Paidy payment processing

Version<22.1.0>

Table of contents

Details

Table of contents 2
1. Summary 4
2. Component summary 5
2-1. Features overview 5
2-2. Restrictions and limitations 5
2.3. Use cases 6
2-3-1. Standard payments 6
2-3-2. Subscription payments 9
2-4. Privacy when paying 13
3. Implementation guide 14
3-1. Uploading cartridges (UX Studio) 14
3-2. BusinessManager setup 14
3-2-1. Cartridge assignment 14
3-2-2. Importing metadata 15
3-3. Modifying cartridges on the merchant 20
3-3-1. CheckoutJS installation 20
3-3-2. Controlling purchase confirmation buttons 23
3-3-3. Modifying resource files 24
3-4. Modifying cartridges on the merchant 24
3-4-1. Modifying COBillingjs 24
3-4-2. COPlaceOrder.js modifications 26
3-4-3. COSummary.js modifications 28
3-4-4. Modifying to package.json on the JS controller cartridge at the merchant 29
3-4-5. Modifying package.json on the int_paidy_controllers cartridge 29
3-5. Advice 29
4. Cartridge deployment testing at merchant 31
4-1. Before beginning a test 31
4-1-1. Paidy merchant management screen 31
4-1-2.Commerce Cloud Business Manager 32
4-2.PaidyCheckout test data 32
4-3. Browser compatibility 33
4-4. Test cases 33
4-4-1. Paidy standard payments 33
4-4-2. Paidy subscription payments 34
5. Webhook usage examples 35
5-1. Webhook usage 35
5-2. Detecting inconsistent payments 35
5-3. Handling inconsistencies 36
5-3-1. Processing example 1 36
5-3-2. Processing example 2 39
5-4. Webhook settings 39
6. What to do when the service is down 40

7. Revision history 40

1. Summary

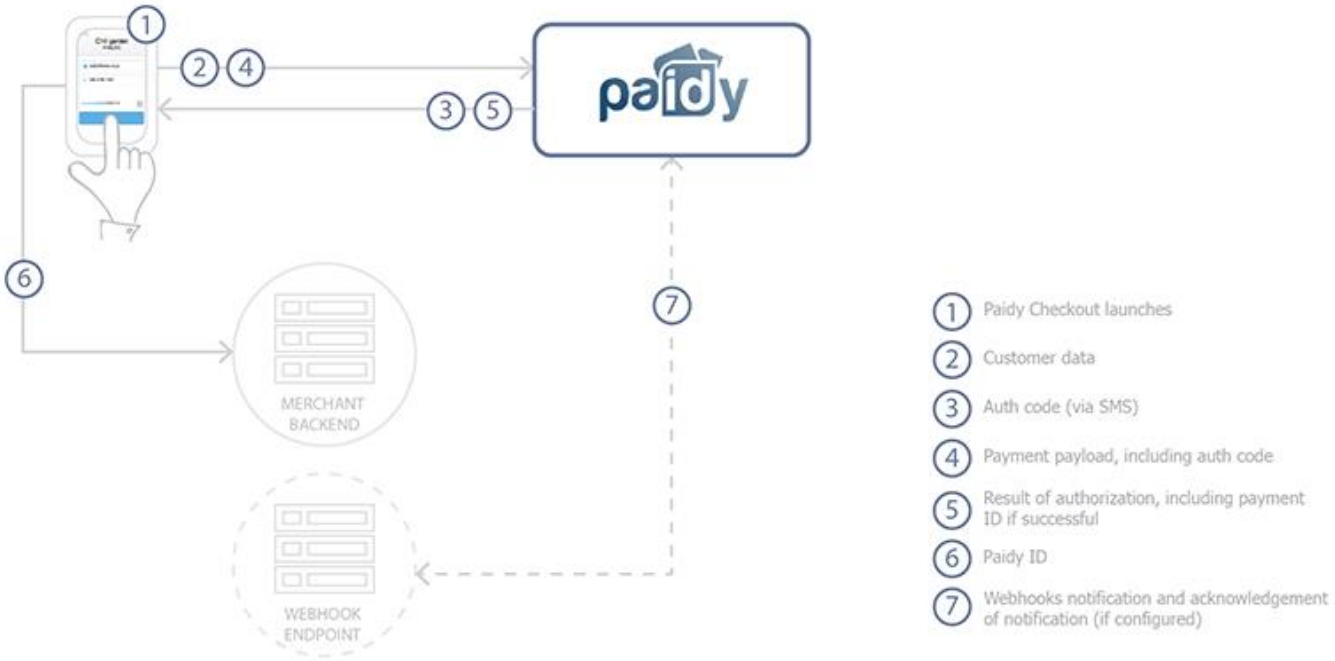
Cartridges allow a Commerce Cloud store to make use of Paidy payments.
The developer must follow this text to install the cartridges and set them up on the online store.
Because Paidy allows for creating test payments by making use of a test API key, the user can test the settings before switching over to a live environment. For details on data needed for testing, see 4-1. Before beginning a test.

Three cartridges must be implemented in order to begin using Paidy payments.
- int_paidy is the core cartridge that enables use of Paidy payments on the store.
- int_paidy_controllers is a cartridge that enables support of Paidy payments on a JSController site.

2. Component summary

2-1. Features overview

The process flow for Paidy is explained in the diagram below.
Paidy cartridges correspond to point 1-6 in this workflow.
In step 1-6, the results of authentication with Paidy are saved on the site.



2-2. Restrictions and limitations

- Only a “creation” process is in place for payment. Capture, Refund, Update, and Close must be implemented by the merchant.

- For first-time subscription payments, Commerce Cloud runs a creation process.
  Subsequent payments must be configured at the merchant, but the payment process can be implemented by executing the functions found in the Paidy cartridge.

- Subscription payments are designed to only work when the user is logged in.

- Confirm notifications from the Webhook on OMS; if the payment process results are incorrect, you can choose to set the order status to shippable in SFCC, or cancel the payment on Paidy. For details, please see 5. Webhook usage examples.

- This cartridge can only be used on Japanese language sites. (language: Japanese; currency: JPY)

2.3. Use cases

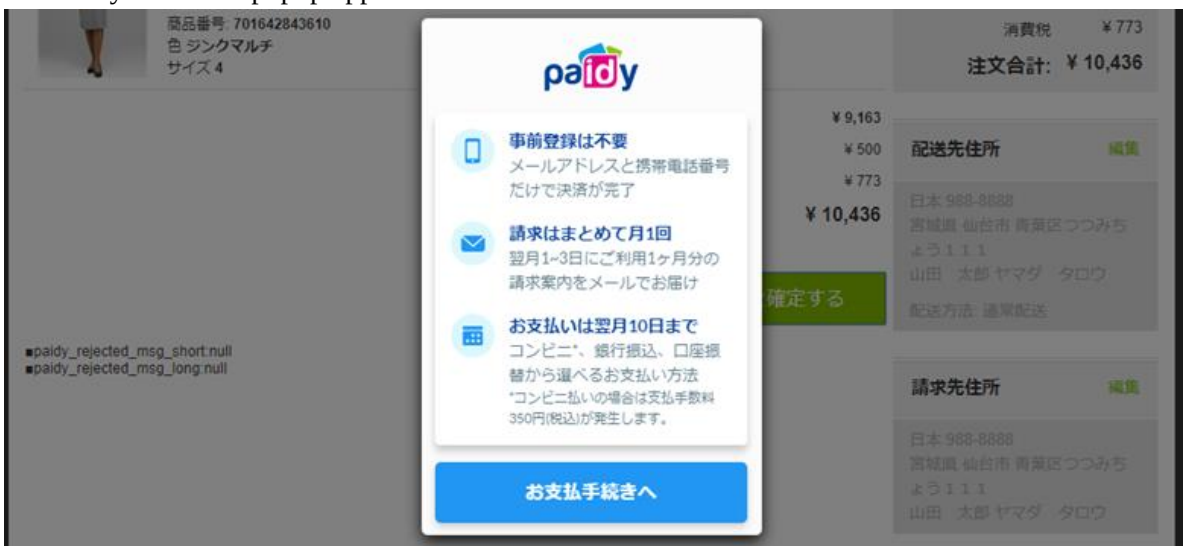
2-3-1. Standard payments

Paidy Checkout launches when a user clicks a payment purchase button on a site.
Consumers enter the requisite information and make a Paidy payment.
If the payment is successful, the process transitions to the product purchase confirmation screen.

- Payment flow

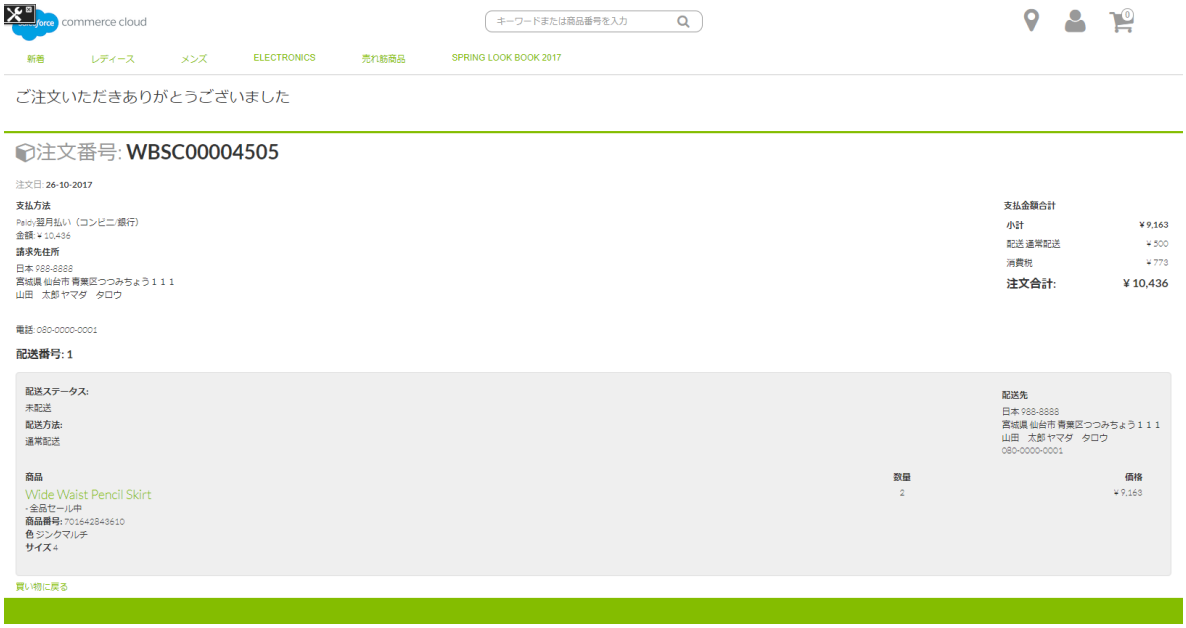
Select Paidy standard payments when selecting a payment method when checking out and proceed to confirming the order.

The Paidy Checkout popup appears.



The requisite information is filled in and Paidy payment is processed.

If the payment succeeds, the order is complete.



Order data created on Commerce Cloud

Payment data created on Paidy

If the payment fails, Paidy Checkout displays a payment error message and order failure data is created on Commerce Cloud.

Payment data will not be created on Paidy.

*Note that if the top-right X button is clicked on the popup, order failure data is created. Payment data will not be created on Paidy.

Order data created on Commerce Cloud (order failure)

2-3-2. Subscription payments

Only usable by logged in users.
When transitioning to a screen to confirm a purchase of products on the site, a check is performed to see if the consumer has a token.

When clicking on the order confirmation button for a first time purchase, there is no token, so Paidy Checkout launches. Consumers enter the requisite information and authenticate with Paidy. If the authentication completes correctly, a token is issued and is assigned to the consumer.

Once a token has been assigned or if the user has a token, the token data is used for Paidy payment.

- Payment flow

Select Paidy subscription payments when selecting a payment method when checking out and proceed to confirming the order.

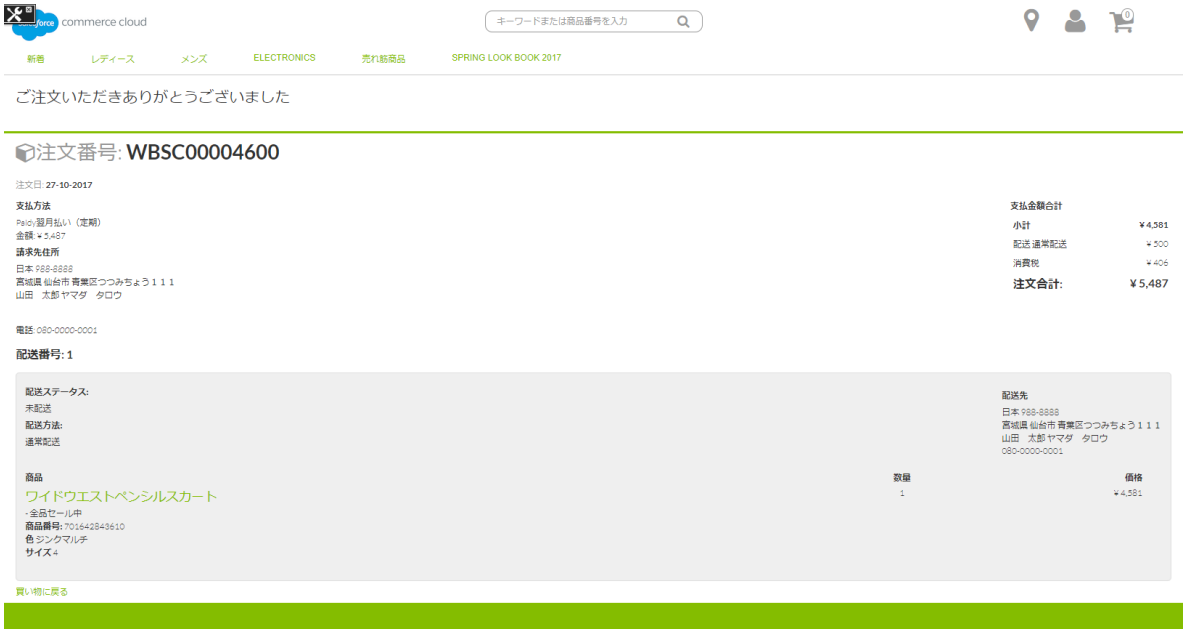
An order confirmation screen is displayed and the order is confirmed.

The Paidy Checkout popup appears (only the first time).

Enter the requisite information on Paidy Checkout and a token is issued.

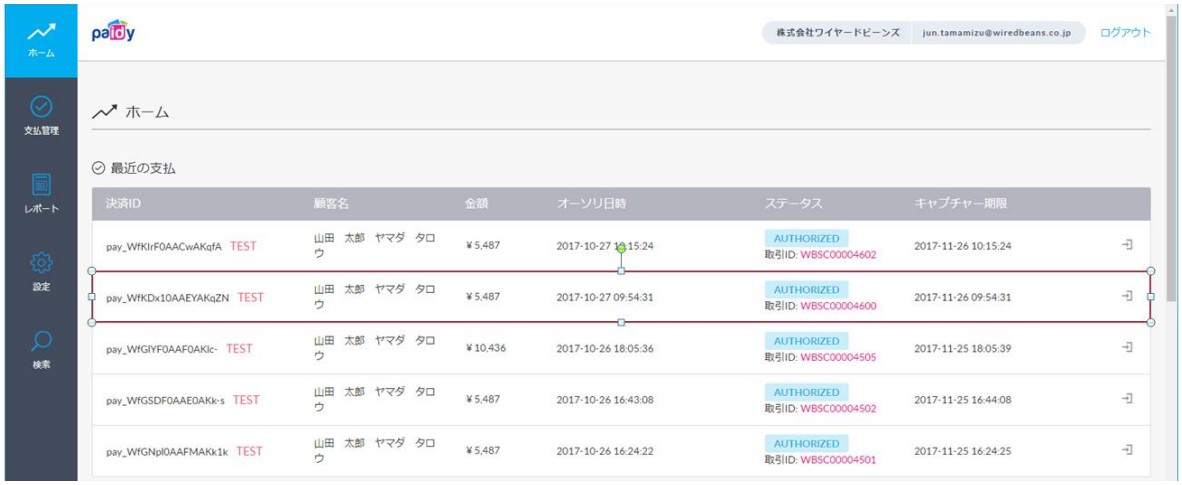
*If the popup is closed during the process by the user clicking the X button on the top right, the popup will no longer display, but a token can be reissued and the payment processed.

After issuing a token, the process moves to payment processing. If the payment succeeds, we transition to the order confirmation screen.



Order data created on Commerce Cloud

Payment data created on Paidy



If payment fails, a payment error will be displayed on the order confirmation page and order failure data is created on Commerce Cloud.

Payment data will not be created on Paidy.

Order data created on Commerce Cloud (order failure)

マーチャントツール > 注文 > 注文 > 注文: WBSC00004601(Paidy_Master)

全般 属性 支払 メモ 履歴

注文 'WBSC00004601' の詳細

情報:	1 件の配送先への 1 項目を含む。 合計金額は ¥ 5.487,00 です。
受取日:	10/27/17 1:06:16 AM Etc/UTC
サイト:	Paidy_Master
作成者:	顧客
顧客:	rio hongo
顧客ナンバー:	00027002
IP アドレス:	該当なし
Eメール:	rejected.payment@paidy.com
電話:	080-0000-0001

注文ステータス:	失敗	確認ステータス:	未確認
配送ステータス:	未配送	エクスポートステータス:	未エクスポート

配送

個数	商品 ID	名前	ステータス	Manufacturer (製造元)	税率	小売単価	税基準	項目合計
1	701642843610	ワイドウエストペンシルスカート 調整 全品セール中	CANCELLED		8,00 %	¥ 9.163,00	¥ 9.163,00	~~¥ 9.163,00~~ ~~¥ -4.563,00~~
							荷物配送料 : CANCELLED	~~¥ 500,00~~

© 2017 salesforce.com, inc. All Rights Reserved.

Paidy_Master タイムゾーン: 協定世界時 | インスタンスのタイムゾーン: 東部夏時間 | バージョン: 17.8 最終更新日時: 10月 18, 2017 (互換モード: 15.5)

配送合計:	¥ 500,00
税金合計:	¥ 406,00
合計:	¥ 5.487,00

Eメールの送信 注文の印刷

2-4. Privacy when paying

Paidy payment authentication requires entering a telephone number and e-mail address, but these are not saved on Commerce Cloud.
However, information entered upon registration and shipment and billing details are saved on Commerce Cloud.

3. Implementation guide

3-1. Uploading cartridges (UX Studio)

The cartridge is uploaded to Commerce Cloud.

- int_paidy
- int_paidy_controllers (JSController sites)

The Paidy cartridge can be used on SiteGenesis-based JSController sites.
Use Commerce Cloud UX-studio to upload the int_paidy and int_paidy_controllers cartridges to Commerce Cloud.

3-2. BusinessManager setup

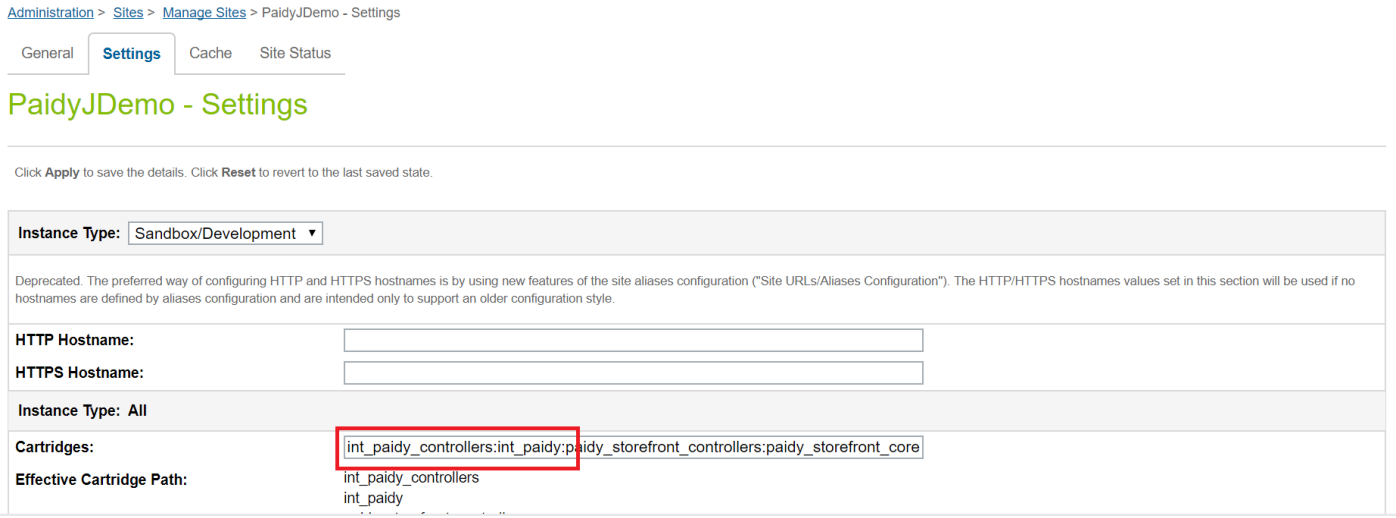
Paidy payment settings are configured on the Commerce Cloud management screen (Business Manager).

3-2-1. Cartridge assignment

A Paidy cartridge is assigned to the site.

Management > Sites > Site Management > Sites Under Management: Settings

- JSController sites
int_paidy_controllers:int_paidy: merchant cartridge



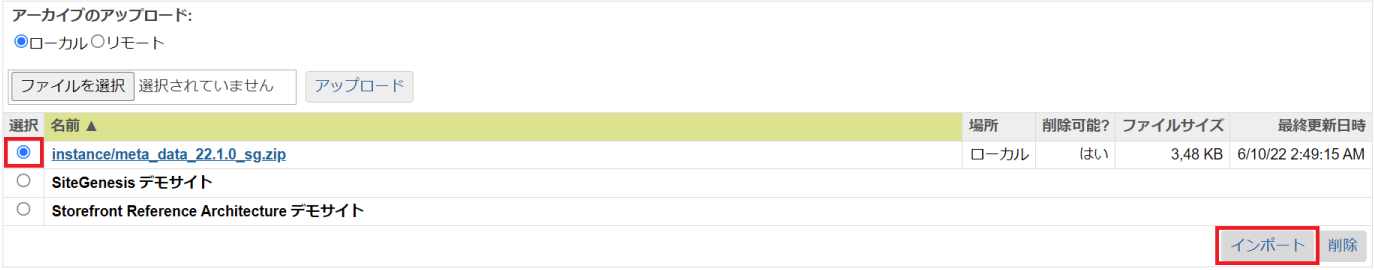
3-2-2. Importing metadata

meta_data_22.1.0_sg.zip is provided in the package.
By importing this zip file, you can add services, add payment methods / payment processors, change system objects and custom objects to use Paidy payment. Details will be described later.

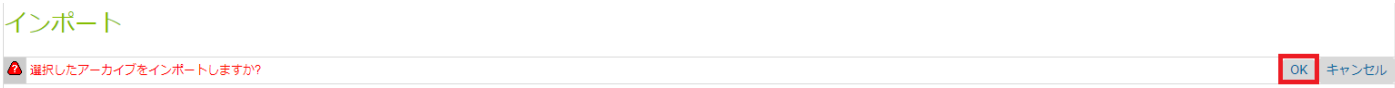
Administration > Site Development > Site Import & Export
Click [Choose File].

Select meta_data_22.1.0_sg.zip in the package and click the [Upload] button.

After uploading the zip file, check meta_data_22.1.0_sg.zip and click [Import].



Click [OK].



-Configuration when importing service metadata
Describes the service configuration when importing metadata.
Note that for services added after import
Please do not change the information described.

<Service Credentials>

Name: paidy.http.payment

URL: https://api.paidy.com

<Service Profiles>

Name: paidy.api.prof

<Services>

Name: paidy.api.payment

Type: HTTP

Enabled: check on

Service Mode: Live

Profile: paidy.api.prof

認証 Credentials: paidy.http.payment

· Payment method · Configuration when importing payment processor metadata

Describes the payment method and payment processor configuration when metadata is imported.
As a point of caution, for payment methods and payment processors added after import
Please do not change the information described.

<Payment Processors >
ID: PAIDY_STANDARD

<Payment Processors >
ID: PAIDY_SUBSCRIPTION

<Payment Methods >
ID: PAIDY_STANDARD

Name: Paidy 翌月払い（コンビニ/銀行）

Enabled: Yes

<Payment Methods >
ID：PAIDY_SUBSCRIPTION

Name: Paidy 翌月払い（コンビニ/銀行）※定期購入

Enabled: Yes

- Object layout for metadata importing
Here we include instructions on the layout of objects when importing metadata.
One word of caution is that system and custom objects added after importing should not be modified.

<Order>
Management > Site Development > System Object Types> Order - Attribute Definitions
System objects: Order is an object that corresponds to order data

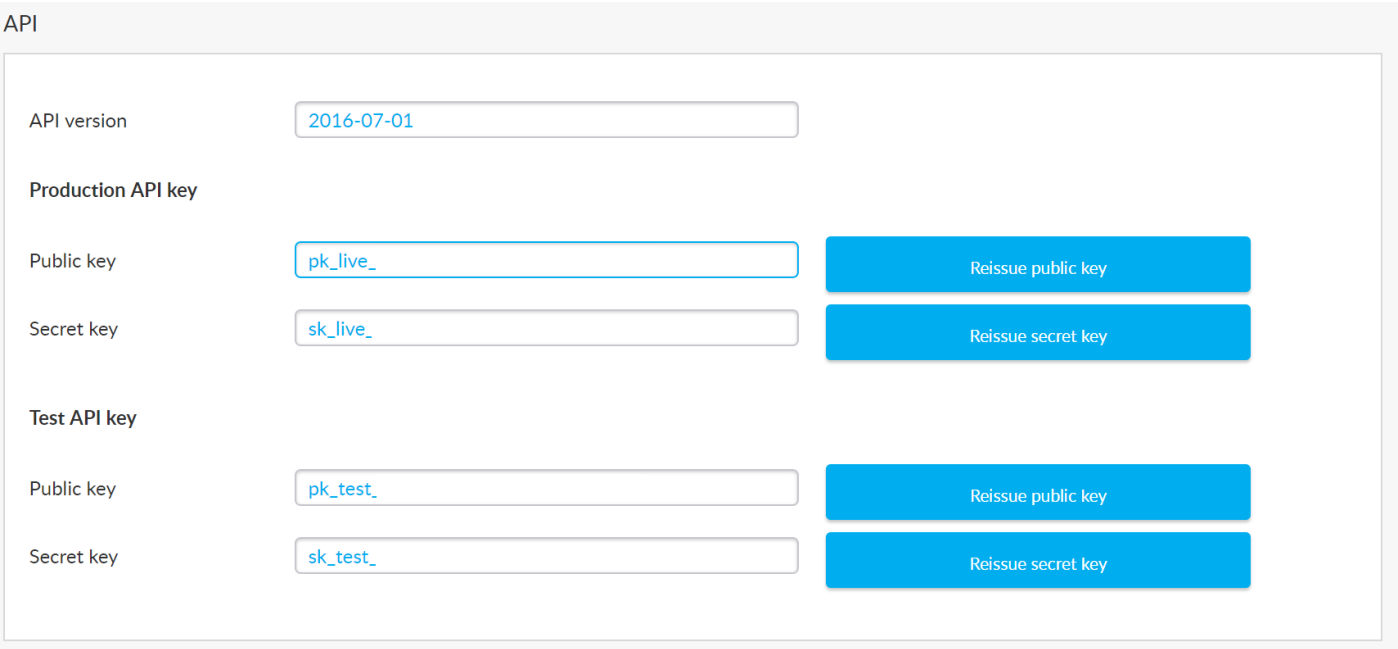
Order is given the below definitions as custom attributes.
paidyPaymentId: saves the ID issued when making a Paidy payment.
paidyToken: saves the token value used when using subscription payments.

Select All		ID	Name
☐		paidyPaymentId	
☐		paidyToken	

<Site Preferences>
Management > Site Development > System Object Types > Site Preferences > Attribute Definitions
System object: Site Preferences is an object that corresponds to site settings.

The below is defined as a custom attribute in Site Preferences.
paidy_api_key: public key data used to connect to the Paidy API.
This can be confirmed on the Paidy merchant management screen.
For a test, save the public key used for testing purposes.
paidy_secret_key: secret key data used to connect to the Paidy API.
This can be confirmed on the Paidy merchant management screen.
When testing, save the secret key used for testing purposes.

- Paidy merchant management screen (details found in section 4-1-1)



paidy_logo_url: the URL of a logo image to be displayed on the Paidy Checkout application.
If nothing is set, the Paidy logo will be shown.
paidy_service_name: key data used to call the information for communicating with Paidy as registered in BM.
This is fixed to paidy.api.payment.
paidy_store_name: the shop name displayed on Paidy Checkout application headers, MyPaidy, and the merchant dashboard.

paidy_time_out_second: the number of seconds before timeout when communicating with the Paidy Checkout application.

paidy_token_description (optional): specifies descriptions of tokens defined by the merchant.

Select All		ID	Name
☐		paidy_api_key	
☐		paidy_logo_url	
☐		paidy_secret_key	
☐		paidy_service_name	
☐		paidy_store_name	
☐		paidy_time_out_second	
☐		paidy_token_description	

<Customer Profile>
Management > Site Development > System Object Types > Customer Profile > Attribute Definitions
System object: Customer Profile is an object relating to customer data.

The below is defined as a custom attribute on the Customer Profile.
paidyToken: token data used for subscription payments. Saved when issuing a token.

Select All	ID	Name
☐	paidyToken	

<Custom site environment settings>
Merchant Tools > Site Environment Settings > Custom Site Environment Settings Group
Once metadata is successfully imported, a group with the Paidy ID will be created in the custom environment settings group.

Merchant Tools / Site Preferences /

Custom Site Preference Groups

View Across Sites | New

Paidy

1-1 of 1

ID	Name	Description	Preferences	View Across Sites
Paidy	Paidy		6	View

Clicking Paidy displays configuration screens for each value. Enter the requisite information and click the Save button at the top right.

Merchant Tools / Site Preferences / Custom Site Preference Groups /

Paidy

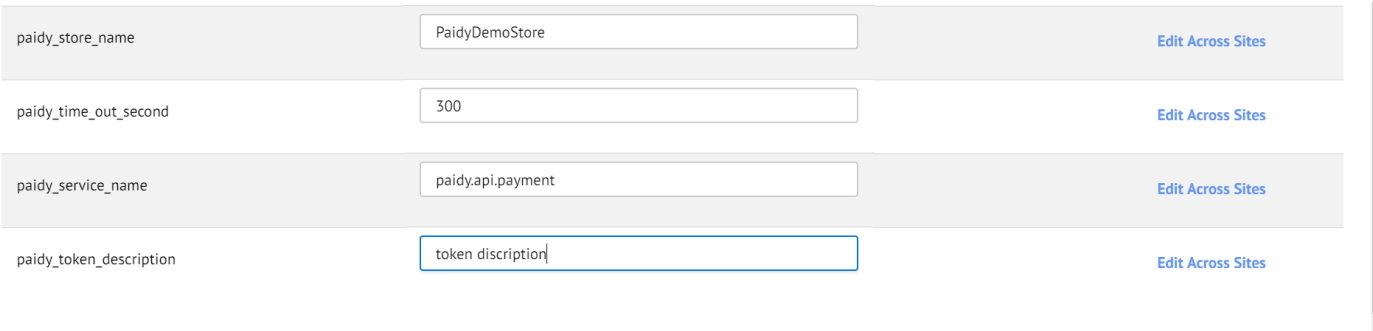
Cancel | Apply to Other Sites | Save

Instance Type
Sandbox

Search by IDs...

1-7 of 7

Name	Value	Default Value	
paidy_api_key	pk_test_h2c8sdht25l4n8ve6j2bs9gbtc		Edit Across Sites
paidy_logo_url	https://cdn.pixabay.com/photo/2017/08/30/17/12/waffle-hea		Edit Across Sites
paidy_secret_key	sk_test_oruc4lrrlvcce2bls930e9hk5l		Edit Across Sites



3-3. Modifying cartridges on the merchant

3-3-1. CheckoutJS installation

In order to call Paidy Checkout and process payment, the Checkout JS script is embedded in the payment page template.

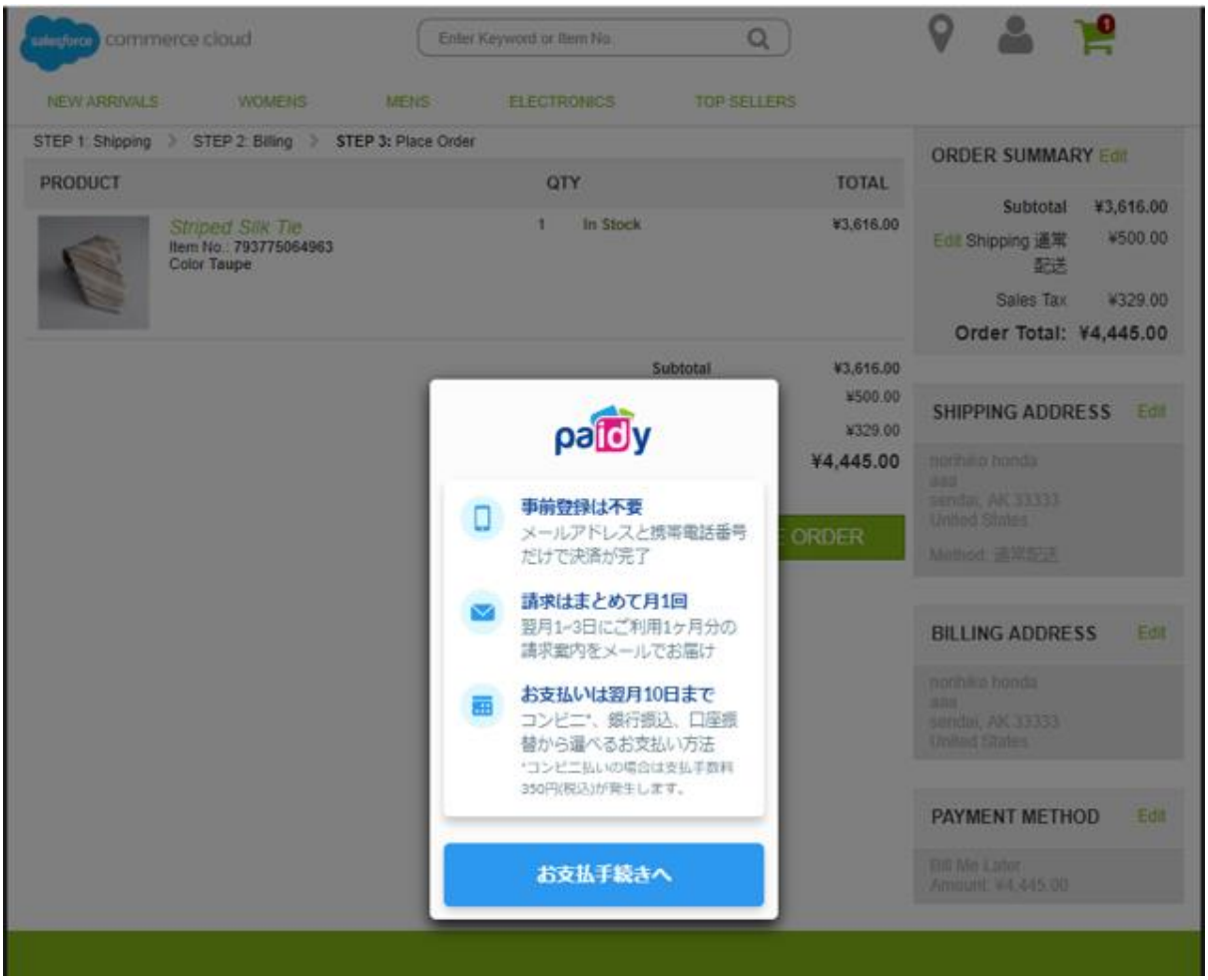
Including this script will call Paidy Checkout.
To embed the template, modify the following file.
- [storefront_core]\cartridge\templates\default\components\footer\footer_UI.isml
Add the below after <script src="${URLUtils.staticURL('/js/app.js')}"></script>

```
 ..........................
<isscript>
if (!empty(pdict.Basket)) {
    var paymentMethods  = pdict.Basket.getPaymentInstruments();
    var paymentMethod  = '';

   if (paymentMethods != null && !paymentMethods.isEmpty() ) {
    for( var i = 0; i < paymentMethods.length ; i++  ){
    if(  paymentMethods[i] != null && paymentMethods[i].paymentMethod != "GIFT_CERTIFICATE" )
      paymentMethod  = paymentMethods[i].paymentMethod;
    }
   }
}
</isscript>
<isif condition="${paymentMethod=='PAIDY_STANDARD'}">
<script type="text/javascript" src="https://apps.paidy.com/"></script>
<script src="${URLUtils.staticURL('js/paidyStandard.js')}"></script>
<iselseif condition="${paymentMethod=='PAIDY_SUBSCRIPTION'}">
<script type="text/javascript" src="https://apps.paidy.com/"></script>
<script src="${URLUtils.staticURL('js/paidySubscription.js')}"></script>
<iselse/>
<script>
(function(){
        var button = $('.submit-order button[type="submit"]');
        if(button.length) {
                button.prop('disabled', false);
```

```
                button.removeClass('disabled');
        }
}());
</script>
</isif>
```

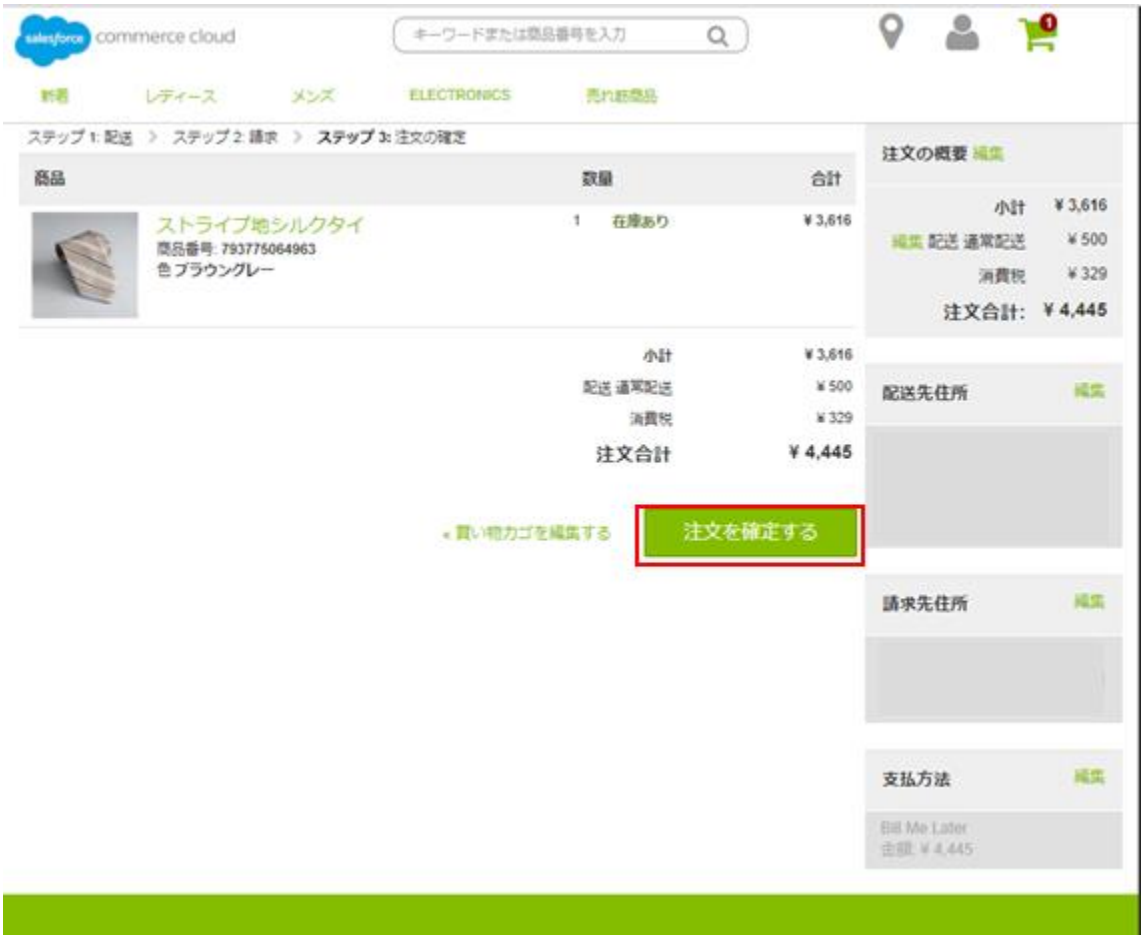
.....

```
<script src="${URLUtils.staticURL('/js/app.js')}"></script>

<isscript>
if (!empty(pdict.Basket)) {
    var paymentMethods  = pdict.Basket.getPaymentInstruments();
    var paymentMethod  = '';

   if (paymentMethods != null && !paymentMethods.isEmpty() ) {
    for( var i = 0; i < paymentMethods.length ; i++  ){
    if(  paymentMethods[i] != null && paymentMethods[i].paymentMethod != "GIFT_CERTIFICATE" )
      paymentMethod  = paymentMethods[i].paymentMethod;
    }
   }
}
</isscript>
<isif condition="${paymentMethod=='PAIDY_STANDARD'}">
<script type="text/javascript" src="https://apps.paidy.com/"></script>
<script src="${URLUtils.staticURL('js/paidyStandard.js')}"></script>
<iselseif condition="${paymentMethod=='PAIDY_SUBSCRIPTION'}">
<script type="text/javascript" src="https://apps.paidy.com/"></script>
<script src="${URLUtils.staticURL('js/paidySubscription.js')}"></script>
<iselse/>
<script>
(function(){
    var button = $('.submit-order button[type="submit"]');
    if(button.length) {
        button.prop('disabled', false);
        button.removeClass('disabled');
    }
}());
</script>
</isif>

<isif condition="${!('pageContext' in this) || empty(pageContext)}">
```


commerce cloud
Enter Keyword or Item No.
NEW ARRIVALS
WOMENS
MENS
ELECTRONICS
TOP SELLERS
STEP 1: Shipping
STEP 2: Billing
STEP 3: Place Order
PRODUCT
QTY
TOTAL
Striped Silk Tie
Item No.: 793775064963
Color Taupe
1
In Stock
¥3,616.00
Subtotal
¥3,616.00
¥500.00
¥329.00
¥4,445.00
ORDER SUMMARY Edit
Subtotal ¥3,616.00
Edit Shipping 通常配送 ¥500.00
Sales Tax ¥329.00
Order Total: ¥4,445.00
SHIPPING ADDRESS Edit
BILLING ADDRESS Edit
PAYMENT METHOD Edit
paidy
事前登録は不要
メールアドレスと携帯電話番号だけで決済が完了
請求はまとめて月1回
翌月1~3日にご利用1ヶ月分の請求案内をメールでお届け
お支払いは翌月10日まで
コンビニ*、銀行振込、口座振替から選べるお支払い方法
*コンビニ払いの場合は支払手数料350円(税込)が発生します。
お支払手続きへ

3-3-2. Controlling purchase confirmation buttons

When displaying the purchase confirmation screen and before the HTML source code is fully imported, if the confirm purchase button is clicked, the Paidy Checkout popup may fail to function properly.



Therefore, the purchase confirmation button must be styled such that it cannot be clicked when first displayed.
The template implements this as follows.
- [storefront_core]\cartridge\templates\default\checkout\summary\summary.isml
A button with the button-fancy-large class is also given the disabled class and disabled attribute.

```
<button class="button-fancy-large disabled" type="submit" name="submit"
value="${Resource.msg('global.submitorder','locale',null)}" disabled>
```

The tag is added immediately after the close tag on the <form action="${URLUtils.https('COSummary-Submit')}" method="post" class="submit-order"> add tags immediately after the form tag.

```
<input type="hidden" name="selected_payment_method"
value="${pdict.CurrentForms.billing.paymentMethods.selectedPaymentMethodID.value}"/>
```

The above form tag also includes the submit-order class; this class should be left in and not deleted.
The purchase confirmation button will now only be clickable when Paidy Checkout is ready.

```
215  <form action="${URLUtils.https('COSummary-Submit')}" method="post" class="submit-order">
216
217      <input type="hidden" name="selected_payment_method" value="${pdict.CurrentForms.billing.paymentMethods.selectedPaymentMethodID.value}"/>
218
219      <fieldset>
220          <div class="form-row">
221              <a class="back-to-cart" href="${URLUtils.url('Cart-Show')}">
222                  <isprint value="${Resource.msg('summary.editcart','checkout',null)}" encoding="off" />
223              </a>
224
225              <button class="button-fancy-large disabled" type="submit" name="submit" value="${Resource.msg('global.submitorder','locale',null)}" disabled>
226
227                  ${Resource.msg('global.submitorder','locale',null)}
228              </button>
229          </div>
230          <input type="hidden" name="${dw.web.CSRFProtection.getTokenName()}" value="${dw.web.CSRFProtection.generateToken()}"/>
231
232      </fieldset>
233  </form>
```

3-3-3. Modifying resource files

Using Paidy payments requires including URL and timeout information in a configuration file called a resource file.

- [storefront_core]\cartridge\scripts\util\Resource.ds

Add the below at the end of the urls section of ResourceHelper.getUrls

.....

```
getPaidyStandardPaidyConfig   : URLUtils.https("PAIDY_STANDARD-GetPaidyConfig").toString(),
doPaidyStandardCOSummarySubmit: URLUtils.https("COSummary-Submit").toString(),
doPaidyStandardFailOrder      : URLUtils.https("PAIDY_STANDARD-FailOrder").toString(),
doPaidyStandardPlaceOrder     : URLUtils.https("PAIDY_STANDARD-PlaceOrder").toString(),
getPaidyStandardOrderToken    : URLUtils.https("PAIDY_STANDARD-GetOrderToken").toString(),
getPaidySubscriptionConfig    : URLUtils.url('PAIDY_SUBSCRIPTION-GetPaidyConfig').toString(),
setPaidySubscriptionToken     : URLUtils.url('PAIDY_SUBSCRIPTION-SetPaidyToken').toString()
```

.....

Add the below at the end of the resources section of ResourceHelper.getResources の resources

.....

```
PAIDY_ERROR_TIMEOUT                : Resource.msg('paidy.payment.error.timeout', 'checkout', null)
```

.....

3-4. Modifying cartridges on the merchant

3-4-1. Modifying COBillingjs

A routine is added to clear the fields if the payment method is changed from credit card or BML to Paidy.

.....

```
function resetPaymentForms() {

    var cart = app.getModel('Cart').get();

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

```
cart.removePaymentInstruments(cart.getPaymentInstruments(PaymentInstrument.METHOD_CREDIT_CARD));
        cart.removePaymentInstruments(cart.getPaymentInstruments(PaymentInstrument.METHOD_BML));
        cart.removePaymentInstruments(cart.getPaymentInstruments('PAIDY_STANDARD'));
        cart.removePaymentInstruments(cart.getPaymentInstruments('PAIDY_SUBSCRIPTION'));
    } else if
(app.getForm('billing').object.paymentMethods.selectedPaymentMethodID.value.equals(PaymentInstrument.METHO
D_CREDIT_CARD)) {
        app.getForm('billing').object.paymentMethods.bml.clearFormElement();

        cart.removePaymentInstruments(cart.getPaymentInstruments(PaymentInstrument.METHOD_BML));
        cart.removePaymentInstruments(cart.getPaymentInstruments('PayPal'));
        cart.removePaymentInstruments(cart.getPaymentInstruments('PAIDY_STANDARD'));
        cart.removePaymentInstruments(cart.getPaymentInstruments('PAIDY_SUBSCRIPTION'));
    } else if
(app.getForm('billing').object.paymentMethods.selectedPaymentMethodID.value.equals(PaymentInstrument.METHO
D_BML)) {
        app.getForm('billing').object.paymentMethods.creditCard.clearFormElement();

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


cart.removePaymentInstruments(cart.getPaymentInstruments(PaymentInstrument.METHOD_CREDIT_CARD));
        cart.removePaymentInstruments(cart.getPaymentInstruments('PayPal'));
        cart.removePaymentInstruments(cart.getPaymentInstruments('PAIDY_STANDARD'));
        cart.removePaymentInstruments(cart.getPaymentInstruments('PAIDY_SUBSCRIPTION'));
    } else if
(app.getForm('billing').object.paymentMethods.selectedPaymentMethodID.value.equals('PAIDY_STANDARD')) {
        app.getForm('billing').object.paymentMethods.creditCard.clearFormElement();
        app.getForm('billing').object.paymentMethods.bml.clearFormElement();


cart.removePaymentInstruments(cart.getPaymentInstruments(PaymentInstrument.METHOD_CREDIT_CARD));
        cart.removePaymentInstruments(cart.getPaymentInstruments(PaymentInstrument.METHOD_BML));
        cart.removePaymentInstruments(cart.getPaymentInstruments('PayPal'));
        cart.removePaymentInstruments(cart.getPaymentInstruments('PAIDY_SUBSCRIPTION'));
    } else if
(app.getForm('billing').object.paymentMethods.selectedPaymentMethodID.value.equals('PAIDY_SUBSCRIPTION'))
{
            app.getForm('billing').object.paymentMethods.creditCard.clearFormElement();
            app.getForm('billing').object.paymentMethods.bml.clearFormElement();


cart.removePaymentInstruments(cart.getPaymentInstruments(PaymentInstrument.METHOD_CREDIT_CARD));
            cart.removePaymentInstruments(cart.getPaymentInstruments(PaymentInstrument.METHOD_BML));
            cart.removePaymentInstruments(cart.getPaymentInstruments('PayPal'));
```

```
                cart.removePaymentInstruments(cart.getPaymentInstruments('PAIDY_STANDARD'));
            }
        return true;

    });

    return status;
}.........................
```

```
function resetPaymentForms() {
    var cart = app.getModel('Cart').get();

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

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

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

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

            cart.removePaymentInstruments(cart.getPaymentInstruments('PAIDY_STANDARD'));
            cart.removePaymentInstruments(cart.getPaymentInstruments('PAIDY_SUBSCRIPTION'));
        } else if (app.getForm('billing').object.paymentMethods.selectedPaymentMethodID.value.equals(PaymentInstrument.METHOD_BML)) {
            app.getForm('billing').object.paymentMethods.creditCard.clearFormElement();

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

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

            cart.removePaymentInstruments(cart.getPaymentInstruments('PAIDY_STANDARD'));
            cart.removePaymentInstruments(cart.getPaymentInstruments('PAIDY_SUBSCRIPTION'));
        } else if (app.getForm('billing').object.paymentMethods.selectedPaymentMethodID.value.equals('PAIDY_STANDARD')) {
            app.getForm('billing').object.paymentMethods.creditCard.clearFormElement();
            app.getForm('billing').object.paymentMethods.bml.clearFormElement();

            cart.removePaymentInstruments(cart.getPaymentInstruments(PaymentInstrument.METHOD_CREDIT_CARD));
            cart.removePaymentInstruments(cart.getPaymentInstruments(PaymentInstrument.METHOD_BML));
            cart.removePaymentInstruments(cart.getPaymentInstruments('PayPal'));
            cart.removePaymentInstruments(cart.getPaymentInstruments('PAIDY_SUBSCRIPTION'));
        } else if (app.getForm('billing').object.paymentMethods.selectedPaymentMethodID.value.equals('PAIDY_SUBSCRIPTION')) {
            app.getForm('billing').object.paymentMethods.creditCard.clearFormElement();
            app.getForm('billing').object.paymentMethods.bml.clearFormElement();

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

    return status;
}
```

3-4-2. COPlaceOrder.js modifications

When creating an order in Ajax on Paidy, the below is added to suspend the process before executing payment.

```
.........................
function handlePayments(order) {
(Abbreviated)
} else {

var authorizationResult = PaymentProcessor.authorize(order, paymentInstrument);

if (authorizationResult.not_supported || authorizationResult.error) {
return {
error: true,
PlaceOrderError: authorizationResult.PlaceOrderError
};
}
}
        }
    }

    return {};
}

function start() {
(Abbreviated)
    var handlePaymentsResult = handlePayments(order);

    if (handlePaymentsResult.error) {
        return Transaction.wrap(function () {
OrderMgr.failOrder(order);

if (handlePaymentsResult.PlaceOrderError != null) {
                                return {
                                        error : true,
                                        PlaceOrderError : handlePaymentsResult.PlaceOrderError
                                };
                        } else {
                                return {
                                        error : true,
                                        PlaceOrderError : new Status(Status.ERROR,'confirm.error.technical')
                                };

                        }
        });

    } else if (handlePaymentsResult.missingPaymentInfo) {
(Abbreviated)
    }

    if (request.httpParameterMap.format.stringValue == 'ajax') {
        return {
```

```
Order: order,
paidy_order_created: true
        };
    }

    var orderPlacementStatus = Order.submit(order);
    if (!orderPlacementStatus.error) {
        clearForms();
    }
    return orderPlacementStatus;
}
```
.....

3-4-3. COSummary.js modifications

When creating an order in Ajax on Paidy, the below is added to create a JSON response if the process is suspended before payment is processed.

.....

```
function submit() {
    var placeOrderResult = app.getController('COPlaceOrder').Start();
    if (placeOrderResult.error) {
        start({
PlaceOrderError: placeOrderResult.PlaceOrderError
        });
    } else if (placeOrderResult.order_created) {
        showConfirmation(placeOrderResult.Order);
    } else if (placeOrderResult.paidy_order_created) {
        var paidyCartridge = require('~/package.json').paidyCartridge;
        var currentForms = session.forms;

        responseJson = require(paidyCartridge +
'/cartridge/scripts/paidy/standard/Authorize').GetConfirmationPaidyJSON(currentForms.billing.paymentMet
hods.selectedPaymentMethodID.value, customer, placeOrderResult.Order);
        app.getView({
JSONResponse: responseJson
        }).render('util/responsejson');
    }
}
```
.....

```
function submit() {
    // Calls the COPlaceOrder controller that does the place order action and any payment authorization.
    // COPlaceOrder returns a JSON object with an order_created key and a boolean value if the order was created successfully.
    // If the order creation failed, it returns a JSON object with an error key and a boolean value.
    var placeOrderResult = app.getController('COPlaceOrder').Start();
    if (placeOrderResult.error) {
        start({
            PlaceOrderError: placeOrderResult.PlaceOrderError
        });
    } else if (placeOrderResult.order_created) {
        showConfirmation(placeOrderResult.Order);
    } else if (placeOrderResult.paidy_order_created) {
        var paidyCartridge = require('~/package.json').paidyCartridge;
        var currentForms = session.forms;

        app
          .getView({
              JSONResponse: require(paidyCartridge +
              '/cartridge/scripts/paidy/standard/authorize').getConfirmationPaidyJSON(
              currentForms.billing.paymentMethods.selectedPaymentMethodID.value,
              customer,
              placeOrderResult.Order
            )
          })
          .render('util/responsejson');
    }
}
```

3-4-4. Modifying to package.json on the JS controller cartridge at the merchant

Add a reference to the Paidy cartridge to the package.json used on the merchant's JSController cartridge.

.....

```
{
  "hooks": "./cartridge/scripts/hooks.json",
  "paidyCartridge": "int_paidy"
}
```

.....

3-4-5. Modifying package.json on the int_paidy_controllers cartridge

Change the name of the cartridge called by "controllers" to the name of the JSController cartridge used by the merchant.

.....

```
{
   "paidyCartridge": "int_paidy",
   "hooks": "./cartridge/scripts/hooks.json",
   "controllers": "paidy_storefront_controllers"
}
```

.....

3-5. Advice

- Notes on use of subscription purchases

If you followed the steps to add a payment method as indicated in section 3-2-3, by default, you will have two separate entries -- standard payments and subscription payments.

If there are means to detect whether subscription payments have been selected when transitioning to the payment selection screen, CSS or other means can be used to restrict the display to only show Paidy next-month payment (convenience store/bank).

- Notes on tokens
Merchant Tools > Clients > Client Management > Client Numbers: All
Tokens can be checked via client data.

If, for some reason, you want to reissue a token, enter a null value in paidyToken and have the consumer purchase another subscription payment.

- Order confirmation e-mails for subscription purchases
The int_paidy cartridge is designed such that order confirmation e-mails are sent in the same manner as standard e-mails even for subscription purchases. To customize the e-mail sent, you can modify the int_paidy cartridge.

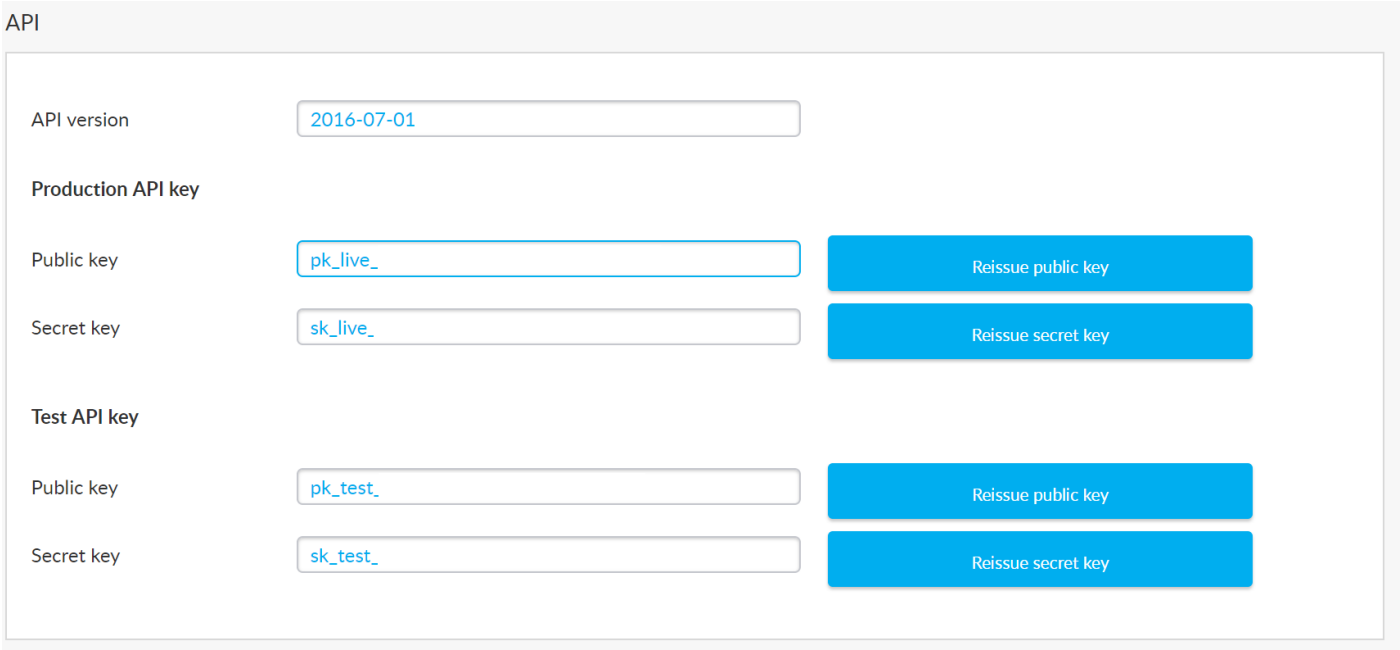
4. Cartridge deployment testing at merchant

4-1. Before beginning a test

4-1-1. Paidy merchant management screen

- Confirming the test API key

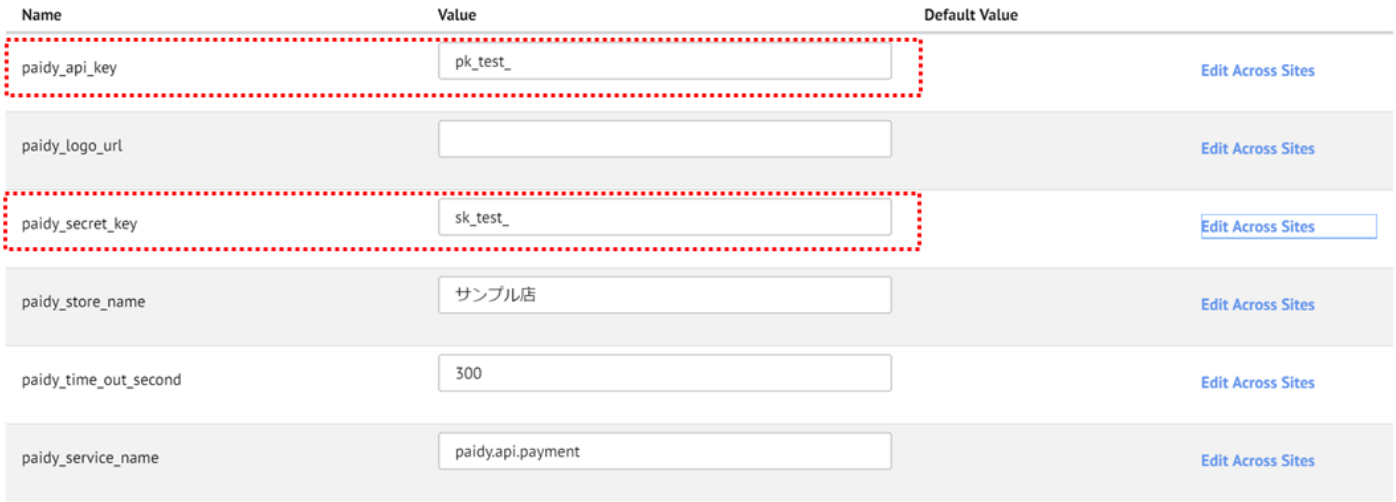
Check the public and secret keys used for the test API; these will be entered into the custom site environment settings screen.



4-1-2.Commerce Cloud Business Manager

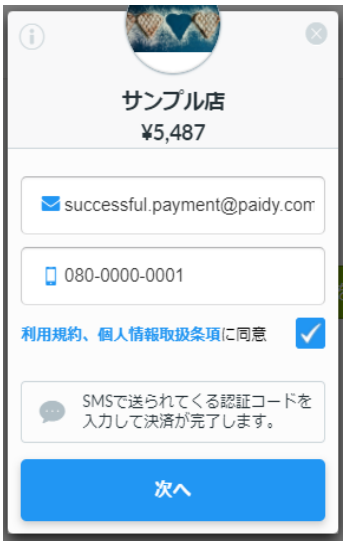
- API key settings

From the above settings screen, enter the test API keys you found for paidy_api_key and paidy_secret_key in section 4-1-1. Paidy merchant management screen.



4-2.PaidyCheckout test data

- E-mail address and phone number



E-mail address and telephone usable for test cases

Test cases	E-mail address	phone number
Success	successful.payment@paidy.com	080-0000-0001
Failure	rejected.payment@paidy.com	

*Note that the e-mail address and phone number for the above member data and shipping and billing address can differ.

- Authentication number

Enter 8888.

4-3. Browser compatibility

This cartridge supports the following OSes and browsers.

Device - OS	Browser
PC: Windows	Edge: latest version
	Chrome: latest version
	Firefox: latest version
PC: MacOS	Chrome: latest version
	Safari: latest version
Smartphone: iOS	iPhone (latest version of iOS), Safari iPhone (latest version of iOS), Chrome
Smartphone: Android	Android 4.4 and later, Chrome

4-4. Test cases

Test cases to run after configuring the cartridge are listed below.

4-4-1. Paidy standard payments

Standard payment is tested using a guest and a member.

1. Successful order as guest
- If not logging in and selecting a payment method for Paidy standard payments, the PaidyCheckout popup appears. Entering a successful e-mail address and phone number will display the product purchase confirmation screen.
- The Commerce Cloud order number created on the Paidy merchant site should be displayed as a transaction ID
- The order status for the corresponding order number on Business Manager on Commerce Cloud will be new

2. Order success as member
- After logging in, when selecting a payment method for Paidy standard payments, the PaidyCheckout popup appears. Entering a successful e-mail address and phone number will display the product purchase confirmation screen.
- The Commerce Cloud order number created on the Paidy merchant site should be displayed as a transaction ID
- The order status for the corresponding order number on Business Manager on Commerce Cloud will be new

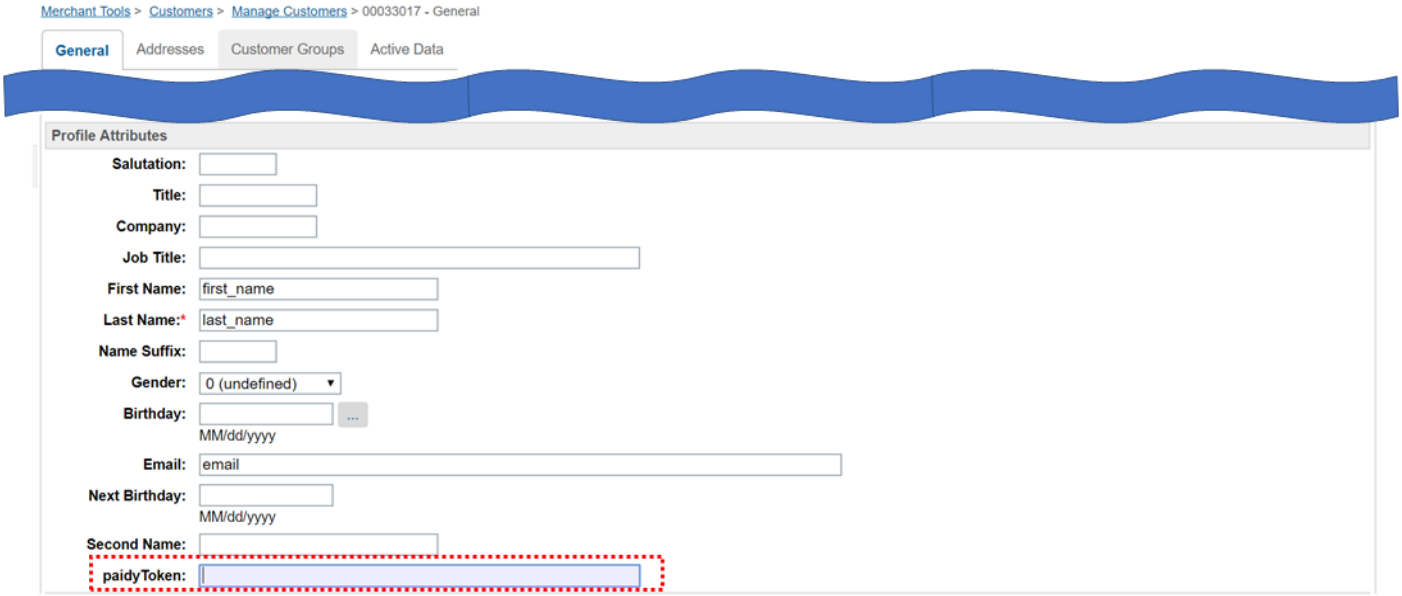
3. Failed order as guest
- If not logging in and selecting a payment method for Paidy standard payments, the PaidyCheckout popup appears. Entering a failing e-mail address and phone number will display an error message.

- The order status for the corresponding order number on Business Manager on Commerce Cloud will have failed

4. Order failure as member
- After logging in, when selecting a payment method for Paidy standard payments, the PaidyCheckout popup appears. Entering a failing e-mail address and phone number will display an error message.
- The order status for the corresponding order number on Business Manager on Commerce Cloud will have failed

4-4-2. Paidy subscription payments

Use of subscription payments always requires logging in. Once the user succeeds at authentication for a subscription payment, the paidyToken containing that member's data is saved. Subsequent purchases do not display the PaidyCheckout popup. For testing purposes, you may elect to delete the paidyToken from Business Manager on Commerce Cloud in order to display the popup again.



1. Success with no token
- When logging in as a member with no token yet issued and selecting a payment method for Paidy subscription payments, the Confirm Order button is clicked and the PaidyCheckout popup appears. Entering a successful e-mail address and phone number and proceeding to processing the payment will display a product purchase confirmation screen
- The Commerce Cloud order number created on the Paidy merchant site should be displayed as a transaction ID
- The order status for the corresponding order number on Business Manager on Commerce Cloud will be new

2. Failure with no token
- When selecting a payment method for Paidy subscription payments and clicking the Confirm Order button, the PaidyCheckout popup appears. Entering a failing e-mail address and phone number and proceeding to processing the payment will display an error message
- The order status for the corresponding order number on Business Manager on Commerce Cloud will have failed

3. Order success with token

- When logging in as an account that has previously succeeded with a Paidy subscription payment and selecting a payment method for a Paidy subscription payment and clicking the Confirm Order button, a product purchase confirmation screen is displayed
- Because a token exists, the PaidyCheckout popup does not appear
- The Commerce Cloud order number created on the Paidy merchant site should be displayed as a transaction ID
- The order status for the corresponding order number on Business Manager on Commerce Cloud will be new

5. Webhook usage examples

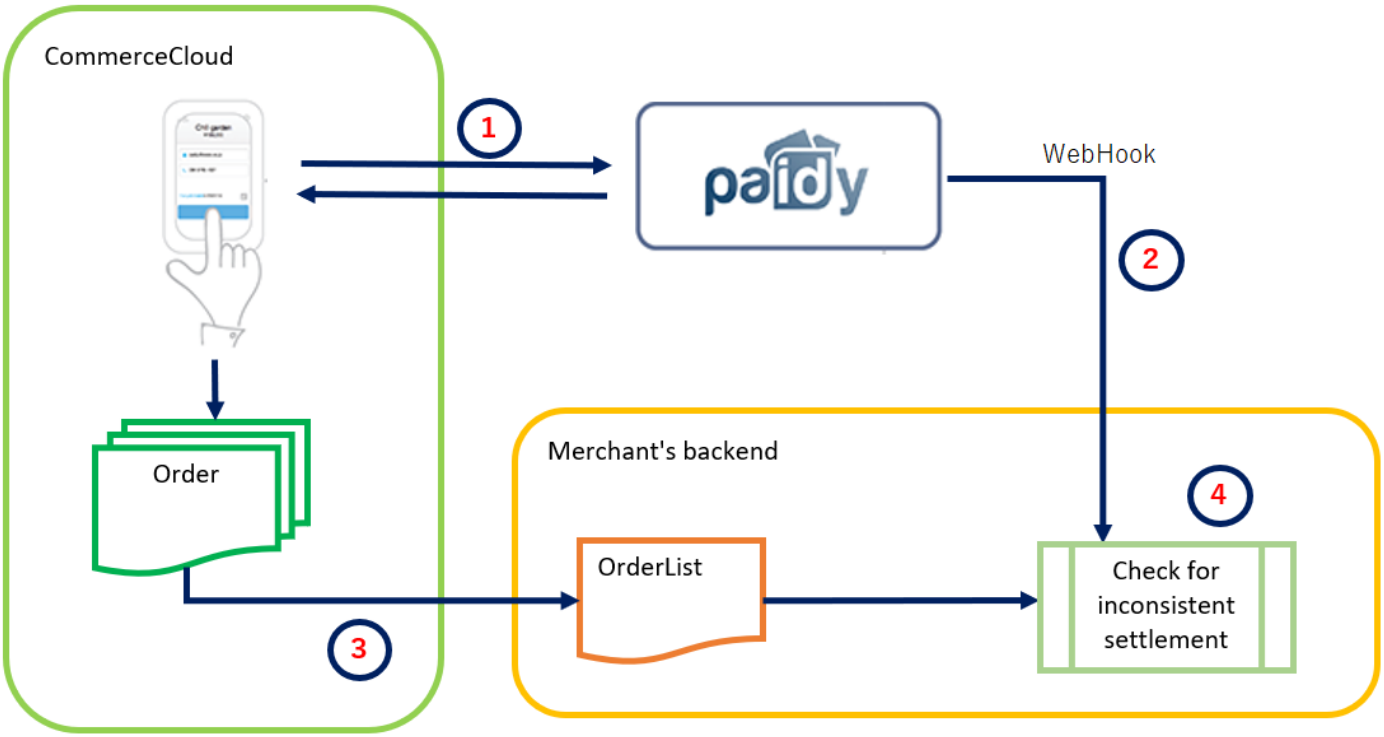
5-1. Webhook usage

Paidy payments run credit holds for a purchase by means of communication directly between a consumer and Paidy over JavaScript. Therefore, one limitation of the system is that if the consumer bounces or the connection is cut, the e-commerce site may fail to obtain the results of payment confirmation and the stale payment data will remain on Paidy.

In the event of an inconsistency like the above, there are two problems.
1. The shop loses the opportunity to complete the order.
2. If the order does not have to be completed, the consumer has a credit hold placed against their Paidy account.

To accommodate these cases, a Webhook is provided.
The merchant can use the Webhook to detect inconsistent payments.

5-2. Detecting inconsistent payments



1. Payment

Payment is made via Commerce Cloud using Paidy. A callback from Paidy's payment results returns one of the three order statuses below.
- Payment success: New Outputted to the order list. Payment possible status
- Payment failure: Failed Not outputted to the order list. Payment not possible status
- Processing: Created Not outputted to the order list. When in a payment not possible status, updating the payment transitions to a payable status.

2. Webhook
   Setting Webhook endpoints on the merchant backend allows for receiving notifications from Paidy of payment completion.
3. Order list
   A list of successful orders processed through batch processing on Commerce Cloud is sent as an xml file to the merchant's backend.
4. Inconsistent payment check processing
   Checks for payments where Webhook data (2) is found, but no payment data (3) exists on Commerce Cloud.

5-3. Handling inconsistencies

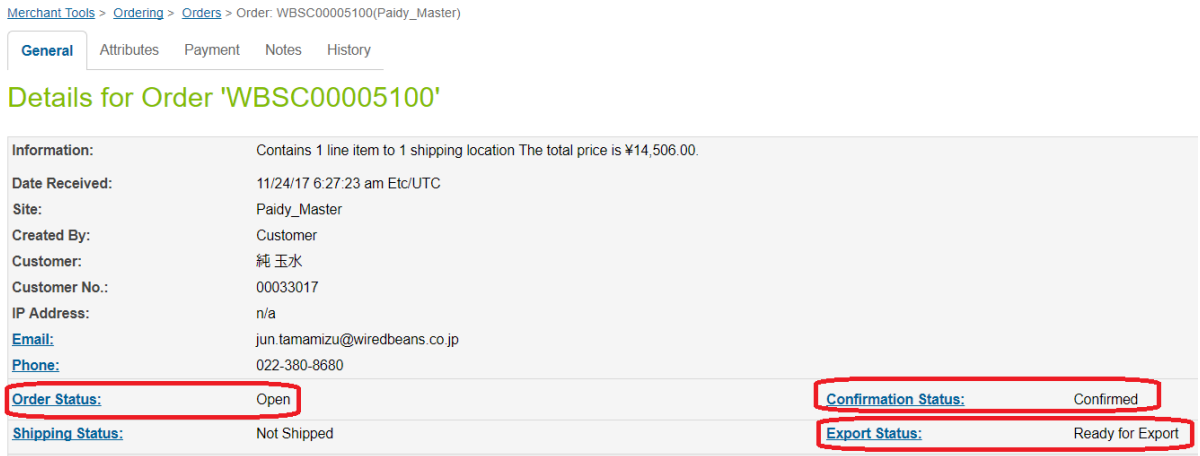
Inconsistent payments that have been detected can be handled as follows, allowing for resolving the above issues.

5-3-1. Processing example 1

In this solution, a list of inconsistent/invalid payments is outputted and can be checked by the merchant.
After a check is performed, the necessary steps are taken (approving or cancelling payment).

1. Establishing payment
   Use the Order function on Business Manager on Commerce Cloud to update the status.

   - General tab



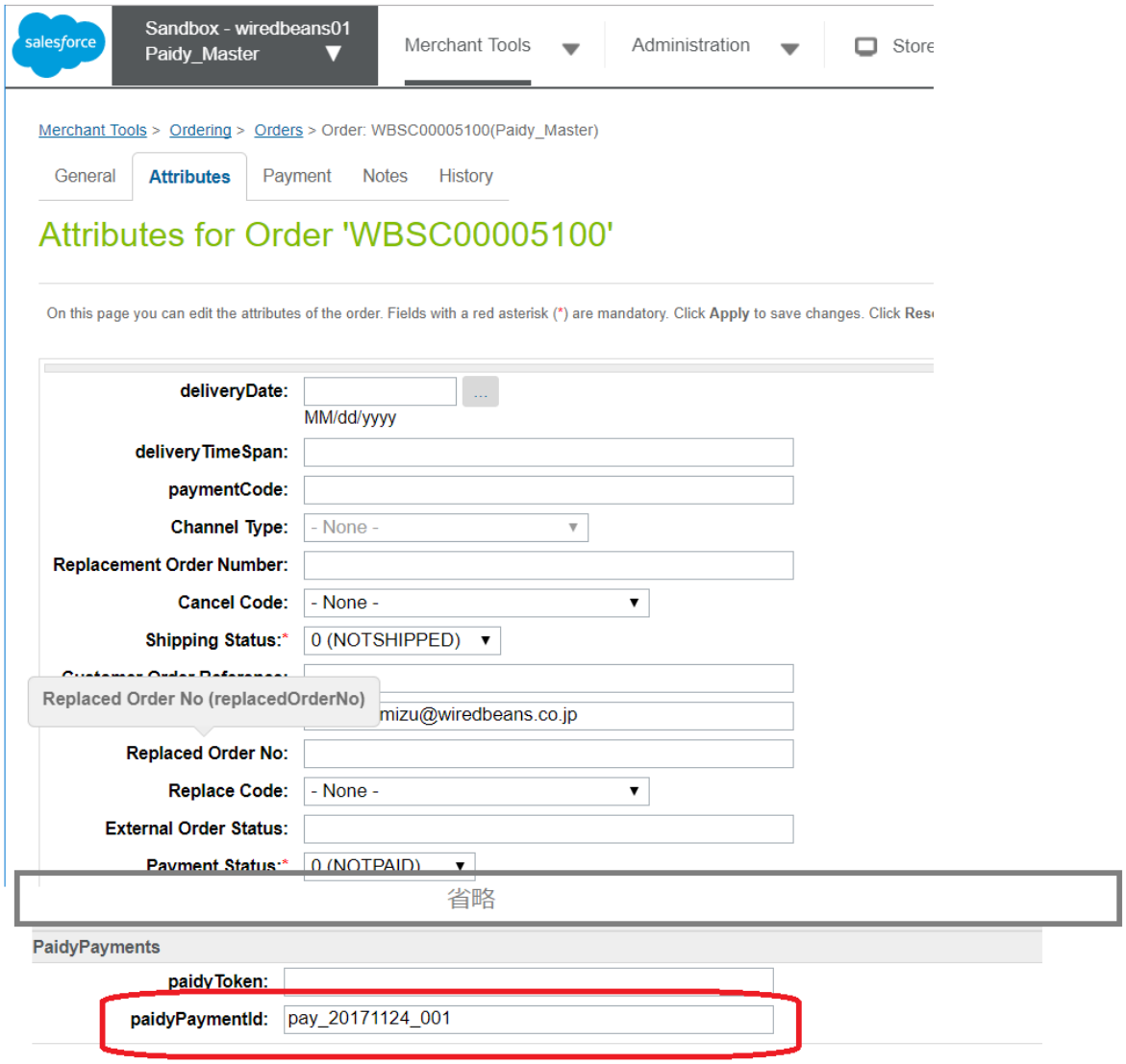
The status of the above three points is changed per below

Order Status: Created→Open

Confirmation Status: Not Confirmed→Confirmed

Export Status:Not Exported→Ready for Export

- Attributes tab



Set the paidyPaymentID that arrived via the Webhook for paidyPaymentId

*In order to e-mail an order confirmation e-mail as you would with StoreFront, it must be customized, as the format differs (see below).

- Order confirmation e-mail (concept)

■SiteGenesis StoreFront e-mail (normally used e-mail)

Salesforce Commerce Cloud SiteGenesis

Salesforce Commerce Cloud SiteGenesis
5 Wall Street
Burlington, MA 01803
salesforce.com
電話:+1.800.555.0199

ご注文いただきありがとうございました

ご注文に関してご質問などがございましたら、お電話 (012-345-XXXX) にてご遠慮なくお問い合わせください

(月曜日～金曜日、午前 8 時～午後 8 時)。

注文日: 24-11-2017

注文番号: WBSC00005102

支払情報

請求先住所
日本 980-0014
宮城県 仙台市青葉区 本町
玉水 純

支払方法
Paidy翌月払い（コンビニ/銀行）
金額: ￥2,492

支払金額合計
小計 ￥1,808
配送 通常配送 ￥500

E-mails sent from SendMail button on bottom of General tab on BusinessManager > Orders

SiteGenesis

Demo SiteGenesis
5 Wall Street
Burlington, MA 01803
SiteGenesis
Phone: +1.888.553.9216

Order Confirmation: Order Number: WBSC00005101 - Thank you for your order!

This email confirms we received your order at SiteGenesis. We will email your shipping confirmation with your tracking number when your items have been shipped. Your order will generally be shipped within 24 hours after we've received it. You can check your order's status on your account pages at SiteGenesis

Thank you for shopping with us!

Please print and save a copy for reference.

Please review the order below and verify that everything is correct.

PLEASE NOTE: Since we begin processing your order shortly after you submit it on our web site, if any changes are necessary you MUST contact us via PHONE ONLY, at **+1.888.553.9216**

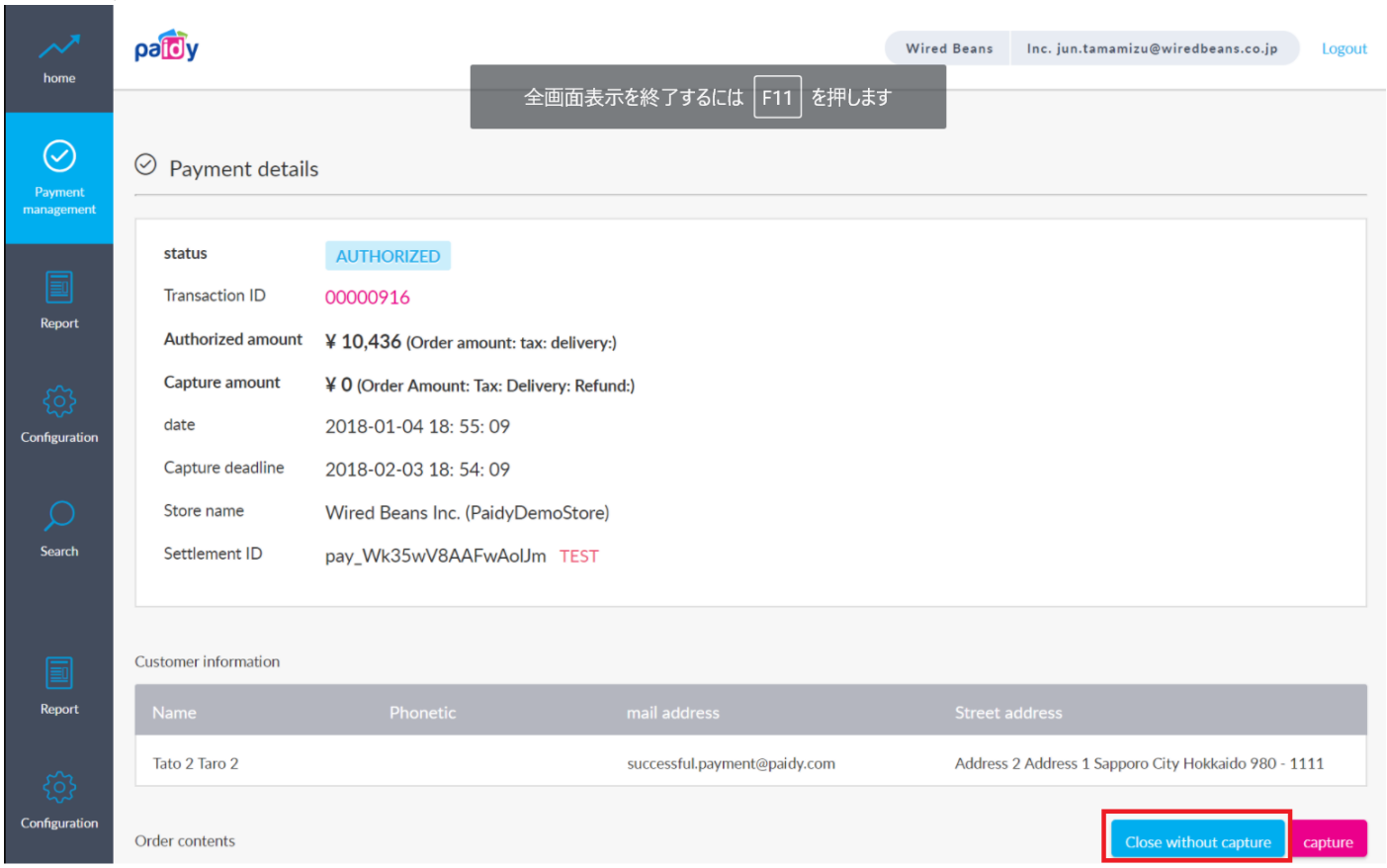
SHIPMENT 1

PRODUCT DETAILS	QTY.	TOTAL	SHIPPING DETAILS
Samsung Series 6 22" LCD High Definition Television Item Number:samsung-ln22a650 全品セール中	1	￥66,318 ￥- 33,159	純 玉水 本町 仙台市青葉区, 宮城県 980-0014 通常配送 - ￥500

PAYMENT METHOD	AMOUNT
Payment Total	**￥36,351**

2. Cancelling payments

Displaying the transaction in question on the Paidy merchant management screen and Canceling it (do not capture and close).



5-3-2. Processing example 2

In this example, the system makes a determination and automatically cancels inconsistent payments.
If, after referencing the order data and Webhook data on Commerce Cloud, the payment does not complete on Commerce Cloud, and authorization data is found only on Paidy, the close API is requested and the payment is closed.

https://paidy.com/docs/jp/payments.html#close

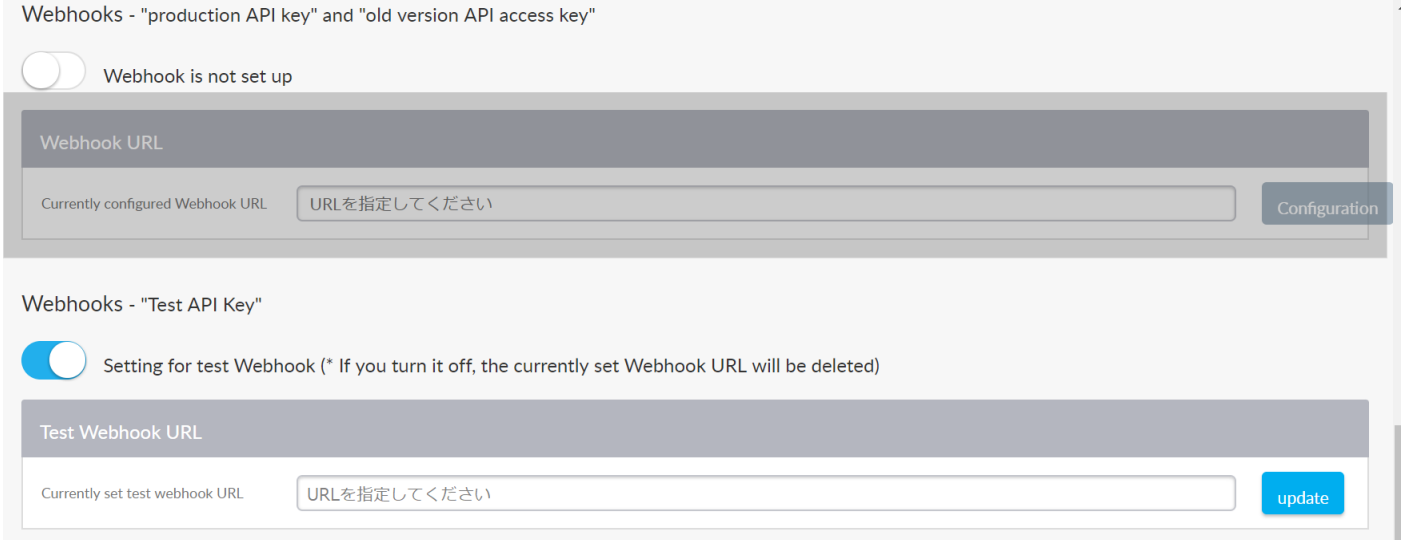
5-4. Webhook settings

In order to receive notifications from Paidy on the backend, the Webhook URL must be saved to the Paidy merchant management screen settings.
URL: https://merchant.paidy.com

You can enable these settings for a live environment by setting the merchant’s live backend URL in the Webhooks: Live API Key and Old Version API Access Key sections.

In order to check connectivity with test payments before configuring the live environment, set the merchant's backend test URL in Webhooks: Test API Key.

Paidy merchant management screen



6. What to do when the service is down

・what happens when the service is down

If you select Paidy payment, you will get an error screen or the pop-up will stop working.

・what the merchant should do to fix it

To get started, please contact Paidy Technical Support.

Paidy Technical Support
Mail: tech-support@paidy.com
TEL : 03-5545-5099 (weekdays 10:00-18:00)

7. Revision history

Version	Date	Revisions
17.10.0	2017/10/13	First edition

18.01.0	2018/01/09	- Revised steps to account for developing cartridges with JSController and use on both Pipeline sites and JSController sites - Revisions pursuant to deprecated Webhook on Commerce Cloud Added 5. Operational examples
19.01.0	2019/10/25	-Correct meta information file and update usage procedure. -The function “ServiceRegistry” used in JScontroller is abolished and “LocalServiceRegistry” is used. -Implement "getResponseLogMessage" in the service process called by JScontroller. -All HTTPClient calls now support service usage. -Add DocComment to each function.
20.1.0	2020/01/15	3-2-2. Importing metadata -Modify 「meta_data_19.1.0.zip」to 「meta_data_20.1.0.zip」
20.1.0	2020/4/5	- The following changes were made due to the file name change 3-3-1. CheckoutJS installation before： <script src="${URLUtils.staticURL('js/paidy_normal.js')}"></script> <script src="${URLUtils.staticURL('js/paidy_regular.js')}"></script> after： <script src="${URLUtils.staticURL('js/paidyNormal.js')}"></script> <script src="${URLUtils.staticURL('js/paidyRegular.js')}"></script> -Add 6. What to do when the service is down - Removed references to the Pipeline site.

21.1.0	2021/6/21	-Update 1. Cartridges version update -Update 15. Changes: Changed "meta_data_20.1.0.zip" to "meta_data_21.1.0_sg.zip"
22.1.0	2022/6/21	Update 1. Cartridges version update -Update document all page. Change Paidy payments name paidy normal->paidy standard paidy regular->paidy subscription -Update 15. Import meta file name update - Update 33. Delete IE11 from Browser compatibility