields. It sends the values of these fields as API to the payment gateway. Each payment gateway package installs different fields based on the needs of the payment gateway provider. For more information, review Merchant Credential Fields.

4. Check the payment gateway's Active field.
5. Check the payment gateway's Default field.

   The payment gateway's Default checkbox indicates that Salesforce Billing uses the gateway's merchant credentials if an ACH or credit card payment transaction doesn't provide stored credentials. Otherwise, the transaction fails. A payment gateway's Active and Default fields must be selected for the gateway entity to pass its information to the payment gateway provider.

Salesforce Billing Payment Gateway Interfaces

A payment gateway package requires three interfaces to communicate with Salesforce Billing. Each interface contains methods that the user-defined PaymentGatewayAPI class implements to process different types of transaction actions.

Creating a PaymentGateway Interface

The PaymentGateway interface contains several basic transaction methods that the `PaymentGatewayAPI` class implements to handle gateway requests. The methods allow PaymentGatewayAPI to create a credit card token, process a payment charge, void a payment, authorize a payment, and refund a payment.

Creating a PaymentGateways Interface

The `PaymentGateways` interface uses the `processPayments` method, which evaluates the payment transaction's type to perform a payment action.

Creating a PaymentGatewayStatus Interface

Salesforce Billing implements the PaymentGatewayStatus interface to populate the `TransactionResult`'s `GatewayStatus` enum with the value of the return code that the external gateway set in `transactionResult`.

Creating a PaymentGateway Interface

The PaymentGateway interface contains several basic transaction methods that the `PaymentGatewayAPI` class implements to handle gateway requests. The methods allow PaymentGatewayAPI to create a credit card token, process a payment charge, void a payment, authorize a payment, and refund a payment.

Create a credit card token

```
Map<String, TransactionResult> generateToken(Map<String, TransactionParameter>
mapOfTransactionParameterById,PaymentGatewayParameter paymentGatewayParameter);
```

Process a payment charge

```
Map<String, TransactionResult> chargeTransaction(Map<String, TransactionParameter>
mapOfTransactionParameterById,PaymentGatewayParameter paymentGatewayParameter);
```

Void a payment

```
Map<String, TransactionResult> voidTransaction(Map<String, TransactionParameter>
mapOfTransactionParameterById,PaymentGatewayParameter paymentGatewayParameter);
```

Authorize a payment

```
Map<String, TransactionResult> authorizeTransaction(Map<String, TransactionParameter>
mapOfTransactionParameterById,PaymentGatewayParameter paymentGatewayParameter);
```

Refund a payment

```
Map<String, TransactionResult> refundTransaction(Map<String, TransactionParameter>
mapOfTransactionParameterById,PaymentGatewayParameter paymentGatewayParameter);
```

So, if you want your gateway package to process a `chargeTransaction` sent from Salesforce Billing, the package must implement `chargeTransaction` within a `PaymentGatewayAPI` class. Let's see how this configuration looks in a sample CyberSource implementation.

Example:

```
global class CyberSourceAPI implements blng.PaymentGateWay
{
    global static Map<String, blng.TransactionResult> generateToken(Map<String,
blng.TransactionParameter> mapOfTransactionParameterById)
    {
        return service().generateToken(mapOfTransactionParameterById);
    }

   global static Map<String, blng.TransactionResult> authorizeTransaction(Map<String,
 blng.TransactionParameter> mapOfTransactionParameterById)
    {
        return service().authorizeTransaction(mapOfTransactionParameterById);
    }

    global static Map<String, blng.TransactionResult> chargeTransaction(Map<String,
blng.TransactionParameter> mapOfTransactionParameterById)
    {
        return service().chargeTransaction(mapOfTransactionParameterById);
    }
```

```
    global static Map<String, blng.TransactionResult> voidTransaction(Map<String,
blng.TransactionParameter> mapOfTransactionParameterById)
    {
        return service().voidTransaction(mapOfTransactionParameterById);
    }

    global static Map<String, blng.TransactionResult> refundTransaction(Map<String,
blng.TransactionParameter> mapOfTransactionParameterById)
    {
        return service().refundTransaction(mapOfTransactionParameterById);
    }
}
```

Creating a PaymentGateways Interface

The `PaymentGateways` interface uses the `processPayments` method, which evaluates the payment transaction's type to perform a payment action.

The `PaymentGateways` interface calls the `processPayments` method, which evaluates the TransactionParameter object.

```
global interface PaymentGateways
{
    Map<String, TransactionResult> processPayments(Map<String, TransactionParameter>
mapOfTransactionParameterById,PaymentGatewayParameter paymentGatewayParameter);
}
```

The `processPayments` method can perform different actions based on the `TransactionType` value of the TransactionParameter's `paymentGatewayParameter`. When you configure your `PaymentGateways` interface, define all the `TransactionType` string variables that `processPayments` can accept. Then, add your logic for identifying the transaction type and calling the appropriate method. We've provided a sample here for a CyberSource payment gateway implementation.

Example:

```
global class CyberSourceAPI implements blng.PaymentGateWay, blng.PaymentGateWays,
blng.PaymentGateWayStatus
{
    // ============================================================================
    // CONSTANT
    // ============================================================================

    // ============================================================================
    // STATIC VARIABLES
    // ============================================================================

    // Attribute to implement singleton pattern for CyberSource class
    private static CyberSource CyberSourceInstance;

    // ============================================================================
    // VARIABLES
    // ============================================================================
    //Billing sends following strings for the transactions
    private static final String GENERATE_TOKEN = 'generateToken';
    private static final String AUTHORIZE_TRANSACTION = 'authorizeTransaction';
    private static final String CHARGE_TRANSACTION = 'chargeTransaction';
```

```
    private static final String VOID_TRANSACTION = 'voidTransaction';
    private static final String REFUND_TRANSACTION = 'refundTransaction';
    private static final String CAPTURE_TRANSACTION = 'captureTransaction';
    private static final String VOID_REFUND_TRANSACTION = 'voidRefundTransaction';
    private static final String VOID_TOKEN_TRANSACTION = 'voidTokenTransaction';
    private static final String GET_PAYMENT_STATUS = 'getPaymentStatus';
    private static final String GET_REFUND_STATUS = 'getRefundStatus';
    private static final String NON_REFERRED_REFUND = 'nonReferredRefund';
    private static final String PROCESS_PAYMENTS = 'processPayments';
    // ================================================================================
    // Methods
    // ================================================================================

    global static Map<String, blng.TransactionResult> processPayments(Map<String,
blng.TransactionParameter> mapOfTransactionParameterById, blng.PaymentGatewayParameter
 paymentGatewayParameter)
    {
        if(paymentGatewayParameter.getTransactionType() == GENERATE_TOKEN)
        {
            return generateToken(mapOfTransactionParameterById);
        }
       else if(paymentGatewayParameter.getTransactionType() == AUTHORIZE_TRANSACTION)

        {
            return authorizeTransaction(mapOfTransactionParameterById);
        }
        else if(paymentGatewayParameter.getTransactionType() == CHARGE_TRANSACTION)
        {
            return chargeTransaction(mapOfTransactionParameterById);
        }
        else if(paymentGatewayParameter.getTransactionType() == VOID_TRANSACTION)
        {
            return voidTransaction(mapOfTransactionParameterById);
        }
        else if(paymentGatewayParameter.getTransactionType() == REFUND_TRANSACTION)
        {
            return refundTransaction(mapOfTransactionParameterById);
        }
        else if(paymentGatewayParameter.getTransactionType() == CAPTURE_TRANSACTION)
        {
            return captureTransaction(mapOfTransactionParameterById);
        }
       else if(paymentGatewayParameter.getTransactionType() == VOID_REFUND_TRANSACTION)

        {
            return voidRefundTransaction(mapOfTransactionParameterById);
        }
       else if(paymentGatewayParameter.getTransactionType() == VOID_TOKEN_TRANSACTION)

        {
            return voidTokenTransaction(mapOfTransactionParameterById);
        }
        else if(paymentGatewayParameter.getTransactionType() == GET_PAYMENT_STATUS)
        {
```

```
            return getPaymentStatus(mapOfTransactionParameterById);
        }
        else if(paymentGatewayParameter.getTransactionType() == GET_REFUND_STATUS)
        {
            return getRefundStatus(mapOfTransactionParameterById);
        }
        else if(paymentGatewayParameter.getTransactionType() == NON_REFERRED_REFUND)
        {
            return nonReferredRefund(mapOfTransactionParameterById);
        }

        return NULL;
    }
}
```

Creating a PaymentGatewayStatus Interface

Salesforce Billing implements the PaymentGatewayStatus interface to populate the `TransactionResult`'s `GatewayStatus` enum with the value of the return code that the external gateway set in `transactionResult`.

```
global interface PaymentGateWayStatus
{
    void populateGatewayStatus(TransactionResult transactionResult);
}
```

So, if you want your gateway package to process a `chargeTransaction` sent from Salesforce Billing, the package must implement `chargeTransaction` within a `PaymentGatewayAPI` class. Let's see how this configuration looks in a sample CyberSource implementation.

```
global class CyberSourceAPI implements blng.PaymentGateWay, blng.PaymentGateWays,
blng.PaymentGateWayStatus
{
    /**
    Implement other methods as described above
    */
    private static CyberSource service()
    {
        if (NULL == CyberSourceInstance)
        {
            CyberSourceInstance = CyberSource.getInstance();
        }
        return CyberSourceInstance;
    }
    /**
    * @name populateGatewayStatus
    * @description Method that populates the GatewayStatus enum on the TransactionResult,

    * given return codes that are already set in the TransactionResults
    * @param transactionResult
    */
    global static void populateGatewayStatus(blng.TransactionResult transactionResult)
    {
        service().populateGatewayStatus(transactionResult);
```

```
    }
}
```

Configuring Salesforce Billing to Access Your PaymentGatewayAPI Class

To integrate with a payment gateway package, Salesforce Billing must know which gateway package class contains the methods required to process a payment request and convert the results to data that Salesforce Billing can read. The Salesforce Billing custom setting Payment Gateway Config defines the class that Salesforce Billing calls from the gateway package. You must also add code that installs a record of the class in PaymentGatewayConfig.

First, create your primary gateway package class. For ease of reference, name it after your payment gateway, followed by "API." The class must implement all three of our payment gateway interfaces. Here's an example for CyberSource.

```
global class CyberSourceAPI implements blng.PaymentGateWay, blng.PaymentGateWays,
blng.PaymentGateWayStatus
{}
```

Next, add code that lets you install a record of the class in PaymentGatewayConfig. For this example, we've titled our method `insertPaymentGatewayConfigCustomSettings`.

```
public void insertPaymentGatewayConfigCustomSettings()
{
    String gatewayName = 'CyberSource';
    List<blng__PaymentGatewayConfig__c> listofConfiguration = new
List<blng__PaymentGatewayConfig__c>();
    blng__PaymentGatewayConfig__c gateway = new blng__PaymentGatewayConfig__c(
            Name = gatewayName,
            blng__ClassName__c = CyberSource.class.getName().substringBefore(gatewayName)
 + gatewayName + 'API');
    listofConfiguration.add(gateway);

    // get the all custom settings to check whether system has already settings or not
    Map<String, blng__PaymentGatewayConfig__c> mapOfConfigurationValues =
blng__PaymentGatewayConfig__c.getAll();
    // if system does not have the gateway, then insert it
    if(!mapOfConfigurationValues.containskey(gatewayName)) {
        insert listofConfiguration;
    }
}
```

Mapping Gateway Response Codes to Salesforce Billing Gateway Statuses

Payment gateways have many response codes for payment operations. Salesforce Billing categorizes these codes into one of seven Gateway Status field values on your payment transaction record. The gateway status lets you know the result of a gateway call and whether you must correct anything. When you configure a payment gateway package, create metadata that maps your chosen gateway's response codes to one of the statuses.

First, let's look at the values for the payment transaction's Status field in Salesforce Billing. When your gateway integration sets Status to a value, Salesforce Billing attempts to perform the action listed in the Salesforce Billing Action column in the following table.

Response	Definition	Salesforce Billing Action
Success	The gateway call succeeded.	Attempt payment creation and allocation.

Response	Definition	Salesforce Billing Action
Decline	The gateway call failed, but it can still work if you try again. For example, the customer had insufficient funds or briefly lost their connection to the internet. This response is also known as a "soft decline."	Update the payment run's decline count. Create payment transaction with gateway status of Decline.
Validation Error	Customer payment data is incorrect, such a misspelling in the credit card address or incorrect CVV.	Create a payment transaction with a gateway status of Validation Error.
Permanent Fail	The gateway call failed and won't work even if you try again. If you receive this response, don't make further payments using the related payment method.	Create a payment transaction with a gateway status of Permanent Fail.
Requires Review	The customer bank requires additional information before completing the payment.	Create a payment transaction with gateway status of Requires Review.
Indeterminate	The gateway didn't respond to the call. This response usually happens when Salesforce Billing times out waiting for the response. This response doesn't increase the payment run's failure count.	Lock the invoice from further payment runs. Create a payment transaction with a gateway status of Indeterminate. Allow users to autovoid or manually create a payment.
System Fail	Salesforce Billing ended the payment request call before receiving a response. For example, Salesforce Billing lost credentials or lost access to its server. This response increases the payment run's failure count. Salesforce Billing ends payment calls if it doesn't receive a response from the gateway in two minutes.	Create a payment transaction with a gateway status of System Fail. Allow users to autovoid or manually create a payment.

Response codes and descriptions vary between payment gateway providers. Let's look at a few examples and see how they can map to a gateway status.

Gateway	Code	Gateway's Description	Salesforce Billing Equivalent
Payeezy	100	"Approved"	Success
CyberSource	151	"The request was received, but a server timeout occurred. This error doesn't include timeouts between the client and the server." "To avoid duplicating the transaction, don't resend the	Indeterminate

Gateway	Code	Gateway's Description	Salesforce Billing Equivalent
		request until you have reviewed the transaction status at the Business Center. See the documentation for your CyberSource client for more information."	
PayFlowPro	4	"Invalid amount format. Use the format '#####.##' Don't include currency symbols or commas."	System
AuthorizeDotNet	E00057	"The user doesn't have permissions to submit requests from a mobile device."	Fail

Configure Response Code Metadata in Your Payment Gateway Package

Payment gateway packages require a custom metadata type file to communicate with Salesforce Billing. The custom metadata type defines the custom object and fields that the gateway package uses to store request and response information sent from the payment gateway. Your custom metadata type file evaluates the value of your gateway package's custom gateway field. Then, the file must convert the gateway field to a value that Salesforce Billing sends to the payment transaction's Status field.

Configuring Salesforce Billing to Read Your Mapping Metadata

After you've set up your mapping metadata, configure your payment gateway adapter Apex class to read the gateway status and assign it to `TransactionResult`.

Configure Response Code Metadata in Your Payment Gateway Package

Payment gateway packages require a custom metadata type file to communicate with Salesforce Billing. The custom metadata type defines the custom object and fields that the gateway package uses to store request and response information sent from the payment gateway. Your custom metadata type file evaluates the value of your gateway package's custom gateway field. Then, the file must convert the gateway field to a value that Salesforce Billing sends to the payment transaction's Status field.

The fields used to store gateway request and response values vary between payment gateway packages. Let's look at a few examples from the custom metadata created for the AuthorizeDotNet, CyberSource, and Payeezy packages. In each, we provide a recommended metadata name.

Package	Metadata Name	API Names	Field Labels
AuthorizeDotNet	`adnblng__GatewayStatusMapping__mdt`	• Response from Gateway • Additional Gateway Response • Corresponding Gateway Status	• `adnblng_ResponseCode__c` • `adnblng_ResponseCode2__c` • `adnblng__GatewayStatus__c`
CyberSource	`csblng__GatewayStatusMapping__mdt`	• Response from Gateway	• `csblng_ResponseCode__c`

Package	Metadata Name	API Names	Field Labels
		• Corresponding Gateway Status	• csblng__GatewayStatus__c
Payeezy	pyzblng__GatewayStatusMapping__mdt	• Response from Gateway • Corresponding Gateway Status	• pyzblng__ResponseCode__c • pyzblng__GatewayStatus__c

Example: Cybersource's response code 100 represents a successful communication with the payment gateway. Let's look at a metadata file that evaluates the Cybersource gateway package field `ResponseCode__c` for a value of *100* and then sets the payment transaction's status field to Success.

```
<?xml version="1.0" encoding="UTF-8"?>
<CustomMetadata xmlns="http://soap.sforce.com/2006/04/metadata"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <label>100</label>
    <values>
        <field>ResponseCode__c</field>
        <value xsi:type="xsd:string">100</value>
    </values>
    <values>
        <field>GatewayStatus__c</field>
        <value xsi:type="xsd:string">Success</value>
    </values>
</CustomMetadata>
```

Example: Some payment gateways return multiple response codes for one gateway transaction. Responses with multiple codes can occur when the gateway processes the transaction successfully but then the customer card provider responds with another value. For example, the AuthorizeDotNet code *I00001* indicates that the payment gateway successfully processed a request. However, the card provider can then send a response code of *3*, which indicates that the transaction was declined. The provider can also send a response code of *7*, which indicates that the credit card expiration date is invalid. Let's look at metadata samples for processing *I00001 – 3* and *I00001 – 7* respectively.

In this case, we assign *I00001 – 3* to a Salesforce Billing value of Permanent Fail.

```
<?xml version="1.0" encoding="UTF-8"?>
<CustomMetadata xmlns="http://soap.sforce.com/2006/04/metadata"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<label>I00001-3</label>
     <values>
        <field>ResponseCode__c</field>
        <value xsi:type="xsd:string">I00001</value>
    </values>
    <values>
        <field>ResponseCode2__c</field>
```

```
        <value xsi:type="xsd:string">3</value>
    </values>
    <values>
        <field>GatewayStatus__c</field>
        <value xsi:type="xsd:string">PermanentFail</value>
    </values>
</CustomMetadata>
```

In this case, we assign *I00001 - 7* to a Salesforce Billing value of Decline.

```
<?xml version="1.0" encoding="UTF-8"?>
<CustomMetadata xmlns="http://soap.sforce.com/2006/04/metadata"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<label>I00001-2</label>
     <values>
        <field>ResponseCode__c</field>
        <value xsi:type="xsd:string">I00001</value>
    </values>
    <values>
        <field>ResponseCode2__c</field>
        <value xsi:type="xsd:string">2</value>
    </values>
    <values>
        <field>GatewayStatus__c</field>
        <value xsi:type="xsd:string">Decline</value>
    </values>
</CustomMetadata>
```

Configuring Salesforce Billing to Read Your Mapping Metadata

After you've set up your mapping metadata, configure your payment gateway adapter Apex class to read the gateway status and assign it to `TransactionResult`.

First, define a `GatewayStatusType` enum.

```
global Enum GatewayStatusType { Success,
                                Decline,
                                ValidationError,
                                PermanentFail,
                                RequiresReview,
                                Indeterminate,
                                SystemError }

// Set gatewayStatus
global void setGatewayStatus(GatewayStatusType gatewayStatus)
{
    this.gatewayStatus = gatewayStatus;
}

// Get gatewayStatus
global GatewayStatusType getGatewayStatus()
{
```

```
    return gatewayStatus;
}
```

Next, add code that reads your GatewayStatusMapping metadata and sets the correct gateway status in your TransactionResult object. Here's one way you can configure it.

```
private static Map<string, blng.TransactionResult.GatewayStatusType>
mapGatewayStatusEnumTypesByStrings =
                new Map<string, blng.TransactionResult.GatewayStatusType>();
static {
    List<blng.TransactionResult.GatewayStatusType> enumValues =
blng.TransactionResult.GatewayStatusType.values();
    for (Integer i = 0; i < enumValues.size(); i++) {
       mapGatewayStatusEnumTypesByStrings.put(enumValues.get(i).name(), enumValues.get(i));

    }
}

// default Gateway Status - It should be always Indeterminate to avoid duplicate payments
private static final blng.TransactionResult.GatewayStatusType defaultGatewayStatus =
blng.TransactionResult.GatewayStatusType.Indeterminate;

/**
* Does a SOQL lookup on the mapper table and gets the gateway status mapped to the return
 code
* Returns the default enum if no match is found
* @param transactionResult
*/
public void populateGatewayStatus(blng.TransactionResult transactionResult) {
    blng.TransactionResult.GatewayStatusType gatewayStatus = defaultGatewayStatus;
    String returnCode = transactionResult.getResponseCode();
    if (returnCode != null) {
        List<GatewayStatusMapping__mdt> gatewayStatusMaps = [
                SELECT GatewayStatus__c
                FROM GatewayStatusMapping__mdt
                WHERE ResponseCode__c = :returnCode
                LIMIT 1
        ];
        if (!gatewayStatusMaps.isEmpty()) {
            gatewayStatus =
mapGatewayStatusEnumTypesByStrings.get(gatewayStatusMaps.get(0).GatewayStatus__c);
            if (gatewayStatus == null) {
                gatewayStatus = defaultGatewayStatus;
            }
        }
    }
    transactionResult.setGatewayStatus(gatewayStatus);
}
```

Capture Externally Authorized Payments

Authorize payments in an external system, then capture the payment in Salesforce Billing using our Capture Transaction API methods. You can then use Salesforce Billing to manage the payment lifecycle and settle invoices.

Delayed capture is useful when the payment is authorized at the time of sale and settled after an event occurs. For example, you can capture a payment after a subscription is provisioned or an item is delivered.

EDITIONS

Available in: Salesforce Billing Winter '21 and later

Important: Our Capture Payment API works out of the box with the Salesforce Billing Cybersource gateway integration package. Customers using other payment gateways must work with their payment gateway vendors to extend the gateway integration packages to the Capture Payment API.

Managing Payment Authorizations

Salesforce Billing doesn't process payment authorizations, but you can store information from an external authorization in a Salesforce Billing payment authorization record. To store external authorization information in Salesforce, add information from the external authorization to the following fields in the Salesforce Billing payment authorization record:

- `Account ID`
- `Original Amount`
- `Currency ISO Code` (for multi-currency orgs only)
- `Payment Gateway ID`
  - Use the ID of the Salesforce Billing payment gateway record that you use for storing payment capture information. The payment gateway record must have the same merchant credentials used to process the external payment authorization.
- `Gateway Reference Number`
  - The payment gateway supplies the reference number.
- `Status` this field must have the value `Processed`

Calling the Capture Transaction API

After you've created a payment authorization, you can pass it to the Capture Transaction API. To call the Capture Transaction API, add the following method to your `PaymentGateway` Apex class.

```
List<blng.OutputResult> result = blng.TransactionAPI.captureTransaction(lcip);
```

The CaptureTransaction API accepts information through the `CaptureInputParameter` class.

Table 2: CaptureInputParameter Parameters

Parameter	Description	Type	Required
`PaymentAuthorization`	ID of the payment authorization record from Salesforce Billing.	ID	Yes
`captureAmount`	Amount of payment to capture from the payment.	Decimal	Yes
`additionalParameters`	If your capture request requires more information, you can provide a map of additional	Map<String, String>	No

Parameter	Description	Type	Required
	key-value pairs to the gateway adapter		

The `CaptureOutputResult` class returns the results of your capture request. If you successfully capture the payment, Salesforce Billing creates a posted payment record related to your payment authorization. The CaptureTransaction API also sends a capture success message. If the capture fails, the CaptureTransaction API sends a capture failure message.

Table 3: CaptureOutputResult Parameters

Parameter	Description
`isSuccess`	Shows whether the payment capture request succeeded.
`paymentID`	The Salesforce Billing payment record that the CaptureAPI created as a result of successfully capturing a payment from the gateway.
`paymentTransactionID`	The Salesforce Billing payment transaction record containing information about the gateway interaction for the capture request.
`errorMessage`	If the payment capture was unsuccessful, contains information about the error.

Permissions

To perform a capture transaction, users require access to the following fields.

Payment

Write access on the following fields:

- `Account`
- `Amount`
- `Status`
- `Payment Authorization`
- `Payment Gateway`
- `Payment Transaction`
- `Payment Gateway ID`

Payment Transaction

Write access on the following fields:

- `Account`
- `Amount`
- `Type`
- `Source Transaction ID`
- `Gateway Request`
- `Gateway Response`
- `Payment Gateway`
- `Gateway Status`

- `Response`
- `Response Code`
- `Response Message`
- `Payment Gateway ID`
- `Gateway Date`
- `Status`

Referenced Classes

- `CaptureInputParameter` on page 28
- `InputParameter` on page 33
- `CaptureOutputResult` on page 30
- `OutputResult` on page 34

Example:

Request

In this example, `CaptureInputParameter` receives the payment authorization record from Salesforce Billing, a payment capture amount of 4.23, and additional parameters. The parameters are stored in a list and sent to `TransactionAPI`, which performs the payment capture. `CaptureOutputResult` contains information about the capture.

```
String authId = 'a1wxxxxxxxxxxxxxx';
Map<String, Schema.SObjectField> fieldMap =
blng__PaymentAuthorization__c.sObjectType.getDescribe().fields.getMap();
Set<String> fieldNames = fieldMap.keySet();
SObject myrecord = Database.query('select ' +
String.join((Iterable<String>)fieldNames, ',') + ' from blng__PaymentAuthorization__c
 Where Id = \''+authId+'\'');
Map<String,String> additionalParams = new Map<String,String>();
additionalParams.put('merchantReferenceCode','demoCapture');

blng.CaptureInputParameter cip = new blng.CaptureInputParameter();
cip.setPaymentAuthorization((blng__PaymentAuthorization__c)myrecord);
cip.setAmount(4.23);
cip.setAdditionalParameters(additionalParams);
List<blng.CaptureInputParameter> lcip = new List<blng.CaptureInputParameter>();
lcip.add(cip);
List<blng.CaptureOutputResult> result = blng.TransactionAPI.captureTransaction(lcip);
system.debug(result[0]);
```

Success Response

Following a successful payment capture, `CaptureOutputResult` returns a success response with the IDs of the payment transaction and payment created in Salesforce Billing.

```
[CaptureOutputResult.errorMessage=null, CaptureOutputResult.isSuccess=true,
paymentId=a25xxxxxxxxxxxxxx, paymentTransactionId=a24xxxxxxxxxxxxxx]
```

Failure Response

Following a failed payment capture, `CaptureOutputResult` returns a failure response with an error message and the ID of the payment transaction created in Salesforce Billing. The payment transaction contains information about the failed gateway communication.

```
[CaptureOutputResult.errorMessage=Invalid data. c:authRequestID,
CaptureOutputResult.isSuccess=false, paymentId=null,
paymentTransactionId=a24xxxxxxxxxxxxx]
```

Merchant Credential Fields

Each payment gateway package installs merchant credential fields on the Salesforce Billing payment gateway object. You can use these fields to provide merchant information when making a transaction request to the payment gateway.

Table 4: AuthorizeDotNet Merchant Credential Fields

Attribute Name	API Name	Required	Type
APILoginId	adnblng__APILoginId__c	No	Text
APITransactionKey	adnblng__APITransactionKey__c	No	Text
TestMode	adnblng__TestMode__c	No	Checkbox

Table 5: CyberSource Merchant Credential Fields

Attribute Name	API Name	Required	Type
Merchant ID	csblng__MerchantId__c	No	Text
Merchant Reference	csblng__MerchantReference__c	No	Text
Test Mode	csblng__TestMode__c	No	Checkbox
Transaction Security Key	csblng__TransactionSecurityKey__c	No	Text

Table 6: Payeezy Merchant Credential Fields

Attribute Name	API Name	Required	Type
APIKey	pzyblng__APIKey__c	No	Text
APILoginId	pyzblng__APILoginId__c	No	Text
APISecret	pyzblng__APISecret__c	No	Text
APITransactionKey	pyzblng__APITransactionKey__c	No	Text
Enable User Identification	pyzblng__EnableUserIdentification__c	No	Checkbox
JSSecurityKey	pyzblng__JSSecurityKey__c	No	Text
TestMode	pyzblng__TestMode__c	No	Text
Token	pyzblng__Token__c	No	Text

Attribute Name	API Name	Required	Type
TransarmorToken	pyzblng__TransarmorToken__c	No	Text

Required Gateway Classes

All payment gateway and payment method records require several fields to communicate with the payment gateway, regardless of the gateway type. Specific gateway types may also require additional fields.

EDITIONS

Available in: Salesforce Billing 7.0 and later

Payment Method Fields

Label	API Name
Payment Method Name	Name
Account	blng__Account__c
Active	blng__Active__c
Auto Pay	blng__AutoPay__c
Bank Account Name	blng__BankAccountName__c
Bank Account Number	blng__BankAccountNumber__c
Bank Account Type	blng__BankAccountType__c
Bank Name	blng__BankName__c
Bank Routing Code	blng__BankRoutingCode__c
Card BIN	blng__CardBIN__c
Card CVV	blng__CVV__c
Card Expiration Month	blng__CardExpirationMonth__c
Card Expiration Year	blng__CardExpirationYear__c
Card Last 4	blng__CardLastFour__c
Card Number	blng__CardNumber__c
Card Type	blng__CardType__c
City	blng__ChequeNumber__c
Company	blng__BillingCity__c
Country	blng__BillingCompany__c
Email	blng__BillingEmail__c
Fax	blng__BillingFax__c
First Name	blng__BillingFirstName__c

Gateway Response	blng__GatewayResponse__c
Last Name	blng__BillingLastName__c
Name on Card	blng__Nameoncard__c
Nick Name	blng__NickName__c
Payment Gateway	blng__PaymentGateway__c
Payment Gateway Token	blng__PaymentGatewayToken__c
Payment Type	blng__PaymentType__c
Phone Number	blng__BillingPhone__c
Postal Code	blng__BillingZipPostal__c
State	blng__BillingStateProvince__c
Street Address 1	blng__BillingStreet__c
Street Address 2	blng__StreetAddress2__c

Payment Gateway Fields

Gateway integrations require the following payment gateway record fields regardless of gateway type. If you have a field required by a specific gateway such as Authorize.Net, make sure to add that field to your payment gateway record as well.

Label	API Name
Payment Gateway Name	Name
Active	blng__Active__c
Default	blng__Default__c
Gateway Type	blng__GatewayType__c

Configure a Payment Gateway in Salesforce Billing

Configure Salesforce Billing to send data to your chosen payment gateway.

EDITIONS

Available in: Salesforce Billing 7.0 and later

1. Set up remote site settings.
   a. From Setup, enter `Remote Site Settings`, and then select **Remote Site Settings**.
   b. Click **New Remote Site**.
   c. Add the domain production URL and domain sandbox URL for your gateway.

      We provide a list of URLs for our default-supported gateways, but you can add others based on your custom gateway's requirements.

Gateway	Production URL	Sandbox URL
Authorize.net	https://api.authorize.net	https://apitest.authorize.net

Gateway	Production URL	Sandbox URL
CyberSource	https://ics2wsa.ic3.com	https://icswstesta.ic3.com
Payeezy	https://api.payeezy.com	https://api-cert.payeezy.com

2. Set up custom settings.
   a. From Setup, enter `Custom Settings`, and then select **Custom Settings**.
   b. Click **Payment Gateway Config**, then click **Manage**.
   c. Click **New**.
   d. Enter a name for your payment gateway.
      Match this value to the name on your payment gateway record's Gateway Type field.
   e. Enter your gateway class name.
      This value takes the form of `PackagePrefix.APIClassName`. We provide a list of class names for our default gateways, but you can add others based on your custom gateway's requirements.

Name	Gateway Class Name
AuthorizeDotNet	AuthorizeDotNetAPI
CyberSource	CyberSourceAPI
Payeezy	PayeezyAPI

3. Save your changes.

GitHub Repository for Sample Payment Gateway Implementation

We've created a GitHub repository containing code samples for a sample Cybersource payment gateway implementation. Review the sample code if you need help with configuring your payment gateway implementation.

Salesforce Billing Payment Gateway Reference Implementation for Cybersource

Payment Gateway Class Reference

Review the classes and methods available when working with payment gateways in Salesforce Billing.

CaptureInputParameter Class

Receives the payment authorization record, payment capture amount, and an optional list of additional parameters from Salesforce Billing.

CaptureOutputResult Class

Stores the results of the payment capture request.

Payment Gateway Core Classes

Your payment gateway references these classes to receive transaction information from Salesforce Billing. They can't be modified.

Custom Gateway Classes

Salesforce Billing provides three classes that users can customize to integrate with their chosen payment gateway.

CaptureInputParameter Class

Receives the payment authorization record, payment capture amount, and an optional list of additional parameters from Salesforce Billing.

Namespace

blng

Example

In this example, t `CaptureInputParameter` receives the payment authorization record from Salesforce Billing, a payment capture amount of 4.23, and additional parameters. The parameters are stored in a list and sent to the `TransactionAPI` class, which performs the payment capture. The `CaptureOutputResult` class contains the result of the capture.

```
blng.CaptureInputParameter cip = new blng.CaptureInputParameter();
cip.setPaymentAuthorization((blng__PaymentAuthorization__c)myrecord);
cip.setAmount(4.23);
cip.setAdditionalParameters(additionalParams);
List<blng.CaptureInputParameter> lcip = new List<blng.CaptureInputParameter>();
lcip.add(cip);
List<blng.CaptureOutputResult> result = blng.TransactionAPI.captureTransaction(lcip);
system.debug(result[0]);
```

CaptureInputParameter Methods

CaptureInputParameter Methods

The following are methods for `CaptureInputParameter`.

setPaymentAuthorization(paymentAuthorization)

Sets the ID of the payment authorization.

getPaymentAuthorization()

Returns the payment authorization record from Salesforce Billing.

setAmount(amount)

Sets the amount of the payment capture.

getAmount()

Returns the amount of the payment capture to be performed.

`setPaymentAuthorization(paymentAuthorization)`

Sets the ID of the payment authorization.

Signature

```
global static void setPaymentAuthorization(blng__PaymentAuthorization__c
paymentAuthorization)
```

Parameters

paymentAuthorization
Type: `blng__PaymentAuthorization__c`

The payment authorization record from Salesforce Billing.

Return Value

Type: void

getPaymentAuthorization()

Returns the payment authorization record from Salesforce Billing.

Signature

```
global static blng__PaymentAuthorization__c getPaymentAuthorization()
```

Return Value

Type: blng__PaymentAuthorization__c

setAmount(amount)

Sets the amount of the payment capture.

Signature

```
global static void setAmount(Decimal amount)
```

Parameters

amount
Type: Decimal

The amount of the payment capture. Can be none, part, or all of the payment.

Return Value

Type: void

getAmount()

Returns the amount of the payment capture to be performed.

Signature

```
global static Decimal getAmount()
```

Return Value

Type: Decimal

CaptureOutputResult Class

Stores the results of the payment capture request.

Namespace

blng

Example

Example:

Request

In this example, `CaptureInputParameter` receives the payment authorization record from Salesforce Billing, a payment capture amount of 4.23, and additional parameters. The parameters are stored in a list and sent to `TransactionAPI`, which performs the payment capture. `CaptureOutputResult` contains information about the capture.

```
String authId = 'a1wxxxxxxxxxxxxxx';
Map<String, Schema.SObjectField> fieldMap =
blng__PaymentAuthorization__c.sObjectType.getDescribe().fields.getMap();
Set<String> fieldNames = fieldMap.keySet();
SObject myrecord = Database.query('select ' +
String.join((Iterable<String>)fieldNames, ',') + ' from blng__PaymentAuthorization__c
 Where Id = \''+authId+'\'');
Map<String,String> additionalParams = new Map<String,String>();
additionalParams.put('merchantReferenceCode','demoCapture');

blng.CaptureInputParameter cip = new blng.CaptureInputParameter();
cip.setPaymentAuthorization((blng__PaymentAuthorization__c)myrecord);
cip.setAmount(4.23);
cip.setAdditionalParameters(additionalParams);
List<blng.CaptureInputParameter> lcip = new List<blng.CaptureInputParameter>();
lcip.add(cip);
List<blng.CaptureOutputResult> result = blng.TransactionAPI.captureTransaction(lcip);
system.debug(result[0]);
```

Success Response

Following a successful payment capture, `CaptureOutputResult` returns a success response with the IDs of the payment transaction and payment created in Salesforce Billing.

```
[CaptureOutputResult.errorMessage=null, CaptureOutputResult.isSuccess=true,
paymentId=a25xxxxxxxxxxxxxx, paymentTransactionId=a24xxxxxxxxxxxxxx]
```

Failure Response

Following a failed payment capture, `CaptureOutputResult` returns a failure response with an error message and the ID of the payment transaction created in Salesforce Billing. The payment transaction contains information about the failed gateway communication.

```
[CaptureOutputResult.errorMessage=Invalid data. c:authRequestID,
CaptureOutputResult.isSuccess=false, paymentId=null,
paymentTransactionId=a24xxxxxxxxxxxxx]
```

CaptureOutputResult Methods

CaptureOutputResult Methods

The following are methods for `CaptureOutputResult`.

`setPaymentId(paymentId)`

Sets the ID of the Salesforce Billing payment record created after a successful capture transaction.

Signature

```
global static void setPaymentId(Id paymentId)
```

Parameters

`paymentId`
Type: Id

ID of the payment record created in Salesforce Billing. The payment record stores information about the payment.

Return Value

Type: void

`getPaymentId()`

Returns the ID of the Salesforce Billing payment record that was created after a successful capture transaction.

Signature

```
global static Id getPaymentId()
```

Return Value

Type: Id

setPaymentTransactionId(paymentTransactionId)

Sets the ID of the Salesforce Billing payment transaction record that was created after a successful capture transaction.

Signature

```
public static void setPaymentTransactionId(Id paymentTransactionId)
```

Parameters

paymentTransactionId
Type: Id

ID of the payment transaction record created in Salesforce Billing. This record stores the results of the communication with the payment gateway.

Return Value

Type: void

getPaymentTransactionId()

Returns the ID of the Salesforce Billing payment transaction record created after a successful capture transaction.

Signature

```
global static Id getPaymentTransactionId()
```

Return Value

Type: Id

Payment Gateway Core Classes

Your payment gateway references these classes to receive transaction information from Salesforce Billing. They can't be modified.

The following are the `blng` namespace classes required for payment gateway integration.

InputParameter Class
Captures and retrieves additional parameters to be sent during the capture payment transaction.

OutputResult Class
Stores and shows the results of the capture payment request made to the payment gateway.

PaymentGateway Interface
Interface for standard payment transaction operations such as voids and refunds.

PaymentGateways Interface
Interface with common payment gateway operation across all gateways.

PaymentGatewayParameter Class
Global parameter class for the payment gateway.

PaymentGatewayStatus Interface
Interface to populate the TransactionResult value object from the gateway response.

TransactionAPI Class
The TransactionAPI class includes global methods for several payment features.

TransactionParameter Class
Acts as a container to store calculation-specific information.

TransactionResult Class
Holds the result of the transaction.

TransactionResult.GatewayStatusType Enum
An enum defining the appropriate gateway status values returned by the payment gateway.

InputParameter Class

Captures and retrieves additional parameters to be sent during the capture payment transaction.

Namespace

blng

InputParameter Methods

InputParameter Methods

The following are methods for `InputParameter`.

setAdditionalParameters(additionalParams)
Creates a map of any additional parameters that the user sent with the payment capture request. This method and its parameters are optional.

getAdditionalParameters()
Returns additional parameters sent during the capture payment request.

`setAdditionalParameters(additionalParams)`

Creates a map of any additional parameters that the user sent with the payment capture request. This method and its parameters are optional.

Signature

```
global static void setAdditionalParameters(Map<String,String> additionalParams)
```

Parameters

additionalParams
Type: Map<String,String>

Return Value

Type: void

getAdditionalParameters()

Returns additional parameters sent during the capture payment request.

Signature

```
global static Map<string, string> getAdditionalParameters()
```

Return Value

Type: Map<String, String>

OutputResult Class

Stores and shows the results of the capture payment request made to the payment gateway.

Namespace

blng

OutputResult Methods

OutputResult Methods

The following are methods for `OutputResult`.

setIsSuccess(isSuccess)
Sets the results of the capture request gateway communication as either `truetrue` (successful) or `falsefalse` (failed).

isSuccess(isSuccess)
Shows whether the capture payment request was successful in the payment gateway.

setErrorMessage(errorMessage)
Sets the error message following a failed payment capture request made to the payment gateway.

getErrorMessage()
Returns the error message that the payment gateway sent following a failed capture payment request.

setIsSuccess(isSuccess)

Sets the results of the capture request gateway communication as either `truetrue` (successful) or `falsefalse` (failed).

Signature

```
global static void setIsSuccess(boolean isSuccess)
```

Parameters

isSuccess
Type: boolean

Return Value

Type: Void

isSuccess(isSuccess)

Shows whether the capture payment request was successful in the payment gateway.

Signature

```
global static Boolean isSuccess
```

Return Value

Type: Boolean

setErrorMessage(errorMessage)

Sets the error message following a failed payment capture request made to the payment gateway.

Signature

```
global static void setErrorMessage(String errorMessage)
```

Parameters

errorMessage
Type: String

Error message containing information about the failed gateway communication. For information about the error message, see the documentation for your gateway.

Return Value

Type: void

getErrorMessage()

Returns the error message that the payment gateway sent following a failed capture payment request.

Signature

```
global static String getErrorMessage()
```

Return Value

Type: String

PaymentGateway Interface

Interface for standard payment transaction operations such as voids and refunds.

Namespace

blng

Usage

A gateway implementor uses logic such as `paymentGatewayParameter.getTransactionType() == "generateToken"` and calls underlying payment transaction functions. Similar logic applies for voids, refunds, and other actions.

PaymentGateway Methods

The following are methods for `PaymentGateway`.

`generateToken(mapOfTransactionParameterById)`

Generates a token for a payment method.

Signature

```
global Map<String, TransactionResult> generateToken(Map<String, TransactionParameter>
mapOfTransactionParameterById)
```

Parameters

`mapOfTransactionParameterById`
Type: Map<String, TransactionParameter>

Map of parameter is based on the payment gateway.

Return Value

Type: Map<String, TransactionResult>

List of Transaction Results : List<TransactionResult>

`chargeTransaction(mapOfTransactionParameterById)`

Authorize and capture a credit card payment and ACH payment.

Signature

```
global Map<String, TransactionResult> chargeTransaction(Map<String, TransactionParameter>
mapOfTransactionParameterById)
```

Parameters

`mapOfTransactionParameterById`
Type: Map<String, TransactionParameter>

Map of parameter is based on the payment gateway.

Return Value

Type: Map<String, TransactionResult>

List of Transaction Results : List<TransactionResult>

`voidTransaction(mapOfTransactionParameterById)`

The Void transaction type can cancel either an original transaction or a transaction that hasn't been settled.

Signature

```
global Map<String, TransactionResult> voidTransaction(Map<String, TransactionParameter>
mapOfTransactionParameterById)
```

Parameters

`mapOfTransactionParameterById`
Type: Map<String, TransactionParameter>

Parameter map is based on the payment gateway.

Return Value

Type: Map<String, TransactionResult>

List of Transaction Results : List<TransactionResult>

`authorizeTransaction(mapOfTransactionParameterById)`

Authorizes a credit card payment. To perform a charge transaction, the user must follow the authorization with a capture transaction.

Signature

```
global Map<String, TransactionResult> authorizeTransaction(Map<String,
TransactionParameter> mapOfTransactionParameterById)
```

Parameters

`mapOfTransactionParameterById`
Type: Map<String, TransactionParameter>

Map of parameter is based on the payment gateway.

Return Value

Type: Map<String, TransactionResult>

List of Transaction Results : List<TransactionResult>

`refundTransaction(mapOfTransactionParameterById)`

Refund a customer for a transaction that wsa successfully settled through the payment gateway.

Signature

```
public Map<String, TransactionResult> refundTransaction(Map<String, TransactionParameter>
mapOfTransactionParameterById)
```

Parameters

`mapOfTransactionParameterById`
Type: Map<String, TransactionParameter>

Map of parameter is based on the payment gateway.

Return Value

Type: Map<String, TransactionResult>

List of Transaction Results : List<TransactionResult>

PaymentGateway Example Implementation

The YourGatewayAPI class implements the PaymentGateway, PaymentGateways, and PaymentGatewayStatus interfaces.

PaymentGateways Interface

Interface with common payment gateway operation across all gateways.

Namespace

blng

Usage

Test.

PaymentGateways Methods

The following are methods for `PaymentGateways`.

`processPayments(mapOfTransactionParameterById, paymentGatewayParameter)`

Generates a token.

Signature

```
global Map<String, TransactionResult> processPayments(Map<String, TransactionParameter>
mapOfTransactionParameterById, PaymentGatewayParameter paymentGatewayParameter)
```

Parameters

`mapOfTransactionParameterById`
Type: Map<String, TransactionParameter>

`paymentGatewayParameter`
Type: PaymentGatewayParameter

Return Value

Type: Map<String, TransactionResult>

PaymentGateways Example Implementation

The YourGatewayAPI class implements the PaymentGateway, PaymentGateways, and PaymentGatewayStatus interfaces.

PaymentGatewayParameter Class

Global parameter class for the payment gateway.

Namespace

blng

PaymentGatewayParameter Methods

The following are methods for `PaymentGatewayParameter`.

`setTransactionType(transactionType)`

Sets TransactionType.

Signature

```
global void setTransactionType(String transactionType)
```

Parameters

`transactionType`
Type: String

Return Value

Type: Void

`getTransactionType()`

Gets TransactionType.

Signature

```
global String getTransactionType()
```

Return Value

Type: String

PaymentGatewayStatus Interface

Interface to populate the TransactionResult value object from the gateway response.

Namespace

blng

Usage

This method takes in the transaction result object and defines its attributes.

PaymentGatewayStatus Methods

The following are methods for `PaymentGatewayStatus`.

`PopulateGatewayStatus(transactionResult)`

Populates the GatewayStatus enum on the TransactionResult, given return codes that are already set in the TransactionResults.

Signature

```
global void PopulateGatewayStatus(TransactionResult transationResult)
```

Parameters

`transationResult`
Type: TransactionResult

Return Value

Type: Void

PaymentGatewayStatus Example Implementation

The YourGatewayAPI class implements the PaymentGateway, PaymentGateways, and PaymentGatewayStatus interfaces.

TransactionAPI Class

The TransactionAPI class includes global methods for several payment features.

Namespace

blng

TransactionAPI Methods

The following are methods for `TransactionAPI`.

generateToken(mapOfTransactionParameterById)
Generates a token for a payment method.

authorizeTransaction(mapOfTransactionParameterById)
Authorizes a credit card payment. To perform a charge transaction, the user must follow the authorization with a capture transaction.

captureTransaction(mapOfTransactionParameterById)
Captures a payment for an authorize transaction.

captureTransaction(captureInputParameters)
Captures a payment for an authorization transaction using input parameters.

chargeTransaction(mapOfTransactionParameterById)
Authorize and capture a credit card payment or ACH payment.

getPaymentStatus(mapOfTransactionParameterById)
Gets the status of a payment transaction.

voidTransaction(mapOfTransactionParameterById)
Cancels an original transaction that hasn't been settled.

refundTransaction(mapOfTransactionParameterById)
Refunds a customer for a transaction that has been succesfully settled through the payment gateway.

nonReferredRefund(mapOfTransactionParameterById)
Performs a non-referred refund of a credit card payment or an ACH payment.

voidRefundTransaction(mapOfTransactionParameterById)
Voids an unsettled refund transaction from a credit card payment or ACH payment.

getRefundStatus(mapOfTransactionParameterById)
Gets the status of a refund transaction.

voidTokenTransaction(mapOfTransactionParameterById)
Voids a credit card token.

refundPayment(listofRefundParameters)
Refunds a payment.

resetInvoiceCorrectiveAction(invoices)
Unlocks an invoice.

validateRefundTransactionId(transactionId)
Validates whether a payment transaction can be refunded.

authorizeTransaction(invoiceID)
Authorizes a payment for an invoice.

chargeTransaction(invoiceID)
Authorizes a payment for an invoice.

`generateToken(mapOfTransactionParameterById)`

Generates a token for a payment method.

Signature

```
global static List<TransactionResult> generateToken(Map<String, TransactionParameter>
mapOfTransactionParameterById)
```

Parameters

mapOfTransactionParameterById
Type: Map<String, TransactionParameter>

Parameter map is based on the payment gateway.

Return Value

Type: List<TransactionResult>

List of Transaction Results : List<TransactionResult>

`authorizeTransaction(mapOfTransactionParameterById)`

Authorizes a credit card payment. To perform a charge transaction, the user must follow the authorization with a capture transaction.

Signature

```
public static List<TransactionResult> authorizeTransaction(Map<String,
TransactionParameter> mapOfTransactionParameterById)
```

Parameters

mapOfTransactionParameterById
Type: Map<String, TransactionParameter>

Parameter map is based on the payment gateway.

Return Value

Type: List<TransactionResult>

List of Transaction Results : List<TransactionResult>

`captureTransaction(mapOfTransactionParameterById)`

Captures a payment for an authorize transaction.

Signature

```
global static List<TransactionResult> captureTransaction(Map<String,
TransactionParameter> mapOfTransactionParameterById)
```

Parameters

mapOfTransactionParameterById
Type: Map<String, TransactionParameter>

Parameter map is based on the payment gateway.

Return Value

Type: List<TransactionResult>

List of Transaction Results : List<TransactionResult>

`captureTransaction(captureInputParameters)`

Captures a payment for an authorization transaction using input parameters.

Signature

```
global static List<CaptureOutputResult> captureTransaction(List<CaptureInputParameter>
captureInputParameters)
```

```
blng.TransactionAPI, captureTransaction, [List<CaptureInputParameter>],
List<CaptureOutputResult>
```

Parameters

`captureInputParameters`
Type: `List<CaptureInputParameter>`

A list of input parameters passed from the `CaptureInputParameter` class.

Return Value

Type: List<CaptureOutputResult on page 30>

`chargeTransaction(mapOfTransactionParameterById)`

Authorize and capture a credit card payment or ACH payment.

Signature

```
public static List<TransactionResult> chargeTransaction(Map<String, TransactionParameter>
mapOfTransactionParameterById)
```

Parameters

`mapOfTransactionParameterById`
Type: Map<String, TransactionParameter>

Parameter map is based on the payment gateway.

Return Value

Type: List<TransactionResult>

List of Transaction Results : List<TransactionResult>

`getPaymentStatus(mapOfTransactionParameterById)`

Gets the status of a payment transaction.

Signature

```
global static List<TransactionResult> getPaymentStatus(Map<String, TransactionParameter>
mapOfTransactionParameterById)
```

Parameters

mapOfTransactionParameterById
Type: Map<String, TransactionParameter>

Parameter map is based on the payment gateway.

Return Value

Type: List<TransactionResult>

List of Transaction Results : List<TransactionResult>

`voidTransaction(mapOfTransactionParameterById)`

Cancels an original transaction that hasn't been settled.

Signature

```
global static List<TransactionResult> voidTransaction(Map<String, TransactionParameter>
mapOfTransactionParameterById)
```

Parameters

mapOfTransactionParameterById
Type: Map<String, TransactionParameter>

Parameter map is based on the payment gateway.

Return Value

Type: List<TransactionResult>

List of Transaction Results : List<TransactionResult>

`refundTransaction(mapOfTransactionParameterById)`

Refunds a customer for a transaction that has been succesfully settled through the payment gateway.

Signature

```
global static List<TransactionResult> refundTransaction(Map<String, TransactionParameter>
mapOfTransactionParameterById)
```

Parameters

mapOfTransactionParameterById
Type: Map<String, TransactionParameter>

Parameter map is based on the payment gateway.

Return Value

Type: List<TransactionResult>

List of Transaction Results : List<TransactionResult>

`nonReferredRefund(mapOfTransactionParameterById)`

Performs a non-referred refund of a credit card payment or an ACH payment.

Signature

```
global static List<TransactionResult> nonReferredRefund(Map<String, TransactionParameter>
mapOfTransactionParameterById)
```

Parameters

`mapOfTransactionParameterById`
Type: Map<String, TransactionParameter>

Parameter map is based on the payment gateway.

Return Value

Type: List<TransactionResult>

List of Transaction Results : List<TransactionResult>

`voidRefundTransaction(mapOfTransactionParameterById)`

Voids an unsettled refund transaction from a credit card payment or ACH payment.

Signature

```
global static List<TransactionResult> voidRefundTransaction(Map<String,
TransactionParameter> mapOfTransactionParameterById)
```

Parameters

`mapOfTransactionParameterById`
Type: Map<String, TransactionParameter>

Parameter map is based on the payment gateway.

Return Value

Type: List<TransactionResult>

List of Transaction Results : List<TransactionResult>

`getRefundStatus(mapOfTransactionParameterById)`

Gets the status of a refund transaction.

Signature

```
global static List<TransactionResult> getRefundStatus(Map<String, TransactionParameter>
mapOfTransactionParameterById)
```

Parameters

mapOfTransactionParameterById
Type: Map<String, TransactionParameter>

Parameter map is based on the payment gateway.

Return Value

Type: List<TransactionResult>

List of Transaction Results : List<TransactionResult>

voidTokenTransaction(mapOfTransactionParameterById)

Voids a credit card token.

Signature

```
global static List<TransactionResult> voidTokenTransaction(Map<String,
TransactionParameter> mapOfTransactionParameterById)
```

Parameters

mapOfTransactionParameterById
Type: Map<String, TransactionParameter>

Parameter map is based on the payment gateway.

Return Value

Type: List<TransactionResult>

List of Transaction Results : List<TransactionResult>

refundPayment(listofRefundParameters)

Refunds a payment.

Signature

```
global static List<TransactionResult> refundPayment(List<RefundParameter>
listofRefundParameters)
```

Parameters

listofRefundParameters
Type: List<RefundParameter>

Parameter map is based on the payment gateway.

Return Value

Type: List<TransactionResult>

List of Transaction Results : List<TransactionResult>

`resetInvoiceCorrectiveAction(invoices)`

Unlocks an invoice.

Signature

```
global static Boolean resetInvoiceCorrectiveAction(List<Invoice__c> invoices)
```

Parameters

`invoices`
Type: List<Invoice__c>

List of invoices to unlock due to payment issues.

Return Value

Type: Boolean

Returns True if invoice update is succesful, otherwise returns False.

`validateRefundTransactionId(transactionId)`

Validates whether a payment transaction can be refunded.

Signature

```
global static String validateRefundTransactionId(Id transactionId)
```

Parameters

`transactionId`
Type: Id

Single transactionId passed to validate whether a refund is allowed.

Return Value

Type: String

Returns String values such as the following.

- PAYMENT_TRANSACTION_NO_GATEWAY_ID_MESSAGE - Cannot refund payment transactions without gateway IDs. The invoice has been unlocked.
- INVALID_PAYMENT_TRANSACTION_FOR_REFUND_MESSAGE - Cannot refund payment transaction. This transaction is not currently eligible for refunding.
- ALLOW_REFUND_TRANSACTION - True
- REFUND_INVALID_PAYMENT_TRANSACTION_MESSAGE_ALOHA- Cannot refund invalid transactions or transactions without invoices.

`authorizeTransaction(invoiceID)`

Authorizes a payment for an invoice.

Signature

```
global static TransactionResult authorizeTransaction(Id invoiceID)
```

Parameters

`invoiceID`
Type: Id

Return Value

Type: TransactionResult

`chargeTransaction(invoiceID)`

Authorizes a payment for an invoice.

Signature

```
global static TransactionResult chargeTransaction(Id invoiceID)
```

Parameters

`invoiceID`
Type: Id

Return Value

Type: TransactionResult

TransactionParameter Class

Acts as a container to store calculation-specific information.

Namespace

blng

TransactionParameter Methods

The following are methods for `TransactionParameter`.

Sets RequestingInvoiceId.

getRequestingInvoiceId()

Gets Requesting InvoiceId.

setCardCodeResponse(cardCodeResponse)

Sets cardCodeResponse.

getCardCodeResponse()

Gets cardCodeResponse.

setCardNumber(requestingCreditCardNumber)

Sets requestingCreditCardNumber.

getCardNumber()

Gets requestingCreditCardNumber.

setPayment(paymentInstance)

Sets payment details like merchantRefId, RequestCreditCardNumber__c, CardCodeResponse__c, GatewayID__c. Also sets payment method details for PaymentType__c, CardExpirationYear__c, CardExpirationMonth__c, BankName__c, BankAccountType__c, BankAccountNumber__c, BankAccountName__c, CardType__c, BankRoutingCode__c if they aren't already set.

getPayment()

Gets Payment.

getMerchantId()

Gets merchantRefId.

setMerchantId()

Sets MerchantId.

getGatewayId()

Gets Gateway ID.

getResponseValueByKey()

Gets ResponseValueByKey

setGateway(gateway)

Sets Gateway ID.

setPaymentMethod(paymentMethod)

Sets PaymentMethod.

getPaymentMethod()

Gets PaymentMethod.

setInvoice(invoice)

Sets invoice.

getInvoice()

Gets Invoice.

setInputParameter(inputParameter)

Sets the input parameters for a payment capture transaction.

getInputParameter()

Returns the input parameters sent from the `inputParameter` class.

setInvoiceLine(line)

Sets listOfInvoiceLine.

getInvoiceLine()
Gets listOfInvoiceLine.

setTransaction(transactionInstance)
Sets transaction instance and details like merchantRefId, RequestCreditCardNumber, ResponseCode, and GatewayID.

getTransaction()
Gets Transaction.

setAccount(accountInstance)
Sets Account.

getAccount()
Gets Account.

setRequestBody(requestBody)
Sets RequestBody.

getRequestBody()
Gets RequestBody.

setFirstName(firstName)
Sets FirstName.

getFirstName()
Gets FirstName.

setLastName(lastName)
Sets LastName.

getLastName()
Gets LastName.

setEmailId(emailId)
Sets EmailId.

getEmailId()
Gets EmailId.

setStreet(street)
Sets street.

getStreet()
Gets Street.

setCity(city)
Sets City.

getCity()
Gets City.

setState(state)
Sets state.

getState()
Gets State.

setZipCode(zipCode)
Sets zip code.

getZipCode()
Gets ZipCode.

setCountry(country)
Sets Country.

getCountry()
Gets Country.

setPhone(phone)
Sets phone.

getPhone()
Gets Phone.

setCurrencyId(currencyId)
Sets Currency Id.

getCurrencyId()
Gets Currency Id.

setAmount(amount)
Sets amount.

getAmount()
Gets Amount.

`setRequestingInvoiceId(requestingInvoiceId)`

Sets RequestingInvoiceId.

Signature

```
global void setRequestingInvoiceId(Id requestingInvoiceId)
```

Parameters

`requestingInvoiceId`
Type: Id

Return Value

Type: void

`getRequestingInvoiceId()`

Gets Requesting InvoiceId.

Signature

```
global Id getRequestingInvoiceId()
```

Return Value

Type: Id

`setCardCodeResponse(cardCodeResponse)`

Sets cardCodeResponse.

Signature

```
global void setCardCodeResponse(String cardCodeResponse)
```

Parameters

`cardCodeResponse`
Type: String

Return Value

Type: void

`getCardCodeResponse()`

Gets cardCodeResponse.

Signature

```
global String getCardCodeResponse()
```

Return Value

Type: String

`setCardNumber(requestingCreditCardNumber)`

Sets requestingCreditCardNumber.

Signature

```
global void setCardNumber(String requestingCreditCardNumber)
```

Parameters

`requestingCreditCardNumber`
Type: String

Return Value

Type: void

`getCardNumber()`

Gets requestingCreditCardNumber.

Signature

```
global String getCardNumber()
```

Return Value

Type: String

`setPayment(paymentInstance)`

Sets payment details like merchantRefId, RequestCreditCardNumber__c, CardCodeResponse__c, GatewayID__c. Also sets payment method details for PaymentType__c, CardExpirationYear__c, CardExpirationMonth__c, BankName__c, BankAccountType__c, BankAccountNumber__c, BankAccountName__c, CardType__c, BankRoutingCode__c if they aren't already set.

Signature

```
global void setPayment(Payment__c paymentInstance)
```

Parameters

`paymentInstance`
Type: Payment__c

Return Value

Type: void

`getPayment()`

Gets Payment.

Signature

```
global Payment__c getPayment()
```

Return Value

Type: Payment__c

`getMerchantId()`

Gets merchantRefId.

Signature

```
global String getMerchantId()
```

Return Value

Type: String

`setMerchantId()`

Sets MerchantId.

Signature

```
global String setMerchantId()
```

Return Value

Type: String

`getGatewayId()`

Gets Gateway ID.

Signature

```
global String getGatewayId()
```

Return Value

Type: String

`getResponseValueByKey()`

Gets ResponseValueByKey

Signature

```
global Map<String, String> getResponseValueByKey()
```

Return Value

Type: Map<String, String>

`setGateway(gateway)`

Sets Gateway ID.

Signature

```
global void setGateway(PaymentGateway__c gateway)
```

Parameters

gateway
Type: PaymentGateway__c

Return Value

Type: void

`setPaymentMethod(paymentMethod)`

Sets PaymentMethod.

Signature

```
global void setPaymentMethod(PaymentMethod__c paymentMethod)
```

Parameters

paymentMethod
Type: PaymentMethod__c

Return Value

Type: void

`getPaymentMethod()`

Gets PaymentMethod.

Signature

```
global PaymentMethod__c getPaymentMethod()
```

Return Value

Type: PaymentMethod__c

`setInvoice(invoice)`

Sets invoice.

Signature

```
global void setInvoice(Invoice__c invoice)
```

Parameters

invoice
Type: Invoice__c

Return Value

Type: void

`getInvoice()`

Gets Invoice.

Signature

```
global Invoice__c getInvoice()
```

Return Value

Type: Invoice__c

setInputParameter(inputParameter)

Sets the input parameters for a payment capture transaction.

Signature

```
global void setInputParameter(InputParameter inputParameter)
```

Parameters

inputParameter
Type: InputParameter

Return Value

Type: void

getInputParameter()

Returns the input parameters sent from the `inputParameter` class.

Signature

```
global InputParameter getInputParameter()
```

Return Value

Type: InputParameter

setInvoiceLine(line)

Sets listOfInvoiceLine.

Signature

```
global void setInvoiceLine(InvoiceLine__c line)
```

Parameters

line
Type: InvoiceLine__c

Return Value

Type: void

`getInvoiceLine()`

Gets listOfInvoiceLine.

Signature

```
global List<InvoiceLine__c> getInvoiceLine()
```

Return Value

Type: List<InvoiceLine__c>

`setTransaction(transactionInstance)`

Sets transaction instance and details like merchantRefId, RequestCreditCardNumber, ResponseCode, and GatewayID.

Signature

```
global void setTransaction(PaymentTransaction__c transactionInstance)
```

Parameters

`transactionInstance`
Type: PaymentTransaction__c

Return Value

Type: void

`getTransaction()`

Gets Transaction.

Signature

```
global PaymentTransaction__c getTransaction()
```

Return Value

Type: PaymentTransaction__c

`setAccount(accountInstance)`

Sets Account.

Signature

```
global void setAccount(Account accountInstance)
```

Parameters

accountInstance
Type: Account

Return Value

Type: void

getAccount()

Gets Account.

Signature

```
global Account getAccount()
```

Return Value

Type: Account

setRequestBody(requestBody)

Sets RequestBody.

Signature

```
global void setRequestBody(String requestBody)
```

Parameters

requestBody
Type: String

Return Value

Type: void

getRequestBody()

Gets RequestBody.

Signature

```
global String getRequestBody()
```

Return Value

Type: String

`setFirstName(firstName)`

Sets FirstName.

Signature

```
global void setFirstName(String firstName)
```

Parameters

`firstName`
Type: String

Return Value

Type: void

`getFirstName()`

Gets FirstName.

Signature

```
global String getFirstName()
```

Return Value

Type: String

`setLastName(lastName)`

Sets LastName.

Signature

```
global void setLastName(String lastName)
```

Parameters

`lastName`
Type: String

Return Value

Type: void

`getLastName()`

Gets LastName.

Signature

```
global String getLastName()
```

Return Value

Type: String

setEmailId(emailId)

Sets EmailId.

Signature

```
global void setEmailId(String emailId)
```

Parameters

emailId
Type: String

Return Value

Type: void

getEmailId()

Gets EmailId.

Signature

```
global String getEmailId()
```

Return Value

Type: String

setStreet(street)

Sets street.

Signature

```
global void setStreet(String street)
```

Parameters

street
Type: String

Return Value

Type: void

`getStreet()`

Gets Street.

Signature

```
global String getStreet()
```

Return Value

Type: String

`setCity(city)`

Sets City.

Signature

```
global void setCity(String city)
```

Parameters

city
Type: String

Return Value

Type: void

`getCity()`

Gets City.

Signature

```
global String getCity()
```

Return Value

Type: String

`setState(state)`

Sets state.

Signature

```
global void setState(String state)
```

Parameters

state
Type: String

Return Value

Type: void

getState()

Gets State.

Signature

```
global String getState()
```

Return Value

Type: String

setZipCode(zipCode)

Sets zip code.

Signature

```
global void setZipCode(String zipCode)
```

Parameters

zipCode
Type: String

Return Value

Type: void

getZipCode()

Gets ZipCode.

Signature

```
global String getZipCode()
```

Return Value

Type: String

`setCountry(country)`

Sets Country.

Signature

```
global void setCountry(String country)
```

Parameters

`country`
Type: String

Return Value

Type: void

`getCountry()`

Gets Country.

Signature

```
global String getCountry()
```

Return Value

Type: String

`setPhone(phone)`

Sets phone.

Signature

```
global void setPhone(String phone)
```

Parameters

`phone`
Type: String

Return Value

Type: void

`getPhone()`

Gets Phone.

Signature

```
global String getPhone()
```

Return Value

Type: String

`setCurrencyId(currencyId)`

Sets Currency Id.

Signature

```
global void setCurrencyId(String currencyId)
```

Parameters

currencyId
Type: String

Return Value

Type: void

`getCurrencyId()`

Gets Currency Id.

Signature

```
global String getCurrencyId()
```

Return Value

Type: String

`setAmount(amount)`

Sets amount.

Signature

```
global void setAmount(String amount)
```

Parameters

amount
Type: String

Return Value

Type: void

`getAmount()`

Gets Amount.

Signature

```
global String getAmount()
```

Return Value

Type: String

TransactionResult Class

Holds the result of the transaction.

Namespace

blng

TransactionResult Methods

The following are methods for `TransactionResult`.

setResponseStatus(responseStatus)

Sets ResponseStatus.

getResponseStatus()

Gets ResponseStatus.

setGatewayRequest(gatewayRequest)

Sets the capture payment request sent to the payment gateway.

getGatewayRequest(gatewayRequest)

Returns the string containing the payment capture request sent to the gateway.

setGatewayResponse(gatewayResponse)

Returns the gateway's response following a capture payment transaction.

getGatewayResponse(gatewayResponse)

Returns the string containing the gateway's response from a capture payment transaction.

setResponseMessage(responseMessage)

Sets ResponseMessage.

getResponseMessage()

Gets ResponseMessage.

setResponseCode(responseCode)

Sets ResponseCode.

getResponseCode()

Gets ResponseCodeMessage.

setResponseToValidate(responseToValidate)

Sets ResponseToValidate.

getResponseToValidate()

Gets Response To Validate.

setGatewayId(gatewayId)

Sets Gateway ID.

getGatewayId()

Gets Gateway ID.

setType(type)

Sets Type.

getType()

Gets Type.

setRequestTypeOfPayment(requestTypeOfPayment)

Sets RequestTypeOfPayment.

getRequestTypeofPayment()

Gets RequestTypeOfPayment.

setRequestTransactionType(requestTransactionType)

Sets RequestTransactionType.

getRequestTransactionType()

Gets RequestTransactionType.

setIsSuccess(isSuccess)
Sets IsSuccess

getIsSuccess()
Gets IsSuccess.

setError(error)
Sets Error.

getErrors()
Gets Errors.

setResponseValueByKey(getResponseValueByKey)
Sets ResponseValueByKey

getResponseValueByKey()
Gets ResponseValueByKey.

setId(Id)
Sets ID.

getId()
Gets ID.

setMessage(message)
Sets Message.

getMessage()
Gets Message.

setEntity(entity)
Sets Message.

getEntity()
Gets Message.

setGatewayStatus(gatewayStatus)
Sets gatewayStatus.

getGatewayStatus()
Gets GatewayStatus.

`setResponseCodeMessage(responseCodeMessage)`

Sets ResponseCodeMessage.

Signature

```
global void setResponseCodeMessage(String responseCodeMessage)
```

Parameters

`responseCodeMessage`
Type: String

Return Value

Type: void

`getResponseCodeMessage()`

Gets ResponseCodeMessage.

Signature

```
global String getResponseCodeMessage()
```

Return Value

Type: String

`setCardCodeResponse(cardCodeResponse)`

Sets CardCodeResponse.

Signature

```
global void setCardCodeResponse(String cardCodeResponse)
```

Parameters

`cardCodeResponse`
Type: String

Return Value

Type: void

`getCardCodeResponse()`

Gets CardCodeResponse.

Signature

```
global String getCardCodeResponse()
```

Return Value

Type: String

`setCustomerProfileToken(customerProfileToken)`

Sets customerProfileToken.

Signature

```
global void setCustomerProfileToken(String customerProfileToken)
```

Parameters

customerProfileToken
Type: String

Return Value

Type: void

getCustomerProfileToken()

Gets CustomerProfileToken.

Signature

```
global String getCustomerProfileToken()
```

Return Value

Type: String

setPaymentToken(paymentToken)

Sets PaymentToken.

Signature

```
public void setPaymentToken(String paymentToken)
```

Parameters

paymentToken
Type: String

Return Value

Type: void

getPaymentToken()

Gets PaymentToken.

Signature

```
global String getPaymentToken()
```

Return Value

Type: String

`setResponseStatus(responseStatus)`

Sets ResponseStatus.

Signature

```
global void setResponseStatus(String responseStatus)
```

Parameters

`responseStatus`
Type: String

Return Value

Type: void

`getResponseStatus()`

Gets ResponseStatus.

Signature

```
global String getResponseStatus()
```

Return Value

Type: String

setGatewayRequest(gatewayRequest)

Sets the capture payment request sent to the payment gateway.

Signature

```
global void setGatewayRequest(String gatewayRequest)
```

Parameters

`gatewayRequest`
Type: String

Return Value

Type: Void

getGatewayRequest(gatewayRequest)

Returns the string containing the payment capture request sent to the gateway.

Signature

```
global String GatewayRequest()
```

Return Value

Type: String

setGatewayResponse(gatewayResponse)

Returns the gateway's response following a capture payment transaction.

Signature

```
global void setGatewayResponse(String gatewayResponse)
```

Parameters

gatewayResponse
Type: String

Return Value

Type: void

getGatewayResponse(gatewayResponse)

Returns the string containing the gateway's response from a capture payment transaction.

Signature

```
global void getGatewayResponse()
```

Return Value

Type: Void

setResponseMessage(responseMessage)

Sets ResponseMessage.

Signature

```
global void setResponseMessage(String responseMessage)
```

Parameters

responseMessage
Type: String

Return Value

Type: void

`getResponseMessage()`

Gets ResponseMessage.

Signature

```
global String getResponseMessage()
```

Return Value

Type: String

`setResponseCode(responseCode)`

Sets ResponseCode.

Signature

```
global void setResponseCode(String responseCode)
```

Parameters

`responseCode`
Type: String

Return Value

Type: void

`getResponseCode()`

Gets ResponseCodeMessage.

Signature

```
global String getResponseCode()
```

Return Value

Type: String

`setResponseToValidate(responseToValidate)`

Sets ResponseToValidate.

Signature

```
global void setResponseToValidate(String responseToValidate)
```

Parameters

responseToValidate
Type: String

Return Value

Type: void

getResponseToValidate()

Gets Response To Validate.

Signature

```
public String getResponseToValidate()
```

Return Value

Type: String

setGatewayId(gatewayId)

Sets Gateway ID.

Signature

```
global void setGatewayId(String gatewayId)
```

Parameters

gatewayId
Type: String

Return Value

Type: void

getGatewayId()

Gets Gateway ID.

Signature

```
global String getGatewayId()
```

Return Value

Type: String

`setType(type)`

Sets Type.

Signature

```
global void setType(String type)
```

Parameters

type
Type: String

Return Value

Type: void

`getType()`

Gets Type.

Signature

```
global String getType()
```

Return Value

Type: String

`setRequestTypeOfPayment(requestTypeOfPayment)`

Sets RequestTypeOfPayment.

Signature

```
global void setRequestTypeOfPayment(String requestTypeOfPayment)
```

Parameters

requestTypeOfPayment
Type: String

Return Value

Type: void

`getRequestTypeofPayment()`

Gets RequestTypeOfPayment.

Signature

```
global String getRequestTypeofPayment()
```

Return Value

Type: String

`setRequestTransactionType(requestTransactionType)`

Sets RequestTransactionType.

Signature

```
global void setRequestTransactionType(String requestTransactionType)
```

Parameters

`requestTransactionType`
Type: String

Return Value

Type: void

`getRequestTransactionType()`

Gets RequestTransactionType.

Signature

```
global String getRequestTransactionType()
```

Return Value

Type: String

`setIsSuccess(isSuccess)`

Sets IsSuccess

Signature

```
global void setIsSuccess(Boolean isSuccess)
```

Parameters

`isSuccess`
Type: Boolean

Return Value

Type: void

`getIsSuccess()`

Gets IsSuccess.

Signature

```
global Boolean getIsSuccess()
```

Return Value

Type: Boolean

`setError(error)`

Sets Error.

Signature

```
global void setError(String error)
```

Parameters

error
Type: String

Return Value

Type: void

`getErrors()`

Gets Errors.

Signature

```
public List<String> getErrors()
```

Return Value

Type: List<String>

`setResponseValueByKey(getResponseValueByKey)`

Sets ResponseValueByKey

Signature

```
public void setResponseValueByKey(Map<String,String> getResponseValueByKey)
```

Parameters

getResponseValueByKey
Type: Map<String,String>

Return Value

Type: void

getResponseValueByKey()

Gets ResponseValueByKey.

Signature

```
global Map<string, string> getResponseValueByKey()
```

Return Value

Type: Map<string, string>

setId(Id)

Sets ID.

Signature

```
global void setId(String Id)
```

Parameters

Id
Type: String

Return Value

Type: void

getId()

Gets ID.

Signature

```
global String getId()
```

Return Value

Type: String

`setMessage(message)`

Sets Message.

Signature

```
global void setMessage(String message)
```

Parameters

message
Type: String

Return Value

Type: void

`getMessage()`

Gets Message.

Signature

```
global String getMessage()
```

Return Value

Type: String

`setEntity(entity)`

Sets Message.

Signature

```
global void setEntity(String entity)
```

Parameters

entity
Type: String

Return Value

Type: void

`getEntity()`

Gets Message.

Signature

```
global String getEntity()
```

Return Value

Type: String

`setGatewayStatus(gatewayStatus)`

Sets gatewayStatus.

Signature

```
global void setGatewayStatus(GatewayStatusType gatewayStatus)
```

Parameters

`gatewayStatus`
Type: GatewayStatusType

Return Value

Type: void

`getGatewayStatus()`

Gets GatewayStatus.

Signature

```
global GatewayStatusType getGatewayStatus()
```

Return Value

Type: GatewayStatusType

TransactionResult.GatewayStatusType Enum

An enum defining the appropriate gateway status values returned by the payment gateway.

Enum Values

The following are the values of the `blng.TransactionResult.GatewayStatusType` enum.

Value	Description
`Success`	The gateway has processed the transaction succesfully.
`Decline`	Declined responses occur when the gateway's call to the customer bank fails. This usually happens for customer issues such as insufficient funds, a frozen credit card, or a brief disconnection during the call.

Value	Description
`ValidationError`	Gateways respond with validation errors when Salesforce Billing sends incorrect customer payment information, such as misspelled credit card data or a CVV with missing numbers. Payment runs won't create payments from transactions with a gateway status of Validation Error. Users should change the incorrect payment information on their payment method so that future payment runs can use it for payments.
`PermanentFail`	Gateways respond with a permanent failure code when the customer's bank recognizes their payment account as closed, terminated, or fraudulent. In this case, the gateway won't accept any further calls from the payment method associated with this transaction. Following a permanent fail response, a transaction changes its gateway status to Permanent Fail.
`RequiresReview`	Gateways respond with a review code when the gateway call initially fails, but the payment method may still work following extra processing. For example, some banks send out this type of response when they have further questions about the payment request, and will provide an authorization code manually when the payment manager calls the processor.
`Indeterminate`	Gateways send indeterminate responses when they require the user to check the status of the transaction request. This often occurs following server timeouts, system failure, or any action that unexpectedly interrupts the gateway's ability to process the payment.
`SystemError`	Salesforce Billing provides a system fail response when it has to end its gateway payment request before receiving a response. This can happen due to gateway server errors, invalid customer credentials, or anytime Salesforce Billing's request times out before it receives a gateway response. System failures occur before Salesforce Billing's request reaches the gateway, so there's no risk of an unaccounted payment remaining in the gateway. If your transaction has a System Fail gateway status, you can either manually create the payment or wait for another payment run to pick up your invoice.

Custom Gateway Classes

Salesforce Billing provides three classes that users can customize to integrate with their chosen payment gateway.

YourGatewayAPI

YourGatewayAPI is a base class that connects the Salesforce Billing package to your payment gateway package. All user gateway API classes will be different based on your needs. However, we've provided a template that you can use to help create your own.

YourGatewayName

YourGateWayName is a base class. All user gateway API classes will be different based on your needs. However, we've provided a template that you can use to help create your own.

YourGatewayNameUtils

YourGatewayNameUtils is a singleton utility class that interacts with an external payment gateway. All user gateway API classes will be different based on your needs. However, we've provided a template that you can use to help create your own.

YourHttpService

YourHttpService is an example of common HTTP utilities that might be required while building a partner package. All user gateway API classes will be different based on your needs. However, we've provided a template that you can use to help create your own.

YourGatewayAPI

YourGatewayAPI is a base class that connects the Salesforce Billing package to your payment gateway package. All user gateway API classes will be different based on your needs. However, we've provided a template that you can use to help create your own.

`YourGatewayAPI` references the following public classes.

- YourGatewayName
- YourGatewayNameUtils

EDITIONS

Available in:

```
global class YourGatewayAPI implements blng.PaymentGateWay, blng.PaymentGateWays,
blng.PaymentGateWayStatus
{
    // ================================================================================
    // CONSTANT
    // ================================================================================

    // ================================================================================
    // STATIC VARIABLES
    // ================================================================================

    // Attribute to implement singleton pattern for YourGatewayName  class
    private static YourGatewayName YourGatewayNameInstance;

     // ================================================================================
    // VARIABLES
    // ================================================================================

    // Attribute to implement singleton pattern for AuthorizeDotNet class
    private static final String GENERATE_TOKEN = 'generateToken';
    private static final String AUTHORIZE_TRANSACTION = 'authorizeTransaction';
    private static final String CHARGE_TRANSACTION = 'chargeTransaction';
    private static final String VOID_TRANSACTION = 'voidTransaction';
    private static final String REFUND_TRANSACTION = 'refundTransaction';
    private static final String CAPTURE_TRANSACTION = 'captureTransaction';
    private static final String VOID_REFUND_TRANSACTION = 'voidRefundTransaction';
    private static final String VOID_TOKEN_TRANSACTION = 'voidTokenTransaction';
    private static final String GET_PAYMENT_STATUS = 'getPaymentStatus';
    private static final String GET_REFUND_STATUS = 'getRefundStatus';
    private static final String NON_REFERRED_REFUND = 'nonReferredRefund';
    private static final String PROCESS_PAYMENTS = 'processPayments';

    // ================================================================================
    // Methods
    // ================================================================================
    /**
    * @name service
    * @description service method to create an only instance of TransactionService class
    *              if serviceInstance is NULL then only create a new instance otherwise
```

```
    *              return the existing one
    * @param   NA
    * @return TransactionService instance
    * @exception
    * @author
    * @created
    * @remark
    * @change
    */
    @TestVisible
    private static YourGatewayName service()
    {
        if (NULL == YourGatewayNameInstance)
        {
            YourGatewayNameInstance = YourGatewayName.getInstance();
        }
        return YourGatewayNameInstance;
    }

    /**
    * @name processPayments
    * @description Method to generate Token
    * @param
    * @return
    * @exception
    * @author
    * @created 2016-11-03
    * @remark
    * @change
    */
    global static Map<String, blng.TransactionResult> processPayments(Map<String,
blng.TransactionParameter> mapOfTransactionParameterById, blng.PaymentGatewayParameter
paymentGatewayParameter)
    {
        if(paymentGatewayParameter.getTransactionType() == GENERATE_TOKEN)
        {
            return generateToken(mapOfTransactionParameterById);
        }
        else if(paymentGatewayParameter.getTransactionType() == AUTHORIZE_TRANSACTION)
        {
            return authorizeTransaction(mapOfTransactionParameterById);
        }
        else if(paymentGatewayParameter.getTransactionType() == CHARGE_TRANSACTION)
        {
            return chargeTransaction(mapOfTransactionParameterById);
        }
        else if(paymentGatewayParameter.getTransactionType() == VOID_TRANSACTION)
        {
            return voidTransaction(mapOfTransactionParameterById);
        }
        else if(paymentGatewayParameter.getTransactionType() == REFUND_TRANSACTION)
        {
            return refundTransaction(mapOfTransactionParameterById);
        }
```

```
        else if(paymentGatewayParameter.getTransactionType() == CAPTURE_TRANSACTION)
        {
            return captureTransaction(mapOfTransactionParameterById);
        }
        else if(paymentGatewayParameter.getTransactionType() == VOID_REFUND_TRANSACTION)
        {
            return voidRefundTransaction(mapOfTransactionParameterById);
        }
        else if(paymentGatewayParameter.getTransactionType() == VOID_TOKEN_TRANSACTION)
        {
            return voidTokenTransaction(mapOfTransactionParameterById);
        }
        else if(paymentGatewayParameter.getTransactionType() == GET_PAYMENT_STATUS)
        {
            return getPaymentStatus(mapOfTransactionParameterById);
        }
        else if(paymentGatewayParameter.getTransactionType() == GET_REFUND_STATUS)
        {
            return getRefundStatus(mapOfTransactionParameterById);
        }
        else if(paymentGatewayParameter.getTransactionType() == NON_REFERRED_REFUND)
        {
            return nonReferredRefund(mapOfTransactionParameterById);
        }

        return NULL;
    }

    /**
    * @name generateToken
    * @description Method to generate Token for a Payment Method
    * @param Map[Key => String [unique Id],Value => TransactionParameter]
    * @return Map[Key => String [unique Id],Value => TransactionResult]
    * @exception
    * @author
    * @created
    * @remark
    * @change
    */
    global static Map<String, blng.TransactionResult> generateToken(Map<String,
blng.TransactionParameter> mapOfTransactionParameterById)
    {
        return service().generateToken(mapOfTransactionParameterById);
    }

    /**
    * @name voidTokenTransaction
    * @description Method to void Token for a Payment method Token
    * @param Map[Key => String [unique Id],Value => TransactionParameter]
    * @return Map[Key => String [unique Id],Value => TransactionResult]
    * @exception
    * @author
    * @created
    * @remark
```

```
    * @change
    */
    global static Map<String, blng.TransactionResult> voidTokenTransaction(Map<String,
blng.TransactionParameter> mapOfTransactionParameterById)
    {
        return service().voidTokenTransaction(mapOfTransactionParameterById);
    }


    /**
    * @name authorizeTransaction
    * @description Method to Authorize a payment for a Invoice
    * @param Map[Key => String [unique Id],Value => TransactionParameter]
    * @return Map[Key => String [unique Id],Value => TransactionResult]
    * @exception
    * @author
    * @created
    * @remark
    * @change
    */
    global static Map<String, blng.TransactionResult> authorizeTransaction(Map<String,
blng.TransactionParameter> mapOfTransactionParameterById)
    {
        return service().authorizeTransaction(mapOfTransactionParameterById);
    }

    /**
    * @name chargeTransaction
    * @description Method to Charge a payment for a Invoice
    * @param Map[Key => String [unique Id],Value => TransactionParameter]
    * @return Map[Key => String [unique Id],Value => TransactionResult]
    * @exception
    * @author
    * @created
    * @remark
    * @change
    */
    global static Map<String, blng.TransactionResult> chargeTransaction(Map<String,
blng.TransactionParameter> mapOfTransactionParameterById)
    {
        return service().chargeTransaction(mapOfTransactionParameterById);
    }

    /**
    * @name captureTransaction
    * @description Method to capture a payment for a Authorize Transaction
    * @param Map[Key => String [unique Id],Value => TransactionParameter]
    * @return Map[Key => String [unique Id],Value => TransactionResult]
    * @exception
    * @author
    * @created
    * @remark
    * @change
    */
```

```
    global static Map<String, blng.TransactionResult> captureTransaction(Map<String,
blng.TransactionParameter> mapOfTransactionParameterById)
    {
        return service().captureTransaction(mapOfTransactionParameterById);
    }


    /**
    * @name voidTransaction
    * @description Method to Void a payment for a Invoice
    * @param Map[Key => String [unique Id],Value => TransactionParameter]
    * @return Map[Key => String [unique Id],Value => TransactionResult]
    * @exception
    * @author
    * @created
    * @remark
    * @change
    */
    global static Map<String, blng.TransactionResult> voidTransaction(Map<String,
blng.TransactionParameter> mapOfTransactionParameterById)
    {
        return service().voidTransaction(mapOfTransactionParameterById);
    }

    /**
    * @name refundTransaction
    * @description Method to Refund a payment for a payment Transaction
    * @param Map[Key => String [unique Id],Value => TransactionParameter]
    * @return Map[Key => String [unique Id],Value => TransactionResult]
    * @exception
    * @author
    * @created
    * @remark
    * @change
    */
    global static Map<String, blng.TransactionResult> refundTransaction(Map<String,
blng.TransactionParameter> mapOfTransactionParameterById)
    {
        return service().refundTransaction(mapOfTransactionParameterById);
    }

    /**
    * @name nonReferencedRefund
    * @description Method to non Referenced Refund a payment for a Invoice
    * @param Map[Key => String [unique Id],Value => TransactionParameter]
    * @return Map[Key => String [unique Id],Value => TransactionParameter]
    * @exception
    * @author
    * @created
    * @remark
    * @change
    */
    global static Map<String, blng.TransactionResult> nonReferredRefund(Map<String,
blng.TransactionParameter> mapOfTransactionParameterById)
```

```
    {
        return service().nonReferencedRefund(mapOfTransactionParameterById);
    }

    /**
    * @name voidRefundTransaction
    * @description Method to Void refund for a Refund Transaction
    * @param Map[Key => String [unique Id],Value => TransactionParameter]
    * @return Map[Key => String [unique Id],Value => TransactionResult]
    * @exception
    * @author
    * @created
    * @remark
    * @change
    */
    global static Map<String, blng.TransactionResult> voidRefundTransaction(Map<String,
blng.TransactionParameter> mapOfTransactionParameterById)
    {
        return service().voidRefundTransaction(mapOfTransactionParameterById);
    }

    /**
    * @name getPaymentStatus
    * @description Method to get payment status for a Payment Transaction
    * @param Map[Key => String [unique Id],Value => TransactionParameter]
    * @return Map[Key => String [unique Id],Value => TransactionResult]
    * @exception
    * @author
    * @created
    * @remark
    * @change
    */
    global static Map<String, blng.TransactionResult> getPaymentStatus(Map<String,
blng.TransactionParameter> mapOfTransactionParameterById)
    {
        return service().getPaymentStatus(mapOfTransactionParameterById);
    }

    /**
    * @name getRefundStatus
    * @description Method to get refund status for a Refund Transaction
    * @param Map[Key => String [unique Id],Value => TransactionParameter]
    * @return Map[Key => String [unique Id],Value => TransactionResult]
    * @exception
    * @author
    * @created
    * @remark
    * @change
    */
    global static Map<String, blng.TransactionResult> getRefundStatus(Map<String,
blng.TransactionParameter> mapOfTransactionParameterById)
    {
         return service().getPaymentStatus(mapOfTransactionParameterById);
    }
```

```
     /**
     * @name populateGatewayStatus
     * @description Method that populates the GatewayStatus enum on the TransactionResult,

     * given return codes that are already set in the TransactionResults
     * @param transactionResult
     */
    global static void populateGatewayStatus(blng.TransactionResult transactionResult)
    {
        service().populateGatewayStatus(transactionResult);
    }
}
```

YourGatewayName

YourGateWayName is a base class. All user gateway API classes will be different based on your needs. However, we've provided a template that you can use to help create your own.

This class shows an example of a void token calling the CyberSource payment gateway. The gateway receives a request and then provides a response through the TransactionResult object. Salesforce Billing evaluates the response and filters it into one of several fields shown in the payment transaction record's Gateway Status field. For more information on payment transactions and managing gateway responses in Salesforce Billing, check out Managing Payment Transactions.

EDITIONS

Available in: All Salesforce Billing Editions

```
Note : Map<String, blng.TransactionParameter> is the parameter in all methods of this class

      blng.TransactionParameter class is a global class which list of global method exposed
 detailed in next section
       and String in the map is a Unique Id which could be Invoice or Account Id

       Map<String, blng.TransactionResult> is the return for all the Methods in this class

      blng.TransactionResult class is a global class with a list of global method detailed
 in next section
       the string should be the same string of parameter

       public class YourGatewayName
{
    //Add variables here
    //Gateway Status related variables
    private static Map<string, blng.TransactionResult.GatewayStatusType>
mapGatewayStatusEnumTypesByStrings =
            new Map<string, blng.TransactionResult.GatewayStatusType>();
    static {
        List<blng.TransactionResult.GatewayStatusType> enumValues =
blng.TransactionResult.GatewayStatusType.values();
        for (Integer i = 0; i < enumValues.size(); i++) {
            mapGatewayStatusEnumTypesByStrings.put(enumValues.get(i).name(),
enumValues.get(i));
        }
    }
```

```
 //
    // ============================================================================
    // CONSTANT
    // ============================================================================

        private static final String ACCEPT = 'ACCEPT';
        private static final String REJECT = 'REJECT';
        private static final String SUCCESS = 'SUCCESS';
        private static final String FAILURE = 'FAILURE';
        private static final String ERROR = 'ERROR';
        private static final String DECISION = 'decision';

    /**
    * @name handleError
    * @description populates TransactionResult with error values
    * @Param
    * @return NA
    * @exception NA
    */
    private void handleError(blng.TransactionResult transactionResult,
            String pointOfFailure,
            blng.TransactionResult.GatewayStatusType gatewayStatus,
            Exception e) {
        String newErrorMessage = pointOfFailure + (e != null ? ': ' + e.getMessage() :
'');

        // if there are previous errors, set the response message by concatenating them
in the right order
        if (!transactionResult.getErrors().isEmpty()) {
            String prevErrorMessage = '';
            for (String err : transactionResult.getErrors()) {
                prevErrorMessage += (String.isEmpty(prevErrorMessage) ? '' : '. ') + err;

            }
            transactionResult.setResponseMessage(prevErrorMessage);
        } else {
            transactionResult.setResponseMessage(newErrorMessage);
        }

        // set the rest of the values
        transactionResult.setIsSuccess(false);
        transactionResult.setError(newErrorMessage);
        transactionResult.setResponseToValidate(FAILURE);
        if (transactionResult.getResponseCode() != null) {
            populateGatewayStatus(transactionResult);
        } else {
            // set the response code to ERROR if it does not exist, which will be in all
cases except when the response returns from the gateway
            transactionResult.setResponseCode(ERROR);
            transactionResult.setGatewayStatus(gatewayStatus);
        }
    }

    /**
```

```
    * @name populateTransactionResultForVoidToken
    * @description Method to populate Transaction Result for CyberSource Void Token
    * @param Map[Key => String [unique Id],Value => TransactionParameter]
    * @return Map[Key => String [unique Id],Value => TransactionResult]
    */
    public Map<String, blng.TransactionResult>
populateTransactionResultForVoidToken(Map<String, blng.TransactionParameter>
mapOfTransactionParameterById)
    {
        Map<String, blng.TransactionResult> mapOfTransactionResultById = new Map<String,
blng.TransactionResult>();
        for(String idToProcess : mapOfTransactionParameterById.KeySet())
        {
            blng.TransactionParameter transactionParameterToProcess;
            try {
                transactionParameterToProcess =
mapOfTransactionParameterById.get(idToProcess);
                if (NULL !=
transactionParameterToProcess.transactionResult.getGatewayStatus()) {
                    mapOfTransactionResultById.put(idToProcess,
transactionParameterToProcess.transactionResult);
                    continue;
                }
                //Populating the transaction Parameter class based on the response is
success or failure
                if(transactionParameterToProcess.getResponseValueByKey().get(DECISION) ==
 ACCEPT)
                {
                    //The response is success

transactionParameterToProcess.transactionResult.setResponseToValidate(SUCCESS);
                    transactionParameterToProcess.transactionResult.setIsSuccess(true);
                }
              else if(transactionParameterToProcess.getResponseValueByKey().get(DECISION)
 == REJECT)
                {
                    //The response is failure

transactionParameterToProcess.transactionResult.setResponseToValidate(FAILURE);
                    transactionParameterToProcess.transactionResult.setIsSuccess(false);
                }
                //Populating payment gateway response
                transactionParameterToProcess.transactionResult.setId(idToProcess);
                transactionParameterToProcess.transactionResult.setResponseCode
                 (transactionParameterToProcess.getResponseValueByKey().get(REASONCODE));


             if(NULL != transactionParameterToProcess.transactionResult.getResponseCode())

                {
                   transactionParameterToProcess.transactionResult.setResponseCodeMessage


(YourGatewayNameUtils.getGatewayReturnCode().get(transactionParameterToProcess.transactionResult.getResponseCode()));
```

```
                }
                //Populating payment gateway response
                transactionParameterToProcess.transactionResult.setPaymentToken

(transactionParameterToProcess.getResponseValueByKey().get(SUBSCRIPTIONID));
                //Populate gateway status
                populateGatewayStatus(transactionParameterToProcess.transactionResult);

mapOfTransactionResultById.put(idToProcess,transactionParameterToProcess.transactionResult);

            } catch (Exception e) {
                handleError(transactionParameterToProcess.transactionResult,
FAILED_POPULATING_RESULT + IN_VOID_TOKEN,
blng.TransactionResult.GatewayStatusType.SystemError, e);
            }
        }
        //Return map of transaction results
        return mapOfTransactionResultById;
    }



    // default Gateway Status
    private static final blng.TransactionResult.GatewayStatusType defaultGatewayStatus =
blng.TransactionResult.GatewayStatusType.Indeterminate;

    //Generate Token method Salesforce billing interface class will use this class to
Generate token
    public static Map<String, blng.TransactionResult> generateToken(Map<String,
blng.TransactionParameter> mapOfTransactionParameterById)
    {
       //Actual Implementation for Token takes place here
    }

    //Void Token method Salesforce billing interface class will use this class to void
Token Transaction
    public static Map<String, blng.TransactionResult> voidTokenTransaction(Map<String,
blng.TransactionParameter> mapOfTransactionParameterById)
    {
       //Actual Implementation for Void Token takes place here
       if(!mapOfTransactionParameterById.IsEmpty())
        {
            for(blng.TransactionParameter transactionParameterToProcess :
mapOfTransactionParameterById.Values())
            {
                // validate gateway settings
                try {

YourGatewayNameUtils.validateGatewaySettings(transactionParameterToProcess);
                    if
(!transactionParameterToProcess.transactionResult.getErrors().IsEmpty()) {
                        handleError(transactionParameterToProcess.transactionResult,
FAILED_GATEWAY_SETTINGS + IN_VOID_TOKEN,
```

```
blng.TransactionResult.GatewayStatusType.ValidationError, null);
                        continue;
                    }
                } catch (Exception e) {
                    handleError(transactionParameterToProcess.transactionResult,
FAILED_GATEWAY_SETTINGS + IN_VOID_TOKEN,
blng.TransactionResult.GatewayStatusType.ValidationError, e);
                    continue;
                }
               // Calling YourGatewayNameUtils class to populate the Request for generate
 void token XML
                try {

YourGatewayNameUtils.getInstance().generateVoidTokenXML(transactionParameterToProcess);
                } catch (Exception e) {
                    handleError(transactionParameterToProcess.transactionResult,
FAILED_XML_GENERATION + IN_VOID_TOKEN, blng.TransactionResult.GatewayStatusType.SystemError,
 e);
                    continue;
                }
                try {
                    // Calling YourHttpService class to send a Request
                    YourHttpService sendHttpRequest = YourHttpService.getInstance();
                    sendHttpRequest.addHeader('Content-type', 'text/xml');

sendHttpRequest.setTokenisationHeader(transactionParameterToProcess.getGateWay().MerchantId__c,
 transactionParameterToProcess.getGateWay().TransactionSecurityKey__c);
                    // Sends the request to payment gateway

sendHttpRequest.post(transactionParameterToProcess.getGateWay().TestMode__c ?
YOUR_GATEWAY_TEST_ENDPOINT_URL_SANDBOX :
YOUR_GATEWAY_TEST_ENDPOINT_PRODUCTION,transactionParameterToProcess.getRequestBody());
                    if(!Test.isRunningTest())
                    {
                        // Populating the map of Response for transaction parameter class
 from the response received by payment gateway
                        transactionParameterToProcess.mapOfResponseValueByKey.putAll

(YourGatewayNameUtils.getElements(sendHttpRequest.getResponse().getBodyDocument().getRootElement()));

                    }
                    else
                    {
                        Dom.Document doc = new Dom.Document();
                        doc.load(TEST_RESPONSE_BODY); // You can provide a string for
TEST_RESPONSE_BODY in test mode
                        transactionParameterToProcess.mapOfResponseValueByKey.putAll
                        (YourGatewayNameUtils.getElements(doc.getRootElement()));
                    }
                } catch (Exception e) {
                    handleError(transactionParameterToProcess.transactionResult,
FAILED_CALLOUT + IN_VOID_TOKEN, blng.TransactionResult.GatewayStatusType.SystemError, e);

                }
```

```
            }
        }
        // Calling populate Transaction Result For Void Token class to Return map of
transaction results
        return populateTransactionResultForVoidToken(mapOfTransactionParameterById);
    }

    //Authorize Transaction method Salesforce billing interface class will use this class
 to Authorize Transaction
    public static Map<String, blng.TransactionResult> authorizeTransaction(Map<String,
blng.TransactionParameter> mapOfTransactionParameterById)
    {
        //Actual Implementation for Authorize takes place here
    }

    //Capture Transaction method Salesforce billing interface class will use this class
to capture Transaction
    public static Map<String, blng.TransactionResult> captureTransaction(Map<String,
blng.TransactionParameter> mapOfTransactionParameterById)
    {
        //Actual Implementation for capture takes place here
    }

    //Charge Transaction method Salesforce billing interface class will use this class to
 Charge Transaction
    public static Map<String, blng.TransactionResult> chargeTransaction(Map<String,
blng.TransactionParameter> mapOfTransactionParameterById)
    {
        //Actual Implementation for Charge takes place here
    }

    //void Transaction method Salesforce billing interface class will use this class to
void Transaction
    public static Map<String, blng.TransactionResult> voidTransaction(Map<String,
blng.TransactionParameter> mapOfTransactionParameterById)
    {
        //Actual Implementation for void Transaction takes place here
    }

    //RefundTransaction method Salesforce billing interface class will use this class to
Refund Transaction
    public static Map<String, blng.TransactionResult> refundTransaction(Map<String,
blng.TransactionParameter> mapOfTransactionParameterById)
    {
        //Actual Implementation for Refund takes place here
    }

    //Non Referenced Refund method Salesforce billing interface class will use this class
 to non referenced refund Transaction
    public static Map<String, blng.TransactionResult> nonReferencedRefund(Map<String,
blng.TransactionParameter> mapOfTransactionParameterById)
    {
        //Actual Implementation for non referenced refund takes place here
    }
```

```
    //void Refund method Salesforce billing interface class will use this class to non
void Refund Transaction
    public static Map<String, blng.TransactionResult> voidRefundTransaction(Map<String,
blng.TransactionParameter> mapOfTransactionParameterById)
    {
        //Actual Implementation for void Refund Transaction takes place here
    }

    //Get payment status method Salesforce billing interface class will use this class to
 get payment status Transaction
    public static Map<String, blng.TransactionResult> getPaymentStatus(Map<String,
blng.TransactionParameter> mapOfTransactionParameterById)
    {
        //Actual Implementation for get payment status takes place here
    }

    //Get refund status method Salesforce billing interface class will use this class to
get refund status Transaction
    public static Map<String, blng.TransactionResult> getRefundStatus(Map<String,
blng.TransactionParameter> mapOfTransactionParameterById)
    {
        //Actual Implementation for get refund status takes place here
    }
    /**
     * Does a SOQL lookup on the mapper table and gets the gateway status mapped to the
return code
     * Returns the default enum if no match is found
     * @param transactionResult
     */
     public void populateGatewayStatus(blng.TransactionResult transactionResult) {
         blng.TransactionResult.GatewayStatusType gatewayStatus = defaultGatewayStatus;
         // actual implementation of mapping
         transactionResult.setGatewayStatus(gatewayStatus);
     }
}
```

YourGatewayNameUtils

YourGatewayNameUtils is a singleton utility class that interacts with an external payment gateway. All user gateway API classes will be different based on your needs. However, we've provided a template that you can use to help create your own.

This code shows an example of how to generate a void token request for CyberSource. The class contains several methods.

- A utility method to construct the void token request payload for CyberSource
- A utility method to validate gateway settings
- A utility to return a map of CyberSource Gateway Return code to a friendly description
- A utility method to pass the XML node and retrieve a map of element to value

EDITIONS

Available in: All Salesforce Billing Editions

```
public class YourGatewayNameUtils
{
```

```
    // ================================================================================
    // STATIC VARIABLES
    // ================================================================================

    // private Attribute to implement singleton pattern for YourGatewayNameUtils class
    private static YourGatewayNameUtils yourGatewayNameUtilsInstance;



    /**
     * @name getInstance
     * @description get an Instance of Service class
     * @param NA
     * @return YourGatewayNameUtils Generator Class Instance
     */
    public static YourGatewayNameUtils getInstance() {
        if (NULL == yourGatewayNameUtilsInstance) {
            yourGatewayNameUtilsInstance = new YourGatewayNameUtils();
        }
        return yourGatewayNameUtilsInstance;
    }

    /**
    * @name validateGatewaySettings
    * @description Validates Gateway Details
    * @param TransactionParameter
    * @return NA
    * @exception NA
    */
    public static void validateGatewaySettings(blng.TransactionParameter
transactionParameterToProcess) {
        if (String.IsBlank(transactionParameterToProcess.getGateWay().MerchantId__c)) {
            transactionParameterToProcess.transactionResult.setError('MerchantId is
missing');
        } else if
(String.IsBlank(transactionParameterToProcess.getGateWay().MerchantReference__c)) {
            transactionParameterToProcess.transactionResult.setError('MerchantReference
is missing');
        } else if
(String.IsBlank(transactionParameterToProcess.getGateWay().TransactionSecurityKey__c)) {
          transactionParameterToProcess.transactionResult.setError('TransactionSecurityKey
 is missing');
        }
    }

        /**
        * @name getErrorCyberSourceCode
        * @description Return's Map of Cyber source error description By code
        * @param NA
        * @return Map [Key => String [Code] , Value => String [Error Message]]
        */
      /// include mapping of all  codes based on gateway you are implementing
     public static map<string, string> getGatewayReturnCode() {
        Map<string, string> mapOfMessageByErrorCode = new Map<string, string>();
```

```
        mapOfMessageByErrorCode.put('100', 'Successful transaction');
        mapOfMessageByErrorCode.put('101', 'The request is missing one or more required
fields.
           Possible action: see the reply fields missingField_0...N for which fields are
missing. Resend the request with the complete information. For information about missing
or invalid fields');
        mapOfMessageByErrorCode.put('102', 'Invalid data');
        mapOfMessageByErrorCode.put('151', 'This error does not include timeouts between
the client and the server.
           To avoid duplicating the transaction, do not resend the request until you have
 reviewed the transaction status at the Business Center');
        mapOfMessageByErrorCode.put('152', 'The request was received, but a service did
not finish running in time.
           To avoid duplicating the transaction, do not resend the request until you have
 reviewed the transaction status at the Business Center');
        mapOfMessageByErrorCode.put('201', 'The issuing bank has questions about the
request.
          You will not receive an authorization code programmatically, but you can obtain
 one verbally by calling the processor');
      /// include mapping of all error codes based on gateway you are implementing
        return mapOfMessageByErrorCode;
    }


    /**
     * @name getElements
     * @description Populates's map Of Response Value By Key
     * @param Dom XML Node
     * @return Map [Key => String [Name] , Value => String [text]]
     * @exception NA
     */
    public static Map<string, string> getElements(DOM.XMLNode node) {
        if (node.getNodeType() == DOM.XMLNodeType.ELEMENT) {
            if (String.IsNotBlank(node.getText().trim())) {
                mapOfResponseValueByKey.put(node.getName(), node.getText().trim());
            }

            for (Dom.XMLNode child : node.getChildElements()) {
                getElements(child);
            }
        }
        return mapOfResponseValueByKey;
    }


    public void generateToken(List<blng.TransactionParameter> listOfTransactionParameter)

    {
        //Frame xml/json for generate token api method
    }


    // Example from Cybersource to generate a voidToken Request
   public void generateVoidToken(List<blng.TransactionParameter> listOfTransactionParameter)
```

```
    {
        //Frame xml/json for void token api method
         XmlStreamWriter writer = new XmlStreamWriter();
        // Populate Document start
        writer.writeStartDocument('utf-8', '1.0');
        // Populate Envelope start
        writer.writeStartElement('s', 'Envelope',
'http://schemas.xmlsoap.org/soap/envelope/');
        writer.writeAttribute('xmlns', 'http://schemas.xmlsoap.org/soap/envelope/', 's',
'http://schemas.xmlsoap.org/soap/envelope/');
        // Populate Header start
       writer.writeStartElement('s', 'Header', 'http://schemas.xmlsoap.org/soap/envelope/');

        // Populate Security start
        writer.writeStartElement('wsse', 'Security',
'http://schemas.xmlsoap.org/soap/envelope/');
        writer.writeAttribute('xmlns',
'http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd', 'wsse',
 'http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd');
        // Populate UsernameToken start
        writer.writeStartElement('wsse', 'UsernameToken', '');
        // Populate the Username start
        writer.writeStartElement('wsse', 'Username', '');
        writer.writeCharacters(transactionParameterToProcess.getGateWay().MerchantId__c);

        writer.writeEndElement();
        // Populate the Username end
        // Populate the Password start
        writer.writeStartElement('wsse', 'Password', '');
        writer.writeAttribute(NULL, NULL, 'Type',
'http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0#PasswordText');


writer.writeCharacters(transactionParameterToProcess.getGateWay().TransactionSecurityKey__c);

        writer.writeEndElement();
        // Populate the Password end
        writer.writeEndElement();
        // Populate UsernameToken end
        writer.writeEndElement();
        // Populate Security end
        writer.writeEndElement();
        // Populate Header end
        // Populate Body start
        writer.writeStartElement('s', 'Body', '');
        // Populate RequestMessage start
        writer.writeStartElement('', 'requestMessage', '');
        writer.writeAttribute('', '', 'xmlns',
'urn:schemas-cybersource-com:transaction-data-1.137');
        // Populate merchantID start
        writer.writeStartElement('', 'merchantID', '');
        writer.writeCharacters(transactionParameterToProcess.getGateWay().MerchantId__c);
```

```
        writer.writeEndElement();
        // Populate merchantID end
        // Populate merchantReferenceCode start
        writer.writeStartElement('', 'merchantReferenceCode', '');

writer.writeCharacters(transactionParameterToProcess.getGateWay().MerchantReference__c);
        writer.writeEndElement();
        // Populate merchantReferenceCode end
        // Populate recurringSubscriptionInfo start
        writer.writeStartElement('', 'recurringSubscriptionInfo', '');
        // Populate subscriptionID start
        writer.writeStartElement('', 'subscriptionID', '');
        //Added by DL,W-3931366, for cardmapping

writer.writeCharacters(getCardMapping(transactionParameterToProcess.getPaymentMethod()));

        writer.writeEndElement();
        // Populate subscriptionID end
        writer.writeEndElement();
        // Populate recurringSubscriptionInfo end
        // Populate paySubscriptionDeleteService start
        writer.writeStartElement('', 'paySubscriptionDeleteService', '');
        writer.writeAttribute('', '', 'run', 'true');
        writer.writeEndElement();
        // Populate paySubscriptionDeleteService end
        // Populate RequestMessage end
        writer.writeEndElement();
        // Populate Body end
        writer.writeEndElement();
        // Populate Envelope end
        writer.writeEndDocument();
        transactionParameterToProcess.setRequestBody(writer.getXmlString());
        writer.close();
    }

    public void generateAuthorize(Map<String, blng.TransactionParameter>
mapOfTransactionParameterByInvoiceId)
    {
        //Frame xml/json for generate authorize api method
    }

    public void generateCapture(Map<String, blng.TransactionParameter>
mapOfTransactionParameterByInvoiceId)
    {
        //Frame xml/json for generate capture api method
    }

    public void generateCharge(Map<String, blng.TransactionParameter>
mapOfTransactionParameterByInvoiceId)
    {
        //Frame xml/json for generate charge api method
    }

    public void generateVoid(Map<String, blng.TransactionParameter>
```

```
mapOfTransactionParameterByInvoiceId)
    {
        //Frame xml/json for generate void api method
    }

    public void generateRefund(Map<String, blng.TransactionParameter>
mapOfTransactionParameterByInvoiceId)
    {
        //Frame xml/json for generate refund api method
    }

     public void generatenonReferencedRefund(Map<String, blng.TransactionParameter>
mapOfTransactionParameterByInvoiceId)
    {
        //Frame xml/json for generate non referenced refund api method
    }

    public void generateVoidRefund(Map<String, blng.TransactionParameter>
mapOfTransactionParameterByInvoiceId)
    {
        //Frame xml/json for generate void refund api method
    }

    public void generateGetPaymentStatus(List<blng.TransactionParameter>
listOfTransactionParameter)
    {
        //Frame xml/json for generate get payment status api method
    }

    public void generateGetRefundStatus(List<blng.TransactionParameter>
listOfTransactionParameter)
    {
        //Frame xml/json for generate get refund status api method
    }
}
```

YourHttpService

YourHttpService is an example of common HTTP utilities that might be required while building a partner package. All user gateway API classes will be different based on your needs. However, we've provided a template that you can use to help create your own.

EDITIONS

Available in: All Salesforce Billing Editions

```
public with sharing class YourHttpService {
    // ===========================================================================
    // CONSTANT
    // ===========================================================================

    // ===========================================================================
    // STATIC VARIABLES
    // ===========================================================================
```

```
private static YourHttpService httpServiceInstance;

// ================================================================================
// VARIABLES
// ================================================================================

private HttpResponse httpResponse;
private Map<String,String> mapOfHeaderParameter = new Map<String,String>();
private enum Method {GET, POST}

/**
* @name getInstance
* @description get an Instance of Service class
* @params NA
* @return Http Service Class Instance
* @remark
* @change
*/
public static YourHttpService getInstance()
{
    if (NULL == httpServiceInstance)
    {
        httpServiceInstance =  new YourHttpService();
    }
    return httpServiceInstance;
}

/**
* @name get
* @description Get Method to get a HTTP request
* @param endPoint
* @return
* @remark
* @change
*/
public void get(String endPoint)
{
    send(newRequest(Method.GET, endPoint));
}

/**
* @name post
* @description Post Method to Post a HTTP request
* @param endPoint and requestBody
* @return
* @exception
* @remark
* @change
*/
public void post(String endPoint, String requestBody)
{
    send(newRequest(Method.POST, endPoint, requestBody));
}
```

```
    /**
    * @name addHeader
    * @description addHeader Methods to add all the defualt Header's required fo rthe
request
    * @param name and value
    * @return
    * @exception
    * @remark
    * @change
    */
    public void addHeader(String name, String value)
    {
        mapOfHeaderParameter.put(name, value);
    }

    /**
    * @name setHeader
    * @description setHeader Methods to set setHeader for the request
    * @param request
    * @return
    */
    private void setHeader(HttpRequest request)
    {
        for(String headerValue : mapOfHeaderParameter.keySet())
        {
            request.setHeader(headerValue, mapOfHeaderParameter.get(headerValue));
        }
    }

    /**
    * @name setAuthorizationHeader
    * @description setAuthorizationHeader Methods to set Authorization Header for the
request
    * @param userName and password
    * @return
    */
    public void setAuthorizationHeader(String userName,String password)
    {
        if(String.isNotBlank(userName) && String.isNotBlank(password))
        {
            Blob headerValue = blob.valueOf(userName + ':' + password);
           String authorizationHeader = 'Basic ' + EncodingUtil.base64Encode(headerValue);

            addHeader('Authorization', authorizationHeader);
        }
    }

    /**
    * @name setTokenisationHeader
   * @description setTokenisationHeader Methods to set Tokenisation Header for the request

    * @param userName and password
    * @return
    */
```

```
public void setTokenisationHeader(String userName,String password)
{
    if(String.isNotBlank(userName) && String.isNotBlank(password))
    {
        Blob headerValue = blob.valueOf(userName + ':' + password);
       String tokenisationHeader = 'Basic ' + EncodingUtil.base64Encode(headerValue);

        addHeader('Tokenisation', tokenisationHeader);
    }
}

/**
* @name newRequest
* @description newRequest Methods to make a new request
* @param method and endPoint
* @return newRequest
* @exception
*/
private HttpRequest newRequest(Method method, String endPoint)
{
    return newRequest(method, endPoint, NULL);
}

/**
* @name newRequest
* @description newRequest Methods to make a new request
* @param method, endPoint and requestBody
* @return request
* @exception
*/
private HttpRequest newRequest(Method method, String endPoint, String requestBody)
{
    HttpRequest request = new HttpRequest();
    request.setMethod(Method.name());
    setHeader(request);
    request.setEndpoint(endPoint);
    if (String.isNotBlank(requestBody))
    {
        request.setBody(requestBody);
    }
    request.setTimeout(120000);
    return request;
}

/**
* @name send
* @description send Methods to send a request
* @param request
* @return
* @exception Throws Exception
*/
private void send(HttpRequest request)
{
    try
```

```
        {
            httpResponse = new Http().send(request);
        }
        catch(Exception e)
        {
          throw e;
        }
    }

    /**
    * @name getResponse
    * @description getResponse Method to get the Response
    * @param NA
    * @return httpResponse
    * @exception
    */
    public HttpResponse getResponse()
    {
        return httpResponse;
    }

    /**
    * @name getResponseToString
    * @description getResponse Method to get the Response
    * @param NA
    * @return getResponse
    * @exception
    */
    public String getResponseToString()
    {
        return getResponse().toString();
    }

}
```

Salesforce Billing Tax Integration Developer Guide

Use Salesforce Billing tax integration API to enable communication between Salesforce Billing and external tax engines.

Get Started With Tax Integration API

Configuring settings in the Salesforce Billing package so that your org can communicate with an external tax engine.

Tax Integration Core Classes

Your tax engine references these classes to manage tax information with Salesforce Billing. They can't be modified.

Get Started With Tax Integration API

Configuring settings in the Salesforce Billing package so that your org can communicate with an external tax engine.

Required Tax Integration Fields

All tax integration records require several Salesforce Billing fields to communicate with tax engines, regardless of tax engine type. Certain tax engines also require fields not mentioned here.

EDITIONS

Available in: All Salesforce Billing Editions

Tax Integration Fields

If you need more fields to hold other information regarding your tax engine, add them on the tax integration object.

Label	API Name	Notes
Active	blng__Active__c	If this field is inactive, tax treatments won't list this integration as a possible value for their Tax Integration field.
Default	blng__Default__c	Salesforce Billing uses the default tax integration for all tax calculations in your Salesforce Billing org. You can have only one default tax integration at once.
Tax Engine	blng__TaxEngine__c	Declares the type of tax engine that Salesforce Billing uses for tax calculation. Standard and Avalara AvaTax are available by default. If you're using a different tax engine, add it as a picklist value.

Configure Tax Integrations in Salesforce Billing

Add custom settings and remote site settings for your tax engine in Salesforce Billing.

EDITIONS

Available in: All Salesforce Billing Editions

1. Create a custom setting for the class that Salesforce Billing calls and sends to your tax package.
   a. From Setup, in the Quick Find box, enter `Custom Settings`, and then select **Custom Settings**.
   b. Select **Tax Config**.
   c. Select **Manage**, and then select **New**.
   d. Enter the name of your tax engine.
   e. Enter the tax package class name that your tax engine will call for tax calculations. The class name should have the format `{prefix.classname}`.
2. Create a custom setting for the tax engine that Salesforce Billing calls and sends to your tax package.
   a. From Setup, in the Quick Find box, enter `Custom Settings`, and then select **Custom Settings**.

   b. Select **Billing Config**.
   c. Select **Manage**, and then select **New**.
   d. Enter the name of your tax engine.
   e. In the Category field, enter *Tax*.
   f. In the String Value field, enter the name of your tax engine. This value must match the Tax Engine field on the tax integration record for the tax integration that you're configuring. For example, if your tax integration has a value of Avalara Avatax, your String Value field must have the same value.

3. Configure remote site settings.
   a. From Setup, in the Quick Find box, enter *Remote Site Settings*, and then select **Remote Site Settings**.
   b. Select **New Remote Site**.
   c. Add the domain production URL for your tax engine.
   d. If needed, create another remote site for your tax engine's domain sandbox URL.

Tax Integration Core Classes

Your tax engine references these classes to manage tax information with Salesforce Billing. They can't be modified.

The following are the `blng` namespace classes required for integrating Salesforce Billing with an external tax engine.

TaxParameters Class

Contains a list of setters to populate the data needed for a tax calculation. The data is passed in the tax callout.

TaxResults Class

Contains a list of setter methods to populate the data returned from a tax callout. The data is passed back to Salesforce Billing from the tax connector.

TaxEngines Interface

Your tax engine implements this interface. It contains methods for calculating tax, clearing in-memory entities from the previous call, and clearing legal references from the previous call.

TaxParameters Class

Contains a list of setters to populate the data needed for a tax calculation. The data is passed in the tax callout.

Namespace

blng

TaxParameters Methods

TaxParameters Methods

The following are methods for `TaxParameters`.

setSObjectListofIds(listOfSobjectIds)
Sets the list of IDs for objects that require tax calculation.

setSObjectID(sObjectId)
Sets the ID of a single entity, if any.

setLegalEntityReferences(legalEntityReference)
Sets the legal entity.

setSObjectType(sObjectTypeInstance)
Sets the entity type to invoice, credit note, debit note, or order product.

setTaxMethod(taxMethod)
Sets the method to `POST` or `GET`.

setIsCommitOnGetTax(isCommitOnGetTax)
Indicates whether the `get` method needs a `POST` in the same call.

setSObjectListofIds(listOfSobjectIds)

Sets the list of IDs for objects that require tax calculation.

Signature

```
global static Void setSObjectListofIds(List<Id> listOfSobjectIds)
```

Parameters

listOfSobjectIds
Type: List<Id>

Return Value

Type: Void

setSObjectID(sObjectId)

Sets the ID of a single entity, if any.

Signature

```
global static void setSObjectID(Id sObjectId)
```

Parameters

sObjectId
Type: Id

Return Value

Type: void

`setLegalEntityReferences(legalEntityReference)`

Sets the legal entity.

Signature

```
global static void setLegalEntityReferences(String legalEntityReference)
```

Parameters

`legalEntityReference`
Type: String

Return Value

Type: void

`setSObjectType(sObjectTypeInstance)`

Sets the entity type to invoice, credit note, debit note, or order product.

Signature

```
global static void setSObjectType(SObjectType sObjectTypeInstance)
```

Parameters

`sObjectTypeInstance`
Type: SObjectType

Return Value

Type: void

`setTaxMethod(taxMethod)`

Sets the method to `POST` or `GET`.

Signature

```
global static void setTaxMethod(String taxMethod)
```

Parameters

`taxMethod`
Type: String

Return Value

Type: void

`setIsCommitOnGetTax(isCommitOnGetTax)`

Indicates whether the `get` method needs a `POST` in the same call.

Signature

```
global static Void setIsCommitOnGetTax(Boolean isCommitOnGetTax)
```

Parameters

`isCommitOnGetTax`
Type: Boolean

Return Value

Type: Void

TaxResults Class

Contains a list of setter methods to populate the data returned from a tax callout. The data is passed back to Salesforce Billing from the tax connector.

Namespace

blng

TaxResults Methods

The following are methods for `TaxResults`.

`setRate(rate)`

Sets the tax rate to the percentage specified by the parameter.

Signature

```
global static void setRate(Decimal rate)
```

Parameters

rate
Type: Decimal

Return Value

Type: void

setTax(tax)

Sets the tax amount.

Signature

```
global static void setTax(Decimal tax)
```

Parameters

tax
Type: Decimal

Assigns tax.

Return Value

Type: void

setIsSuccess(isSuccess)

Sets `isSuccess` to true if tax calculation was successful, otherwise sets isSuccess to false.

Signature

```
global static void setIsSuccess(Boolean isSuccess)
```

Parameters

isSuccess
Type: Boolean

Return Value

Type: void

setId(id)

Sets the ID of the entity where tax was calculated.

Signature

```
global static void setId(String id)
```

Parameters

id
Type: String

```
In case of GET Tax Call: Assign Invoiceline/Orderproduct Id
In case of Post/Cancel Tax Call: Assign Invoice Id
```

Return Value

Type: void

setSummary(summary)

If tax isn't calculated successfully, assign error message to the summary.

Signature

```
global static void setSummary(String summary)
```

Parameters

summary
Type: String

Return Value

Type: void

TaxEngines Interface

Your tax engine implements this interface. It contains methods for calculating tax, clearing in-memory entities from the previous call, and clearing legal references from the previous call.

Namespace

blng

TaxEngines Methods

TaxEngines Methods

The following are methods for `TaxEngines`.

resetSObjectListofIds()
Resets in-memory IDs (such as credit note IDs, debit note IDs, etc.) to null for the next run.

resetLegalEntityReferences()
Sets the in-memory list of legal entities to null for the next run.

calculateTax(taxParametersInstance)
Returns the calculated tax for the specified taxParametersInstance.

resetSObjectListofIds()

Resets in-memory IDs (such as credit note IDs, debit note IDs, etc.) to null for the next run.

Signature

```
global static Void resetSObjectListofIds()
```

Return Value

Type: Void

resetLegalEntityReferences()

Sets the in-memory list of legal entities to null for the next run.

Signature

```
global static Void resetLegalEntityReferences()
```

Return Value

Type: Void

calculateTax(taxParametersInstance)

Returns the calculated tax for the specified taxParametersInstance.

Signature

```
global static Map<Id,TaxResults> calculateTax(TaxParameters taxParametersInstance)
```

Parameters

taxParametersInstance
Type: TaxParameters

Return Value

Type: Map<Id,TaxResults>

Return Map<Id, blng.TaxResults>

Revenue Recognition Service Developer Guide

Salesforce Billing Revenue Recognition API lets you run revenue recognition for any Salesforce object in response to triggers, process builders, and REST API calls. The service uses Salesforce Billing revenue recognition rules, treatments, and distribution methods to create a revenue schedule and transaction hierarchy.

Get Started with the Revenue Recognition Service

Review the Revenue Recognition Service's input and output parameters, and general guidelines.

The Revenue Recognition service is helpful when you want to forecast revenue on objects other than invoice lines or order products. For example, you can forecast revenue on quote lines once they're approved, or on a contract after it's activated. You can access the service by calling the `blng.recognizeRevenue` method through process builders, workflow rules, REST API, and APEX triggers.

The service takes several input parameters and passes them to Salesforce Billing. You can define these parameters based on equivalent fields on your source object. For example, if you want to recognize revenue after a contract is activated, you could set the `startDate` parameter to the contract's Activated Date field, and the `revenueAmount` parameter to the List Amount field from the contract's originating quote.

If the service processes all parameters successfully, it returns parameters indicating the success and creates a revenue schedule with revenue transactions. If there are any errors, the service returns parameters for the number and types of errors encountered.

Tip: By default, the Revenue Recognition Service API doesn't return success messages after successfully creating a revenue schedule and its transcations. We recommend building automation to confirm that your revenue schedule and revenue transactions were created successfully.

Revenue Recognition Service Input Parameters

Name	Type	Required	Definition
`currencyIsoCode`	String	No	Required for multicurrency orgs. Defines the currency used in revenue schedules made from this process builder. If not defined in a single-currency org, the revenue recognition service uses the org's default currency.
`endDate`	Date	Required only for daily or monthly revenue recognition	The date when the revenue recognition period ends.

Name	Type	Required	Definition
`legalEntityId`	ID	No	Required for orgs that use legal entities. Used to determine the correct revenue recognition treatments to use for a revenue recognition rule. **Important:** If your org uses legal entities, and you're using revenue recognition API on an object that doesn't have a Legal Entity field, you must create a custom legal entity field and enter its value on your own, or look up the value from a related record.
`revenueAmount`	Decimal	Yes	The amount that the revenue recognition service uses to determine available, deferred, recognized, and total revenue based on revenue recognition rules. Users must ensure that the amount value they pass to revenue recognition service has the currency as the `currencyIsoCode` field.
`revenueRecognitionRuleId`	ID	Yes	The service applies this revenue recognition rule, then applies the rule's appropriate revenue recognition treatment.
`source`	ID	Yes	ID of the entity that the revenue recognition service evaluates for revenue recognition or forecasting. When you configure your revenue recognition service setup, create a lookup field on the revenue schedule object. This field stores the ID of the record where you're recognizing revenue. When you call the revenue recognition service, pass the source record's ID to the lookup field.

Name	Type	Required	Definition
`sourceFieldName`	String	Yes	ID of the source record that's displayed in the revenue schedule.
`startDate`	Date	Yes	The date that the revenue recognition period begins. When the revenue distribution method is set to full recognition, the revenue recognition service uses the start date for full recognition and ignores the end date.

Revenue Recognition Service Output Parameters

Name	Type	Definition
`jobId`	ID	ID of the APEX job that passed the input parameters to the service.
`inputValidationStatus`	String	A list of input parameters that encountered validation errors.
`totalNumberOfInputsWithErrors`	Integer	The number of input parameters that the service couldn't use due to errors (for example, a missing startDate, or a revenueAmount with a non-numeric character).
`totalNumberOfInputsWithoutErrors`	Integer	The number of input parameters that were successfully passed to the service.

Revenue Recognition Service Setup

Prepare the Revenue Recognition Service by configuring several revenue fields in Salesforce Billing.

Your admins may already have configured objects and fields in your org for use with the revenue recognition service. If they haven't, read the following information.

To successfully pass a revenue recognition rule to the revenue recognition service, the rule's revenue distribution methods and revenue recognition treatments require several picklist fields to have a value of *`Other`*. This value indicates that Salesforce Billing should override the default revenue recognition process and use parameters passed to the APEX service instead. When you're configuring a revenue recognition rule for use with the service, make sure that the following fields have a value of *`Other`*.

Revenue Distribution Method

Full Recognition Date

Revenue Term End Date (Needed only for daily or monthly recognition)

Revenue Term Start Date

Type

Revenue Recognition Treatment

Revenue Schedule Creation Action

Revenue Schedule Amount

Note: If you're upgrading to Salesforce Billing Spring '20 from an earlier version, add `Other` as a picklist value to each of these fields before using the service.

Revenue Recognition Service Use Cases

Use process builders, APEX triggers, or REST API to call the Revenue Recognition Service.

Using the Revenue Recognition Service With Process Builders

Process builders provide a convenient way to call the revenue recognition service. After you've defined your Revenue Recognition class, your process builder can call the service while passing it key revenue information from the object where you want to recognize revenue.

Calling the Revenue Recognition Service with APEX Triggers

You can call the Revenue Recognition service with a custom APEX trigger.

Revenue Recognition REST API

Use REST API to call invocable methods from the Revenue Recognition Apex class.

Using the Revenue Recognition Service With Process Builders

Process builders provide a convenient way to call the revenue recognition service. After you've defined your Revenue Recognition class, your process builder can call the service while passing it key revenue information from the object where you want to recognize revenue.

When you use a process builder to trigger the revenue recognition process, configure your conditions carefully so you don't accidentally create multiple revenue schedules at once for the same record. For example, if you want to recognize revenue when a contract's status changes to Activated and your trigger condition checks only that the payment's status equals Posted, any value change on an activated contract will trigger your process. Instead, use conditions that trigger the process only when a contract's status *changes* to Activated.

Your process builder calls the `blng.recognizeRevenue` APEX method, which accepts the input parameters defined in Revenue Recognition Service Developer Guide. Take Remember, your RevenueRecognitionInputs class must at least define a Revenue Amount, Revenue Recognition Rule ID, Start Date, ID Source, and Source Field name. Your builder's Apex variables have to pass at least these five variables to successfully invoke the Revenue Recognition class, and any optional variables that you need.

We recommend defining Apex variables that inherit the value of an equivalent field. When you map a field in the process builder, we recommend using a formula field such as `VALUE(TEXT(Object.[Field]))`. Here are a few examples.

Note: Currently, currency fields aren't supported as invocable variables, so we use decimals to define the revenueAmount.

Object	Apex Variable Type	Object Field	Apex Variable Value
Quote Line	Formula	SBQQ__ListTotal__c	VALUE(Text(SBQQ__QuoteLine__c.SBQQ__ListTotal__c
Order	Formula	TotalPrice	VALUE(Text([OrderItem].TotalPrice))
Contract	Formula	Amount	VALUE(Text(Contract__c.Amount))

Here are a few ways you can define a value for Start Date.

Object	Apex Variable Type	Object Field	Apex Variable Value
Quote Line	Date	SBQQ__Start__Date__c	`[SBQQ__QuoteLine__c].SBQQ__StartDate__c`
Order	Date	Effective-Date	`[OrderItem].Order.Effective-Date`
Contract	Date	ActivatedDate	`Contract.ActivatedDate`

Example: In this example, we want a process builder that recognizes quote line revenue after a quote is approved.

Criteria

Executes when any of the conditions are met.

Conditions

Field	Operator	Type	Picklist
[illegible]	Equals	Picklist	Approved

APEX Variables

Apex Class: Recognize Revenue

Field	Type	Value
startDate	Date	`[SBQQ__QuoteLine__c].SBQQ__StartDate__c`
sourceFieldName	String	`SBQQ__Quote__Line__c`
source	Field Reference	`[SBQQ__QuoteLine__c].Id`
revenueAmount	Formula	`VALUE(text([SBQQ__QuoteLine__c].SBQQ__ListTotal__c))`
revenueRecognitionRuleId	Field Reference	[illegible]
endDate	Date	`[SBQQ__QuoteLine__c].SBQQ__EndDate__c`
LegalEntityID	Field Reference	`[SBQQ__QuoteLine__c].LegalEntity__c` Note: Use a custom quote line field called Legal Entity that looks up to the same legal entity that you plan to use for your order product.

Example: In this example, we want a process builder that forecasts revenue when a contract is activated. We've defined a custom `blng__Contract__c` field on the revenue schedule to store the contract's ID value for `sourceFieldName`. Instead of using a variable from the source entity for our end date, we've defined a static date.

Criteria

Executes when all of the conditions are met.

Conditions

Field	Operator	Type	Picklist
`Contract`	Is changed	Boolean	True
`Contract`	Equals	Picklist	Posted

APEX Variables

Apex Class: Recognize Revenue

Field	Type	Value
startDate	Date	`Contract.ActivatedDate`
sourceFieldName	String	`blng__Contract__c`
source	Field Reference	`Contract.Id`
revenueAmount	Formula	`VALUE(Text(Contract.blng__Amount__c))`
revenueRecognitionRuleId	Field Reference	`[SBQQ__Quote__c.SBQQ__Opportunity__c.blng__RevenueRecognitionRule__c]`
endDate	Date	`12/31/2019`

Example: In this example, we want a process builder that forecasts order product revenue after an order product is activated.

Criteria

Executes when all of the conditions are met.

Conditions

Field	Operator	Type	Picklist
`[OrderItem].SBQQ__Status__c`	Is changed	Boolean	True
`[OrderItem].SBQQ__Status__c`	Equals	Picklist	Activated

APEX Variables

Apex Class: Recognize Revenue

Field	Type	Value
startDate	Date	`[OrderItem].Order.Effective-Date`
sourceFieldName	String	`blng__OrderProduct__c`
source	Field Reference	`[OrderItem].Id`
revenueAmount	Formula	`[OrderItem].Id`
revenueRecognitionRuleId	Field Reference	`[OrderItem].Product2.blng__RevenueRecognitionRule__c`

Field	Type	Value
endDate	Date	*[OrderItem].Order.EndDate*

Calling the Revenue Recognition Service with APEX Triggers

You can call the Revenue Recognition service with a custom APEX trigger.

For a list of input and output parameters, review Revenue Recognition Service Developer Guide.

Example: In this example, we want to create a revenue schedule for an invoice line after its status changes to Posted. The org uses legal entities, so we've included the legal entity in an input. Make sure to always review field validations and mapping to ensure you can trigger your process without errors.

Important: Always review field validations and mapping to ensure you can trigger your process without any errors.

```
//In this example, we want to create a revenue schedule for an invoice line after its
 status changes to Posted.


trigger GenerateRevenueOnInvoiceLineActivation on InvoiceLine (before insert) {
    InvoiceLine[] newInvoiceLine = Trigger.new;
    InvoiceLine[] oldInvoiceLine = Trigger.old;

    RevenueRecognitionInput[] inputs = new List<RevenueRecognitionInput>();

    Integer i = 0;
    for (InvoiceLine newInvoiceLine : newInvoiceLine) {
        if (newInvoiceLine.blng__InvoiceLineStatus__c !=
oldInvoiceLine[i].blng__InvoiceLineStatus__c &&
            newInvoiceLine.blng__InvoiceLineStatus__c == 'Posted') {
            RevenueRecognitionInput input = new RevenueRecognitionInput();
            input.source = newInvoiceLine.ID;
            input.sourceFieldName = 'InvoiceLine';
            input.revenueAmount = newInvoiceLine.blng__Balance__c;
            input.startDate = newInvoiceLine.blng__StartDate__c;
            input.endDate = (Date.today()).addMonths(2);
            input.revenueRecognitionRuleId =
newInvoiceLine.blng__Product__r.blng__RevenueRecognitionRule__c;
            input.legalEntityId = newInvoiceLine.blng__LegalEntity__c;
            inputs.add(input);
        }
        i++;
    }

    if (inputs != null && inputs.size() > 0) {
        List<blng.RevenueRecognitionResponse> response =
blng.RevenueRecognition.recognizeRevenue(inputs);
    }
}
```

Revenue Recognition REST API

Use REST API to call invocable methods from the Revenue Recognition Apex class.

This object is available in API version 48.0 and later.

Supported REST HTTP Methods

URI

`actions/custom/apex/blng__RevenueRecognition`

Formats

JSON, XML

HTTP Methods

POST

Authentication

`Bearer Token`

Parameters

Input	Details
`currencyIsoCode`	**Type** String **Description** Required only for multicurrency orgs. Defines the currency used in revenue schedules made from this process builder. If not defined, in a single-currency org, the revenue recognition service uses the org's default currency.
`legalEntityID`	**Type** ID **Description** Required for orgs that use legal entities. Used to determine the correct revenue recognition treatments to use for a revenue recognition rule. Important: If your org uses legal entities, and you're using revenue recognition API on an object that doesn't have a Legal Entity field, you must create a custom legal entity field and enter its value on your own or look up the value from a related record.
`revenueAmount`	**Type** Decimal **Description** The amount that the revenue recognition service uses to determine available, deferred, recognized, and total revenue based on revenue recognition rules. Users must ensure that the amount value they pass to revenue recognition service has the currency as the currencyIsoCode field. Required.
`revenueRecognitionRuleId`	**Type** ID

Input	Details
	Description The service applies this revenue recognition rule, then applies the rule's appropriate revenue recognition treatment. Required.
`source`	**Type** ID **Description** ID of the entity that the revenue recognition service evaluates for revenue recognition or forecasting. Required.
`sourceFieldName`	**Type** String **Description** This field appears on the revenue schedule and stores a lookup to the source record's ID. Required.
`startDate`	**Type** Date **Description** The date that the revenue recognition period begins. When the revenue distribution method is set to full recognition, the revenue recognition service uses the start date for full recognition and ignores the end date. Required.
`endDate`	**Type** Date **Description** The date when the revenue recognition period ends. Required only for monthly or daily recognition.

Response Body

Name	Type	Definition
`jobId`	ID	The ID of the APEX job that sent your input parameters to the revenue recognition service.
`inputValidationStatus`	String	A list of input parameters that encountered validation errors.
`totalNumberOfInputsWithErrors`	Integer	The number of input parameters that the revenue recognition service couldn't use due to errors (for example, a missing startDate, or a revenueAmount with a non-numeric character).

Name	Type	Definition
totalNumberOfInputsWithoutErrors	Integer	The number of input parameters that were successfully passed to the revenue recognition service.

Samples

Sample Request Body

This payload passes a payment record to the Revenue Recognition Service. The service attempts to create a revenue schedule based off the revenue amount of $12000 and the defined revenue recognition rule.

```
{
  "inputs":[
    {
      "revenueAmount":"12000",
      "currencyIsoCode":"USD",
      "legalEntityId":"a1j5A000001P2ri",
      "revenueRecognitionRuleId":"a2A0x000000jM6REAU",
      "startDate":"2019-08-29",
      "source":"a2217000000G4ecAAC",
      "sourceFieldname":"blng__Contract__c"
     }
   ]
}
```

Sample Response Body

In this case, we receive a response with one error because the sample payload didn't include a `startDate` variable.

```
[
    {
        "actionName" : "blng__RecognizeRevenue",
        "errors" : null,
        "isSuccess" : true,
        "outputValues" : {
            "jobId" : "7070x00000s3Ut9AAE",
            "totalInputWithoutErrors" : "0",
            "totalInputWithErrors" : "1",
            "inputValidationStatus" : [
                [ "Queued"],
                [ "Start date is a required field"]
            ]
        }
    }
]
```

Hosted Card Payments Lightning Component

The `force:cardPayment` (Hosted Card Payments) component provides a user interface in a hosted payment page that uses Salesforce Billing to collect card payment information, make a payment, or perform both actions. Salesforce stores and processes the customer card information under PCI-compliant standards. You can host the payment page in an Experience Cloud site Page or Lightning Page.

EDITIONS

Available in: Salesforce Billing Winter '20 and later

To use this component, you must also have Salesforce Billing installed in your org. For more information, see Install Salesforce Billing in Salesforce Help.

The user interface contains a form that takes customer credit card and payment details, such as cardholder name and address, and a button for submitting them. You can customize the card component to hide fields, make fields required, and change the labels of fields and buttons. These customizations affect what admins see when they add the component to their Experience Cloud site or Lightning Pages, and what end customers see when they make a payment using your component. For example, in this component, we've defined the following customizations.

- The Expiration Month is required
- The Expiration Year is required
- The Email field is hidden
- The first address line has a label of "Street"
- The Pay Button's label has been changed to Save Card.

```
<force:CardPayment
      paymentProvider = "{!v.paymentProvider}"
      transactionType = "{!v.transactionType}"
      transactionParams = "{!v.transactionParams}"
      expiryMonthRequired = "true"
      expiryYearRequired = "true"
      hideEmail = "true"
      addressLine1Label = "Street"
      payButtonLabel="Save Card"/>
```

Once you've configured your component, you can embed it in a wrapper component that's available in Experience Builder. Here's a sample wrapper component with the cardPayment component embedded.

```
<aura:component>

    <aura:handler name="paymentTransactionCompleted"
              event="force:paymentTransactionCompleted"
              action="{!c.handleTransactionResponse}"/>

    <aura:attribute name="accountId" type="String" default="abcd"
        description="Account to which the transaction details are related"/>

    <aura:attribute name="gatewayId" type="String" default="1234"
        description="Gateway to be used for the transaction"/>

    <aura:attribute name="amount" type="Decimal" default="10"
        description="amount to be charged for the transaction"/>
```

```
    <aura:attribute name="paymentProvider" type="String" default='SalesforceBilling'
        description="Payments provider for this component"/>

    <aura:attribute name="transactionType" type="String" default='PaymentSale'
        description="Transaction to be done after submitting the button"/>

    <aura:attribute name="transactionParams" type="Object"
        description="Contains details about account id and gateway id to use"/>

    <aura:handler name="init" value="{!this}" action="{!c.doInit}"/>

    <force:cardPayment
          paymentProvider = "{!v.paymentProvider}"
          transactionType = "{!v.transactionType}"
          transactionParams = "{!v.transactionParams}"
          expiryMonthRequired = "true"
          expiryYearRequired = "true"
          hideEmail = "true"
          addressLine1Label = "Street"
          payButtonLabel="Save Card"/>
</aura:component>
```

You must handle the `force:paymentTransactionCompleted` event, which is fired after the transaction is completed.

```
      handleTransactionResponse : function(cmp, event, helper) {
        var response = event.getParam("response");
        if(response.isSuccess) {
            alert("isSuccess : " + response.isSuccess);
        }
    }
```

You can also add attributes to a design resource so that your admins have the option of hiding fields, making fields required, and changing labels. When your design resource is active, admins can select the payment component in Experience Builder to show a properties pane with checkboxes or text boxes for each of the attributes. Save your design resource as a `.design` file in the same directory as your wrapper component. For example, if you want to present your five customized attributes for your admins to select or deselect from the properties pane, add the following Design section.

```
<design:component>
    <design:attribute name="expiryMonthRequired" label="Expiry month is required"/>
    <design:attribute name="expiryYearRequired" label="Expiry year is required"/>

    <design:attribute name="hideSaveForFuture" label="Hide save for future"/>
    <design:attribute name="hideBillingAddressSection" label="Hide billing address
section"/>

    <design:attribute name="payButtonLabel" label="Pay button label"/>
</design:component>
```

Response Formats

Success

During a successful response, `isSuccess` becomes true. If `transactionType` had a value of PaymentSale, `salesforceResponse` response contains paymentId, paymentMethodId, and paymentTransactionId. If `transactionType` had a value of SavePaymentCard, `salesforceResponse` contains only paymentMethodId. The attributes of `gatewayResponse` varies based on the gateway. This example shows a Success response for a PaymentSale transaction and all possible attributes for `gatewayResponse`.

```
{
 "isSuccess" : true,
 "salesforceResponse" : {
  "paymentId" : "paymentId",
  "paymentMethodId" : "paymentMethodId",
  "paymentTransactionId" : "paymentTransactionId"
 },
 "gatewayResponse": {
  "message": "mes",
  "responseCodeMessage": "rcm",
  "responseMessage": "rm",
  "responseCode": "rc",
  "responseStatus": "rs"
 }
}
```

Failure in the gateway

```
{
 "isSuccess" : false,
 "salesforceResponse" : {
   "paymentTransactionId" : "paymentTransactionId"
 },
 "gatewayResponse": {
   "message": "mes",
   "responseCodeMessage": "rcm",
   "responseMessage": "rm",
   "responseCode": "rc",
   "responseStatus": "rs"
 }
}
```

Failure Internally

```
{
 "isSuccess": false,
 "errorDetails": {
   "message": "message"
  }
}
```

Security Considerations and PCI Compliance

When end users are done entering their credit card information, they click the payment button so that the component sends the information to the payment gateway. Based on how you've configured your component, the gateway either tokenizes the card information and sends it back to the component, or the gateway tokenizes the card information, processes it for payment, and sends the results to the component.

Salesforce Billing uses security guardrails to ensure that end users aren't charged more than once if they accidentally click the Pay Now button multiple times, or in the event of fraudulent activity. After the component sends a request to the payment gateway, it generates a nonce associated with that specific request. Salesforce Billing won't send another request unless the page is refreshed and the Hosted Card Payments component is re-rendered. The component validates the nonce before sending payment information to the gateway. If the nonce is valid, the component removes the nonce and sends the request to the gateway.

You can also use Javascript controllers to manage error handling and response handling. Here's a sample Javascript controller that uses the handler that we introduced earlier.

```
({
    doInit : function(component, event, helper) {
        var transactionParams = {};
        transactionParams.accountId = component.get("v.accountId");
        transactionParams.gatewayId = component.get("v.gatewayId");
        transactionParams.amount = component.get("v.amount");

        component.set("v.transactionParams", transactionParams);
    },

    handleTransactionResponse : function(cmp, event, helper) {
        var response = event.getParam("response");
        if(response.isSuccess) {
            alert("isSuccess : " + response.isSuccess);
        }
    }
})
```

Attributes for the Card Hosted Payments Component
Review the attributes for the Card Hosted Payments component.

Guidelines for the Hosted Card Payments Lightning Component
When you set up the wrapper and design for your hosted payment component, consider important guidelines.

HostedPaymentPageTransactionAPI Class
Global API Apex class for Salesforce Billing. Contains methods that customers can call to save a credit card with tokenization or make a transaction using a new credit card.

Attributes for the Card Hosted Payments Component

Review the attributes for the Card Hosted Payments component.

All attributes have a Default value of False and Access value of Global.

EDITIONS

Available in: Salesforce Billing Winter '20 and later

Name	Type	Description
paymentProvider	String	Defines the source for processing payments. The cardPaymentMethod component requires that this attribute has a value of SalesforceBilling.
transactionType	String	The action that the component performs when users select the component's button. Accepts values of SavePaymentCard or PaymentSale.
transactionParams	Object	Accepts values of amount, accountId, and gatewayId.
emailRequired	Boolean	Specifies whether the Email field is required.
cardHolderNameRequired	Boolean	Specifies whether the Card Holder Name field is required.
cardTypeRequired	Boolean	Specifies whether the Card Type field is required.
expiryMonthRequired	Boolean	Specifies whether the Expiration Month field is required.
expiryYearRequired	Boolean	Specifies whether the Expiration Year field is required.
cvvRequired	Boolean	Specifies whether the CVV field is required.
firstNameRequired	Boolean	Specifies whether the First Name field is required.
lastNameRequired	Boolean	Specifies whether the Last Name field is required.
addressLine1Required	Boolean	Specifies whether the Address Line 1 field is required.
addressLine2Required	Boolean	Specifies whether the Address Line 2 field is required.
cityRequired	Boolean	Specifies whether the City field is required.
stateRequired	Boolean	Specifies whether the State field is required.
countryRequired	Boolean	Specifies whether the Country field is required.
postalCodeRequired	Boolean	Specifies whether the Postal Code field is required.
hideCardHolderName	Boolean	Specifies whether the Cardholder Name field is hidden.

Name	Type	Description
hideCardType	Boolean	Specifies whether the Card Type field is hidden.
hideExpiryMonth	Boolean	Specifies whether the Expiration Month field is hidden.
hideExpiryYear	Boolean	Specifies whether the Expiration Year field is hidden.
hideCvv	Boolean	Specifies whether the CVV field is hidden.
hideSaveForFuture	Boolean	Specifies whether the Save for Future field is hidden.
hideAutoPay	Boolean	Specifies whether the AutoPay field is hidden.
hideFirstName	Boolean	Specifies whether the First Name field is hidden.
hideLastName	Boolean	Specifies whether the Last Name field is hidden.
hideEmail	Boolean	Specifies whether the Email field is hidden.
hideAddressLine1	Boolean	Specifies whether the Address Line 1 field is hidden.
hideAddressLine2	Boolean	Specifies whether the Address Line 2 field is hidden.
hideCity	Boolean	Specifies whether the City field is hidden.
hideState	Boolean	Specifies whether the State field is hidden.
hideCountry	Boolean	Specifies whether the Country field is hidden.
hidePostalCode	Boolean	Specifies whether the Postal Code field is hidden.
creditCardInformationLabel	Boolean	Label for the Credit Card Information section.
addressInformationLabel	Boolean	Label for the Address Information section.
payButtonLabel	String	Label for the Pay button.
cardHolderNameLabel	String	Label for the cardHolderName field.
cardTypeLabel	String	Label for the cardType field.
cardNumberLabel	String	Label for the cardNumber field.
expiryMonthLabel	String	Label for the expiryMonth field.
expiryYearLabel	String	Label for the expiryYear field.

Name	Type	Description
cvvLabel	String	Label for the CVV field.
saveForFutureLabel	String	Label for the saveForFuture field.
autoPayLabel	String	Label for the AutoPay field.
firstNameLabel	String	Label for the firstName field.
lastNameLAbel	String	Label for the lastName field.
emailLabel	String	Label for the email field.
addressLine1Label	String	Label for the addressLine1 field.
addressLine2Label	String	Label for the addressLine2 field.
cityLabel	String	Label for the city field.
stateLabel	String	Label for the state field.
countryLabel	String	Label for the country field.
postalCodeLabel	String	Label for the postalCode field.
hideAddressSection	String	Specifies whether the address section is hidden.
clearFormData	String	Clears the component's form data.

Guidelines for the Hosted Card Payments Lightning Component

When you set up the wrapper and design for your hosted payment component, consider important guidelines.

EDITIONS

Available in: Salesforce Billing Winter '20 and later

- Customers can pay only against an account.
- In multicurrency orgs, customers can make payments only in the account's currency. In single-currency orgs, the component uses the org's currency. When handling currency, the component retrieves SavePaymentCard or PaymentSale's currency from the customer's account. It won't accept CurrencyIsoCode as an input parameter.
- Component field positions can't be changed.
- The component doesn't support custom fields.
- To prevent fraudulent activity, we recommend developing guardrails around the users that can access your payments page. We also recommend that your payments page is not the first page in your Experience Cloud site or Lightning page.

HostedPaymentPageTransactionAPI Class

Global API Apex class for Salesforce Billing. Contains methods that customers can call to save a credit card with tokenization or make a transaction using a new credit card.

Namespace

blng

Usage

The `HostedPaymentPageTransactionAPI` class contains the `chargeTransaction` method, which allows users to make a transaction using a new credit card, and the `savePaymentMethod` method, which allows users to save a credit card with tokenization.

Inputs

Table 7: Address Inputs

Name	Type	Description	Required or Optional	Available Version
`addressLine1`	String	First line in the address of the user making the payment.	Optional	48.0
`addressLine2`	String	Second line in the address of the user making the payment.	Optional	48.0
`city`	String	City	Optional	48.0
`country`	String	Country	Optional	48.0
`postalCode`	String	Postal code	Optional	48.0
`state`	String	State	Optional	48.0

Table 8: CardPaymentMethod Inputs

Name	Type	Description	Required or Optional	Available Version
`cardHolderName`	String	Full name of the card holder.	Required	48.0
`cardNumber`	String	Number of the credit card.	Required	48.0
`cardType`	String	Card network type. Valid values are: • `AmericanExpress` • `Discover` • `MasterCard` • `Visa`	Required	48.0
`cvv`	String	CVV	Required	48.0
`expiryMonth`	Integer	Card expiration month.	Required	48.0
`expiryYear`	Integer	Card expiration year.	Required	48.0

Table 9: PaymentMethod Inputs

Name	Type	Description	Required or Optional	Available Version
`address`	String	Address of the payment method holder.	Required	48.0
`autopay`	Boolean	Defines whether Salesforce Billing saves a record of the payment method to the customer's account for use in future payment runs. One account can have only one payment method with autopay enabled.	Optional	48.0
`email`	String	Email of the payment method holder.	Optional	48.0
`firstName`	String	First name of the payment method holder.	Required	48.0
`lastName`	String	Last name of the payment method holder.	Required	48.0
`cardPaymentMethod`	`cardPaymentMethod`	Card details for the payment method.	Required	48.0
`paymentType`	String	Type of payment method. Valid values are: • `Credit Card`	Required	48.0
`saveForFuture`	Boolean	Indicates whether Salesforce saves a record of the payment method for future use.	Optional	48.0

HostedPaymentPageTransactionAPI Methods

HostedPaymentPageTransactionAPI Methods

The following are methods for `HostedPaymentPageTransactionAPI`.

`chargeTransaction(chargeRequest)`

Use this method to charge the desired amount from the buyer's card.

Signature

```
global static String chargeTransaction(String chargeRequest)
```

Parameters

`chargeRequest`
Type: String

Represents a charge request made to the payment gateway.

Return Value

Type: String

Represents a charge transaction response.

Example

Example format for a charge request input.

```
{
"accountId": "0018A00000QcPDpQAN",
"gatewayId": "a1x8A000000OZG2",
"amount": 10,
"paymentMethod": {
    "firstName": "Muneer Ahmed",
    "lastName": "Shaik",
    "email": "mas@sf.com",
    "saveForFuture": true,
    "autoPay": true,
    "paymentType": "Credit Card",
    "address": {
        "addressLine1": "Address 1",
        "addressLine2": "Address 2",
        "city": "SF",
        "state": "CA",
        "country": "United States",
        "postalCode": "41111"
    },
    "cardPaymentMethod": {
        "cardHolderName": "Muneer Ahmed Shaik",
        "cardType": "Visa",
        "cardNumber": "4111111111111111",
        "cvv": "111",
        "expiryMonth": "2",
        "expiryYear": "2021"
    }
}
}
```

savePaymentMethod(paymentMethodRequest)

Use this method to save a payment method. Currently supports saving only credit cards.

Signature

```
global static String savePaymentMethod(String paymentMethodRequest)
```

Parameters

paymentMethodRequest
Type: String

Represents a request to save a payment method.

Return Value

Type: String

Represents a save payment method response.

Example

Example format for a save payment method request input.

```
{
 "accountId": "0018A00000QcPDpQAN",
 "gatewayId": "a1x8A000000OZG2",
 "paymentMethod": {
     "firstName": "Muneer Ahmed",
     "lastName": "Shaik",
     "email": "mas@sf.com",
     "paymentType": "Credit Card",
     "address": {
         "addressLine1": "Address 1",
         "addressLine2": "Address 2",
         "city": "SF",
         "state": "CA",
         "country": "United States",
         "postalCode": "41111"
     },
     "cardPaymentMethod": {
         "cardHolderName": "Muneer Ahmed Shaik",
         "cardType": "Visa",
         "cardNumber": "4111111111111111",
         "cvv": "111",
         "expiryMonth": "2",
         "expiryYear": "2021"
     }
  }
}
```

REST API for Converting Invoice Lines with Negative Balances

Use the REST API service `blng__NegativeInvoiceToCreditNoteAction` to evaluate invoices in bulk and create credit notes for each invoice containing invoice lines with negative balances.

This object is available in API version 48.0 and later.

Supported REST HTTP Methods

URI

`/services/data/xx.x/actions/custom/apex/blng__NegativeInvoiceToCreditNoteAction`

Formats

JSON, XML

HTTP Methods

POST

Authentication

`Authorization: Bearer token`

Parameters

Parameter	Description
`invoiceId`	**Type** ID **Description** The REST service evaluates this invoice for invoice lines with a negative balance. For each invoice, it creates a credit note containing one credit note line for each negative invoice line, unless an error occurs during credit note line creation. If even one of the `-ve` invoice line conversions errors out, all of them fail. No partial conversion can occur. Each credit note line has a balance that's the positive equivalent of the corresponding negative invoice line. The credit note lines are unallocated, so users must manually allocate them as needed.

Response Body

Parameter	Description
`invoiceId`	**Type** String **Description** The invoice that the REST service evaluated.
`isSuccess`	**Type** Boolean **Description** Indicates whether the invoice was successfully evaluated (`True`) or could not be evaluated (`False`).
`errorMessage`	**Type** String

Parameter	Description
	Description Error message indicating why the REST service couldn't convert any negative invoice lines on the provided invoice. For example, the invoice did not have any negative invoice lines.

Samples

Sample Request Body

This payload passes two invoice IDs to the negative invoice line conversion service.

```
{
"inputs":[
        {
            "invoiceId":"a1oS0000001NEixIAG"
        },
        {
            "invoiceId":"a1oS0000001NEixIAE"
        }
    ]
}
```

Sample Response Body

In this case, we receive a response with one error because the sample payload contained an invoice that didn't have any negative invoice lines.

```
[
    {
        "actionName" : "blng__NegativeInvoiceToCreditNoteAction",
        "errors" : null,
        "isSuccess" : true,
        "outputValues" : {
            "errorMessage" : "Null"
            "invoiceId" : "a1oS0000001NEixIAG"
            "isSuccess" : "true"
        }
    }

 {
        "actionName" : "blng__NegativeInvoiceToCreditNoteAction",
        "errors" : null,
        "isSuccess" : true,
        "outputValues" : {
            "errorMessage" : "The invoice does not have any negative invoice lines."
            "invoiceId" : "a1oS0000001NEixIAE"
            "isSuccess" : "false"
        }
    }

]
```

Set Up Standalone Orders with API

Manage your orders and order products in Salesforce and Salesforce Billing without first creating a CPQ quote. We call this type of order a standalone order. They're useful if you manage your quotes and opportunities in an external platform but plan to use Salesforce Billing for order management and billing.

Our standalone orders documentation contains required fields and setup instructions unique to standalone orders and order products. For concepts shared between standard orders and standalone orders, we provide links to the original CPQ and Billing topics.

EDITIONS

Available in: Salesforce Billing Winter '21 and later

You still need the Salesforce CPQ package installed to create standalone orders and order products. Although it doesn't use any Salesforce CPQ features or objects, Salesforce Billing runs internal validations against the CPQ package for several features and workflows.

Create your objects by executing the following series of REST API requests in a single POST request. All object creation requests use the following endpoint, replacing VERSION with your required API version, such as v48.0 or v49.0:

```
/services/data/v48.0/composite
```

Your JSON request must contain a compositeRequest class that defines both the order and the order item. Here's a generic example with null values for the minimum required fields. Configurations for certain billing and order product features may require additional fields and entities.

Example:

```
{
"allOrNone" : true,
"compositeRequest" : [{
  "method" : "POST",
  "url" : "/services/data/VERSION/sobjects/order",
  "referenceId" : "refOrder",
  "body" : {
      "Status" : "Draft" ,
      "EffectiveDate" : ""
      "Pricebook2Id" : "",
      "AccountId" : ""}
  },{
  "method" : "POST",
  "url" : "/services/data/VERSION/sobjects/OrderItem",
  "referenceId" : "",
  "body" : {
    "Quantity" : "",
    "Unitprice" : "",
    "ServiceDate" : "",
    "EndDate" : "",
    "SBQQ__ChargeType__c" : "",
    "blng__BillableUnitPrice__c" : "",
    "SBQQ__OrderedQuantity__c" : "",
    "SBQQ__Status__c" : "",
    "PricebookEntryId" : "",
    "orderId" : ""
    }
  }]
}
```

Required Configurations for Types of Standalone Order Products

Standalone orders require you to provide all the fields that would otherwise come from a CPQ quote. We provide a collection of topics showing all the required fields and their values for standalone order products used in different types of billing features. Certain fields require a specific value, while some are required but vary based on your specific configuration.

EDITIONS

Available in: Salesforce Billing Winter '21 and later

Requirements for Standalone One-Time Order Products

Create a standalone one-time order product through API in Salesforce Lightning or Salesforce Classic.

EDITIONS

Available in: Salesforce Billing Winter '21 and later

Required Fields

Table 10: Product

Field	Value
Billing Rule	[Required]
Charge Type	One-Time
Revenue Recognition Rule	[Required]

Field	Value
Tax Rule	[Required]

Table 11: Order

Field	Value
Account	[Required]
Effective Date	[Required]
Price Book	[Required]
Status	Draft

Field	Value
Billable Unit Price	Required for invoice runs or Bill Now to pick up the order product for invoicing. We recommend calculating a value based on the formula described in Recurring Billing and Billable Unit Price.
Billing Rule	[Get value from product]
Billing Type	[Required]
Charge Type	One-Time
Charge Type	[Required]
End Date	[Required]
Order	[Required]
Ordered Quantity	[Required]
Price Book Entry	[Required]
Quantity	[Required]
Reference ID	[Required]
Revenue Recognition Rule	[Get value from product]
Service Date	[Required]
Status	Draft
Tax Rule	[Get value from product]

Example:

```
{
"allOrNone" : true,
"compositeRequest" : [{
```

```
  "method" : "POST",
  "url" : "/services/data/v48.0/sobjects/order",
  "referenceId" : "refOrder",
  "body" : {
      "Status" : "Draft" ,
      "EffectiveDate" : "2020-06-16",
      "Pricebook2Id" : "Pricebook records ID",
      "AccountId" : "Account records ID"}
  },{
  "method" : "POST",
  "url" : "/services/data/v48.0/sobjects/OrderItem",
  "referenceId" : "refOrderItem",
  "body" : {
    "Quantity" : "1",
    "Unitprice" : "10",
    "ServiceDate" : "2020-06-16",
    "EndDate" : "2020-06-16",
    "SBQQ__ChargeType__c" : "One-Time",
    "blng__BillableUnitPrice__c" : "10",
    "SBQQ__OrderedQuantity__c" : "1",
    "SBQQ__Status__c" : "Draft",
    "PricebookEntryId" : "PricebookEntry records ID",
    "orderId" : "@{refOrder.id}"
    }
  }]
}
```

Requirements for Standalone Termed Recurring Order Products

Create a termed recurring order product through API in Salesforce Lightning or Salesforce Classic.

EDITIONS

Available in: Salesforce Billing Winter '21 and later

Required Fields

Table 12: Product

Field	Value
Billing Frequency	Monthly, Quarterly, Semiannual, or Annual
Billing Rule	[Required]
Charge Type	Recurring
Revenue Recognition Rule	[Required]
Tax Rule	[Required]

Table 13: Order

Field	Value
Account	[Required]
Effective Date	[Required]

Field	Value
Price Book	[Required]
Status	Draft

Table 14: Order Product

Field	Value
Billable Unit Price	Required for invoice runs or Bill Now to pick up the order product for invoicing. We recommend calculating a value based on the formula described in Recurring Billing and Billable Unit Price.
Billing Frequency	Monthly
Billing Rule	Get value from product
Billing Type	[Required]
Charge Type	Recurring
Charge Type	[Required]
Default Subscription Term	[Required]
End Date	[Required]
Order	[Required]
Ordered Quantity	[Required]
Price Book Entry	[Required]
Quantity	[Required]
Reference ID	[Required]
Revenue Recognition Rule	Get value from product
Service Date	[Required]
Status	Draft
Tax Rule	Get value from product

Example:

```
{
"allOrNone" : true,
"compositeRequest" : [{
  "method" : "POST",
  "url" : "/services/data/v48.0/sobjects/order",
  "referenceId" : "refOrder",
  "body" : {
      "Status" : "Draft" ,
      "EffectiveDate" : "2020-06-16",
```

```
      "Pricebook2Id" : "Price records ID",
      "AccountId" : "Account records ID"}
  },{
  "method" : "POST",
  "url" : "/services/data/v48.0/sobjects/OrderItem",
  "referenceId" : "refOrderItem",
  "body" : {
    "Quantity" : "1",
    "Unitprice" : "1200",
    "ServiceDate" : "2020-06-16",
    "EndDate" : "2021-06-15",
    "SBQQ__ChargeType__c" : "Recurring",
    "SBQQ__BillingType__c" : "Advance",
    "SBQQ__BillingFrequency__c" : "Monthly",
    "blng__BillableUnitPrice__c" : "100",
    "SBQQ__OrderedQuantity__c" : "1",
    "SBQQ__DefaultSubscriptionTerm__c" : "1",
    "SBQQ__Status__c" : "Draft",
    "SBQQ__ContractAction__c" : "New"
    "PricebookEntryId" : "PricebookEntry records ID",
    "orderId" : "@{refOrder.id}"
    }
  }]
}
```

Requirements for Standalone Evergreen Order Products

Create a monthly evergreen order product through API in Salesforce Lightning or Salesforce Classic.

EDITIONS

Available in: Salesforce Billing Winter '21 and later

Required Fields

Table 15: Product

Field	Value
Billing Frequency	Monthly, Quarterly, Semiannual, or Annual
Billing Rule	[Required]
Charge Type	Recurring
Revenue Recognition Rule	[Required]
Tax Rule	[Required]

Table 16: Order

Field	Value
Account	[Required]
Effective Date	[Required]
Price Book	[Required]

Field	Value
Status	Draft

Field	Value
Billable Unit Price	Required for invoice runs or Bill Now to pick up the order product for invoicing. We recommend calculating a value based on the formula described in Recurring Billing and Billable Unit Price.
Billing Frequency	Monthly
Billing Rule	Get value from product
Billing Type	[Required]
Charge Type	Recurring
Charge Type	[Required]
End Date	[Required]
Order	[Required]
Ordered Quantity	[Required]
Price Book Entry	[Required]
Product Subscription Type	Evergreen
Quantity	[Required]
Reference ID	[Required]
Revenue Recognition Rule	Get value from product
Service Date	[Required]
Status	Draft
Subscription Type	Evergreen
Tax Rule	Get value from product

Example:

```
{
"allOrNone" : true,
"compositeRequest" : [{
  "method" : "POST",
  "url" : "/services/data/v48.0/sobjects/order",
  "referenceId" : "refOrder",
  "body" : {
      "Status" : "Draft" ,
      "EffectiveDate" : "2020-06-16",
      "Pricebook2Id" : "Price records ID",
```

```
      "AccountId" : "Account records ID"}
  },{
  "method" : "POST",
  "url" : "/services/data/v48.0/sobjects/OrderItem",
  "referenceId" : "refOrderItem",
  "body" : {
    "Quantity" : "1",
    "Unitprice" : "50",
    "ServiceDate" : "2020-06-16",
    "SBQQ__ProductSubscriptionType__c" : "*Evergreen*",
    "SBQQ__SubscriptionType__c" : "*Evergreen*",
    "SBQQ__ChargeType__c" : "Recurring",
    "SBQQ__BillingType__c" : "Advance",
    "SBQQ__BillingFrequency__c" : "Monthly",
    "blng__BillableUnitPrice__c" : "50",
    "SBQQ__OrderedQuantity__c" : "1",
    "SBQQ__DefaultSubscriptionTerm__c" : "1",
    "SBQQ__Status__c" : "Draft",
    "SBQQ__ContractAction__c" : "New"
    "PricebookEntryId" : "PricebookEntry records ID",
    "orderId" : "@{refOrder.id}"
    }
  }]
}
```

Requirements for Standalone Invoice Plan Order Products

Create a standalone invoice plan order product through API in Salesforce Lightning or Salesforce Classic.

Salesforce Billing invoices standalone invoice plan order products only when the order product has a billing treatment with a New Order Invoice Plan value. You can configure your billing rules and legal entities so that Salesforce Billing assigns your order product a billing treatment. Or, you can create your order product, and then give it a billing treatment on your own.

EDITIONS

Available in: Salesforce Billing Winter '21 and later

Required Fields

Table 17: Product

Field	Value
Billing Frequency	Invoice Plan
Billing Rule	[Required]
Charge Type	Recurring
Revenue Recognition Rule	[Required]
Tax Rule	[Required]

Table 18: Order

Field	Value
Account	[Required]
Effective Date	[Required]
Price Book	[Required]
Status	Draft

Table 19: Order Product

Field	Value
Billable Unit Price	When Enable Billing Order Validations is active, Salesforce Billing sets this value to zero for order products created with a billing frequency of Invoice Plan.
Billing Frequency	Invoice Plan
Billing Rule	[Required]
Billing Rule	Get value from product
Billing Treatment	[Must be a billing treatment with an active invoice plan]
Charge Type	Recurring
End Date	[Required]
Price Book Entry	[Required]
Quantity	[Required]
Revenue Recognition Rule	[Required]
Revenue Recognition Rule	[Get value from product]
Service Date	[Required]
Status	Draft
Tax Rule	[Required]
Tax Rule	[Get value from product]
Unit Price	[Required]

Example:

```
{
"allOrNone" : true,
"compositeRequest" : [{
  "method" : "POST",
  "url" : "/services/data/v48.0/sobjects/order",
  "referenceId" : "refOrder",
```

```
  "body" : {
      "Status" : "Draft" ,
      "EffectiveDate" : "2020-06-16",
      "Pricebook2Id" : "Price records ID",
      "AccountId" : "Account records ID"}
  },{
  "method" : "POST",
  "url" : "/services/data/v48.0/sobjects/OrderItem",
  "referenceId" : "refOrderItem",
  "body" : {
    "Quantity" : "1",
    "Unitprice" : "50",
    "ServiceDate" : "2020-06-16",
    "EndDate" : "2021-06-15",
    "BillingTreatment__c" : "BillingTreatment records ID",
    "SBQQ__ChargeType__c" : "Recurring",
    "SBQQ__BillingType__c" : "Advance",
    "SBQQ__BillingFrequency__c" : "Monthly",
    "blng__BillableUnitPrice__c" : "50",
    "SBQQ__OrderedQuantity__c" : "1",
    "SBQQ__DefaultSubscriptionTerm__c" : "1",
    "SBQQ__Status__c" : "Draft",
    "SBQQ__ContractAction__c" : "New"
    "PricebookEntryId" : "PricebookEntry records ID",
    "orderId" : "@{refOrder.id}"
    }
  }]
}
```

Requirements for Standalone Amendment Order Products

Create a standalone amendment order product through API in Salesforce Lightning or Salesforce Classic

Make sure to provide a value for the revised order product ID. The Revised Order Product field shows the ID of the original order product used before any amendments were made. Remember, an order product can be related to multiple amendment order products.

EDITIONS

Available in: Salesforce Billing Winter '21 and later

Required Fields

Table 20: Product

Field	Value
Billing Frequency	Invoice Plan
Billing Rule	[Required]
Charge Type	Recurring
Revenue Recognition Rule	[Required]
Tax Rule	[Required]

Table 21: Order

Field	Value
Account	[Required]
Effective Date	[Required]
Price Book	[Required]
Status	Draft

Table 22: Order Product

Field	Value
Billable Unit Price	Required for invoice runs or Bill Now to pick up the order product for invoicing. We recommend calculating a value based on the formula described in Recurring Billing and Billable Unit Price.
Billing Frequency	[Required]
Billing Rule	[Get value from product]
Billing Type	[Required]
Charge Type	[Required]
Charge Type	[Required]
Default Subscription Term	[Required]
End Date	[Required]
Order	[Required]
Ordered Quantity	[Required]
Price Book Entry	[Required]
Quantity	[Required]
Reference ID	[Required]
Revenue Recognition Rule	[Get value from product]
Revised Order Product	[Required]
Service Date	[Required]
Status	Draft
Tax Rule	[Get value from product]

Requirements for Standalone Cancellation Order Products

Create a standalone cancellation order product through API in Salesforce Lightning or Salesforce Classic.

Salesforce Billing calculates canceled billings for a canceled standalone order product under the following settings.

EDITIONS

Available in: Salesforce Billing Winter '21 and later

- The cancellation order product has a Contract Action value of Cancel.
- The cancellation order product and related amendment order products all have a Revised Order Product field with the ID of the original order product used before any amendments or cancellations.
- The cancellation order products and related amendment order products all have the same terminated date. The Terminated Date must be the last field you populate on each order product before you activate the cancellation order product.

Note: You can cancel standalone order products under both legacy cancellation and LIFO cancellation.

Section Title

Table 23: Product

Field	Value
Billing Frequency	Invoice Plan
Billing Rule	[Required]
Charge Type	Recurring
Revenue Recognition Rule	[Required]
Tax Rule	[Required]

Table 24: Order

Field	Value
Account	[Required]
Effective Date	[Required]
Price Book	[Required]
Status	Draft

Field	Value
Contract Action	Cancel
Billing Frequency	[Required]
Charge Type	[Required]
Status	Draft
Reference ID	[Required]

Field	Value
Quantity	[Required]
Service Date	[Required]
End Date	[Required]
Price Book Entry	[Required]
Order	[Required]
Charge Type	[Required]
Billing Type	[Required]
Billable Unit Price	Required for invoice runs or Bill Now to pick up the order product for invoicing. We recommend calculating a value based on the formula described in Recurring Billing and Billable Unit Price.
Ordered Quantity	[Required]
Default Subscription Term	[Required]
Billing Rule	[Get value from product]
Revenue Recognition Rule	[Get value from product]
Tax Rule	[Get value from product]
Revised Order Product	[ID number for the original order product used before any amendments]
Terminated Date	[Set to the last day that the canceled order product will be billed]
Contract	[Required for LIFO Cancellation]

Example:

```
{
"allOrNone" : true,
"compositeRequest" : [{
  "method" : "POST",
  "url" : "/services/data/v48.0/sobjects/order",
  "referenceId" : "refOrder",
  "body" : {
      "Status" : "Draft" ,
      "EffectiveDate" : "2020-06-16",
      "Pricebook2Id" : "Price records ID",
      "AccountId" : "Account records ID"}
  },{
  "method" : "POST",
  "url" : "/services/data/v48.0/sobjects/OrderItem",
  "referenceId" : "refOrderItem",
  "body" : {
    "Quantity" : "1",
    "Unitprice" : "1200",
```

```
    "ServiceDate" : "2020-06-16",
    "EndDate" : "2021-06-15",
    "SBQQ__ChargeType__c" : "Recurring",
    "SBQQ__BillingType__c" : "Advance",
    "SBQQ__BillingFrequency__c" : "Monthly",
    "blng__BillableUnitPrice__c" : "100",
    "SBQQ__OrderedQuantity__c" : "1",
    "SBQQ__DefaultSubscriptionTerm__c" : "1",
    "SBQQ__Status__c" : "Draft",
    "PricebookEntryId" : "PricebookEntry records ID",
   "SBQQ__RevisedOrderProduct__c" : "RevisedOrderProduct(original orderproduct) records
 ID",
    "SBQQ__TerminatedDate__c" : "2020-08-15",
    "SBQQ__ContractAction__c" : "Cancel",
    "orderId" : "@{refOrder.id}"
    }
  }]
}
```

Requirements for Standalone Order Products with Consumption Schedules

Create a monthly order product with an order product consumption schedule and consumption rates through API in Salesforce Lightning or Salesforce Classic.

You must create the order, order product, order product consumption schedule, and at least one order product consumption rate. Your order product must be related to a product with a consumption schedule.

Salesforce Billing creates usage summaries when you activate an order product with the following configuration.

- The order product is related to a product with a consumption schedule.
- The order product has an active order product consumption schedule with at least one consumption rate.

EDITIONS

Available in: Salesforce Billing Winter '21 and later

Required Fields

Table 25: Product

Field	Value
Billing Frequency	Invoice Plan
Billing Rule	[Required]
Charge Type	Recurring
Revenue Recognition Rule	[Required]
Tax Rule	[Required]

Table 26: Order

Field	Value
Account	[Required]
Effective Date	[Required]
Price Book	[Required]
Status	Draft

Table 27: Order Product

Field	Value
Billable Unit Price	Required for invoice runs or Bill Now to pick up the order product for invoicing. We recommend calculating a value based on the formula described in Recurring Billing and Billable Unit Price.
Billing Frequency	[Required]
Billing Rule	[Get value from product]
Billing Type	[Required]
Charge Type	[Required]
Charge Type	[Required]
Default Subscription Term	[Required]
End Date	[Required]
Order	[Required]
Ordered Quantity	[Required]
Price Book Entry	[Required]
Quantity	[Required]
Reference ID	[Required]
Revenue Recognition Rule	[Get value from product]
Revised Order Product	[Required]
Service Date	[Required]
Status	Draft
Tax Rule	[Get value from product]

Table 28: Consumption Schedule

Field	Value
Billing Rule	[Required]

Field	Value
Billing Term	[Required]
Billing Term Unit	[Required]
Category	[Required]
Rating Method	[Required]
Revenue Recognition Rule	[Required]
Tax Rule	[Required]
Type	[Required]

Table 29: Consumption Rates

Field	Value
Lower Bound	[Required]
Price	[Required]
Pricing Method	[Required]
Processing Order	[Required]
Upper Bound	[Optional]

Table 30: Order Product Consumption Schedule

Field	Value
Billing Rule	[Required]
Billing Term	[Required]
Billing Term Unit	[Required]
Category	[Required]
Consumption Schedule	[Optional]
Order Product	[Required]
Rating Method	[Required]
Revenue Recognition Rule	[Required]
Tax Rule	[Required]
Type	[Required]

Field	Value
Lower Bound	[Required]

Field	Value
Order Product Consumption Schedule	[Required]
Price	[Required]
Pricing Method	[Required]
Processing Order	[Required]
Upper Bound	[Optional]

Example:

```
{
"allOrNone" : true,
"compositeRequest" : [{
  "method" : "POST",
  "url" : "/services/data/v48.0/sobjects/order",
  "referenceId" : "refOrder",
  "body" : {
      "Status" : "Draft" ,
      "EffectiveDate" : "2020-06-16",
      "Pricebook2Id" : "Price records ID",
      "AccountId" : "Account records ID"}
  },{
  "method" : "POST",
  "url" : "/services/data/v48.0/sobjects/OrderItem",
  "referenceId" : "refOrderItem",
  "body" : {
    "Quantity" : "1",
    "Unitprice" : "1200",
    "ServiceDate" : "2020-06-16",
    "EndDate" : "2021-06-15",
    "SBQQ__ChargeType__c" : "Recurring",
    "SBQQ__BillingType__c" : "Advance",
    "SBQQ__BillingFrequency__c" : "Monthly",
    "blng__BillableUnitPrice__c" : "100",
    "SBQQ__OrderedQuantity__c" : "1",
    "SBQQ__DefaultSubscriptionTerm__c" : "1",
    "SBQQ__Status__c" : "Draft",
    "PricebookEntryId" : "PricebookEntry records ID",
    "SBQQ__ContractAction__c" : "New",
    "orderId" : "@{refOrder.id}"
    }
  },
  {
  "method" : "POST",
  "url" : "/services/data/v48.0/sobjects/SBQQ__OrderItemConsumptionSchedule__c",
  "referenceId" : "refOrderItemConsumptionSchedule",
  "body" : {
    "SBQQ__RatingMethod__c" : "Tier",
    "SBQQ__ConsumptionSchedule__c" : "ConsumptionSchedule record ID",
    "SBQQ__BillingTermUnit__c" : "Month",
```

```
    "SBQQ__BillingTerm__c" : "1",
    "SBQQ__Type__c" : "Range/Slab",
    "SBQQ__Category__c" : "Rates",
    "blng__BillingRule__c" : "BillingRule record ID",
    "blng__RevenueRecognitionRule__c" : "RevenueRecognitionRule record ID",
    "blng__TaxRule__c" : "TaxRule record ID",
    "SBQQ__OrderItem__c" : "@{refOrderItem.id}"
    }
  },{
  "method" : "POST",
  "url" : "/services/data/v48.0/sobjects/SBQQ__OrderItemConsumptionRate__c",
  "referenceId" : "refOrderItemConsumptionRate",
  "body" : {
    "SBQQ__ProcessingOrder__c" : "1",
    "SBQQ__LowerBound__c" : "1",
    "SBQQ__UpperBound__c" : "100 or null",
    "SBQQ__Price__c" : "100",
    "SBQQ__PricingMethod__c" : "Per Unit/ Flat Fee",
    "SBQQ__OrderItemConsumptionSchedule__c" : "@{refOrderItemConsumptionSchedule.id}"

    }
  }]
}
```

Requirements for Standalone Order Products with Price Schedules

Create a standalone order product with price schedules through API in Salesforce Lightning or Salesforce Classic.

Important: Salesforce CPQ doesn't automatically create price schedules for standalone order products. You must create the price schedule and its price tiers on your own. Each price schedule requires at least one price tier.

EDITIONS

Available in: Salesforce Billing Winter '21 and later

Required Fields

Table 31: Product

Field	Value
Billing Frequency	Invoice Plan
Billing Rule	[Required]
Charge Type	Recurring
Revenue Recognition Rule	[Required]
Tax Rule	[Required]

Table 32: Order

Field	Value
Account	[Required]
Effective Date	[Required]
Price Book	[Required]
Status	Draft

Table 33: Order Product

Field	Value
Billing Frequency	[Required]
Charge Type	Usage
End Date	[Required]
Price Book Entry	[Required]
Price Schedule	[Required]
Quantity	[Required]
Service Date	[Required]
Unit Price	[Required]

Table 34: Price Schedule

Field	Value
Discount Unit	[Required]
Price Type	[Required]

Table 35: Price Tier

Field	Value
Lower Bound	[Required]
Optional	[Optional]
Price Model	[Required]
Price Schedule	[Required]

INDEX

B

C

E

I

N

S