Paidy payment processing SFRA

Table of contents

Detailes

1. Summary

This implementation guide supports Store Front Reference Architecture (SFRA 6.1.0) .
Compatibility Mode up to 22.7 is supported.

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 <u>4-1. Before beginning a test</u>..

A cartridges must be implemented in order to begin using Paidy payments.
・”int_paidy_sfra” is the core cartridge that enables use of Paidy payments on the store.
＊Other cartridges are used on SiteGenesis based cartridges.

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.
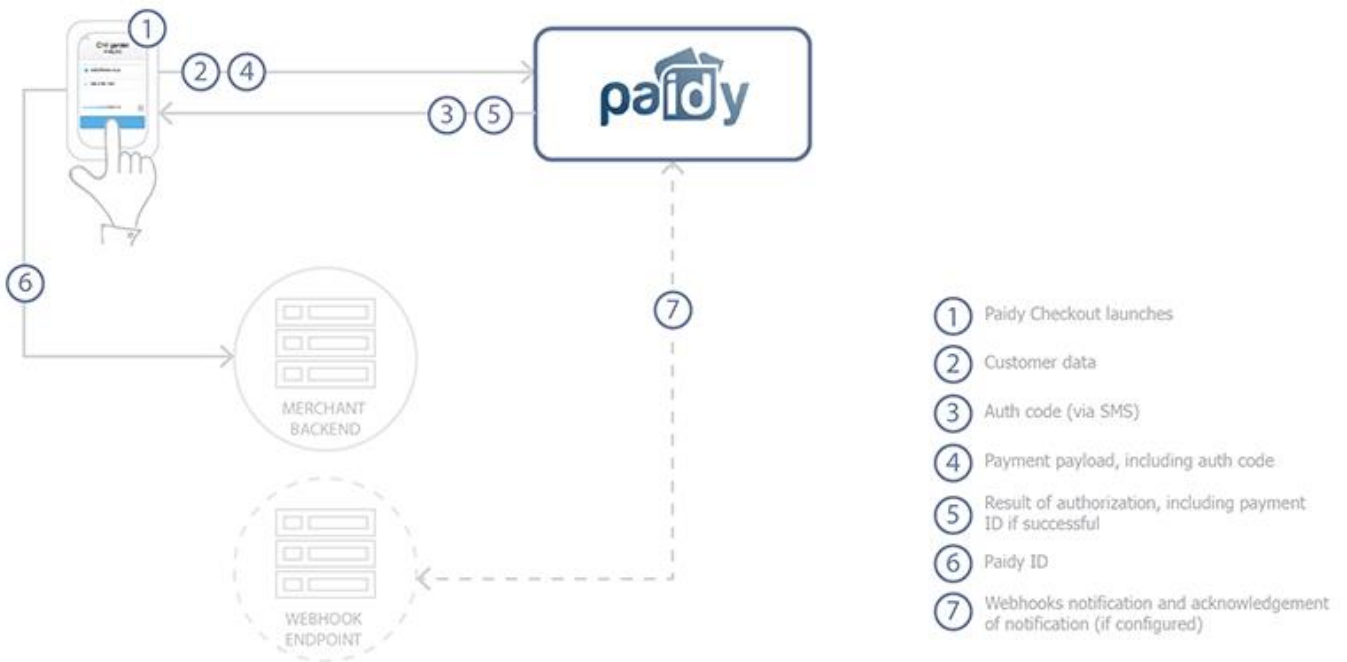
2-2. Restrictions and limitations

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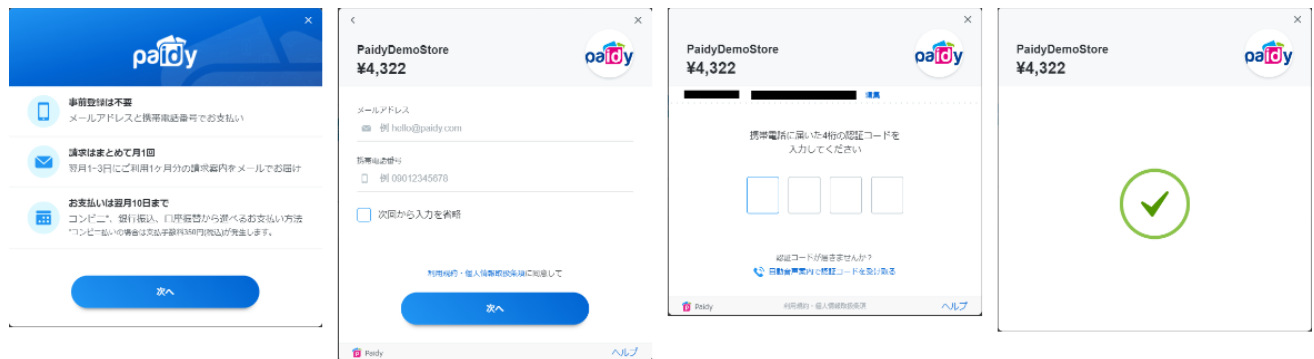
The requisite information is filled in and Paidy payment is processed.

If the payment succeeds, the order is complete.

ご注文いただきありがとうございました。

Eメールによる確認がまもなく ████████████ に送信されます。

領収書

注文番号: 00000505
注文日: 31/10/19

Order data created on Commerce Cloud

注文 '00000506' の詳細

情報:	1 件の配送先への 1 項目を含む。 合計金額は ¥ 4.322,00 です。
受取日:	10/31/19 8:50:19 AM Etc/UTC
サイト:	PaidyDEMO
作成者:	顧客
顧客:	******* *******
顧客ナンバー:	1001
IP アドレス:	該当なし
Eメール:	████████████
電話:	████████

| 注文ステータス: | 未処理 | | 確認ステータス: | 確認済み |
| 配送ステータス: | 未配送 | | エクスポートステータス: | エクスポート可能 |

配送 00002503

個数	商品 ID	名前	Manufacturer (製造元)	税率	小売単価	税基準	項目合計
1	793775362380M	ストライプ地シルクタイ		5,00 %	¥ 3.616,00	¥ 3.616,00	¥ 3.616,00
						荷物配送料: :	¥ 500,00
						合計配送料 (Delivery):	¥ 500,00
						配送合計:	¥ 500,00
						税金合計:	¥ 206,00
						合計:	¥ 4.322,00

Eメールの送信　注文の印刷

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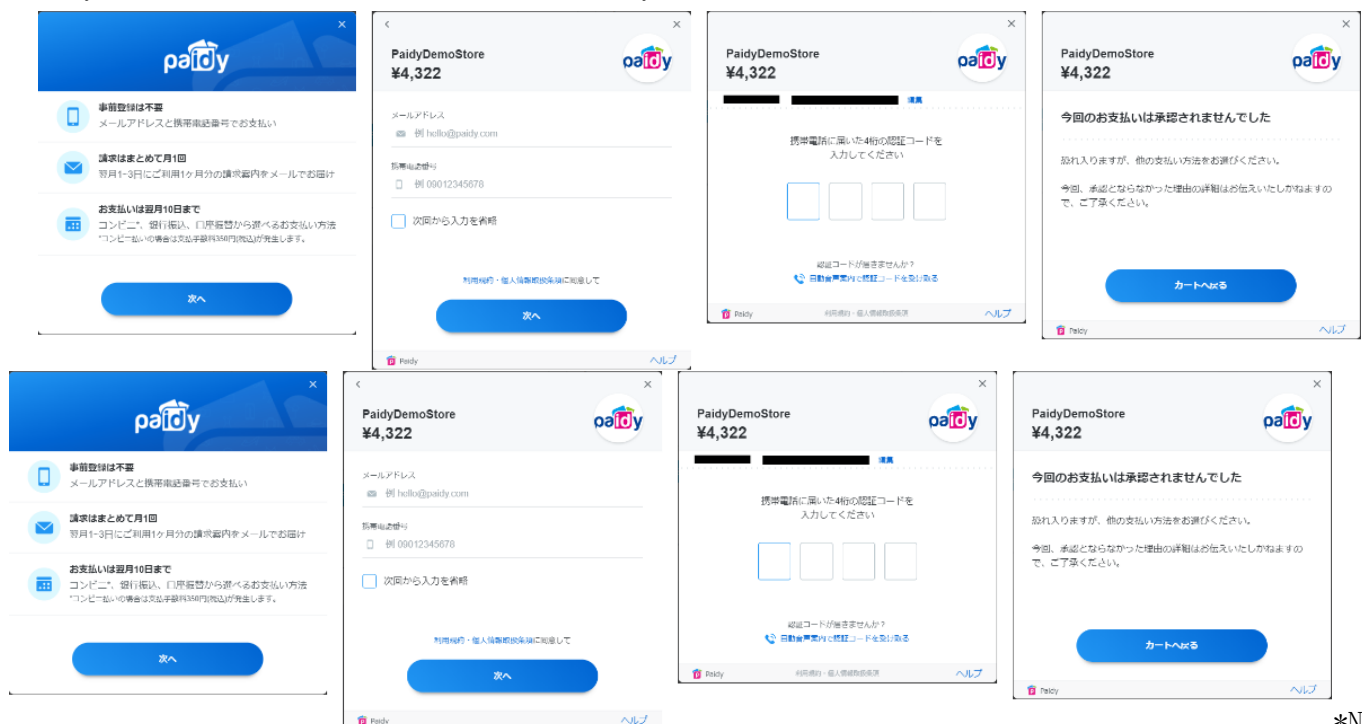
*Note that if the top-right X button is clicked on the popup, order failure data is created. Payment data will not be created on Paidy.

Order data created on Commerce Cloud (order failure)

注文 '00000601' の詳細

情報:	1 件の配送先への 1 項目を含む。合計金額は ¥ 4.322,00 です。
受取日:	11/1/19 1:15:11 AM Etc/UTC
サイト:	PaidyDEMO
作成者:	顧客
顧客:	******* *******
顧客ナンバー:	1001
IP アドレス:	該当なし
Eメール:	████████████████
電話:	████████

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	793775362380M	ストライプ地シルクタイ	CANCELLED		5,00 %	¥ 3.616,00	¥ 3.616,00	~~¥ 3.616,00~~
							荷物配送料: : CANCELLED	~~¥ 500,00~~
							合計配送料 (Delivery):	¥ 500,00
							配送合計:	¥ 500,00
							税金合計:	¥ 206,00
							合計:	¥ 4.322,00

Eメールの送信　注文の印刷

2-3-2. Subscription payments

Only usable by logged in users.

When clicking on the order confirmation button, 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 order.

- 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.

支払

編集

請求先住所:
100-0000
東京都 ********
****
************ *************

████████████████████
████████████

支払:
Paidy翌月払い（コンビニ/銀行）※定期購入

注文を確定する

色: レッド
在庫あり

単価	数量	合計
¥ 4,763 ¥ 3,616	1	¥ 3,616

The Paidy Checkout popup appears.

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)

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_sfra

The cartridge is for Store Front Reference Architecture (SFRA).
*If your website is based SiteGenesis, please refer to SiteGenesis implementation guide.

Use Commerce Cloud UX-studio to upload the int_paidy_sfra 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

RefArch - 設定

適用をクリックして詳細を保存します。前回保存された状態に戻すには、**リセット**をクリックします。

インスタンスタイプ:	Sandbox (サンドボックス)/Development (開発) ⌄

廃止予定。HTTP と HTTPS のホスト名の構成では、サイトのエイリアス構成の新しい機能を使用することをお薦めします ("SEO > エイリアス構成")。
定する HTTP/HTTPS ホスト名の値は、エイリアス構成でホスト名が定義されていない場合に、古い設定スタイルをサポートする目的でのみ使用され

HTTP ホスト名:

HTTPS ホスト名:

インスタンスタイプ: すべて

カートリッジ: int_paidy_sfra:int_paidy:app_storefront_base

有効なカートリッジのパス:
int_paidy_sfra
int_paidy
app_storefront_base
plugin_apple_pay
plugin_facebook
plugin_payments
plugin_pinterest_commerce
plugin_web_payments
bc_content
core

int_paidy_sfra:int_paidy:merchant cartridge
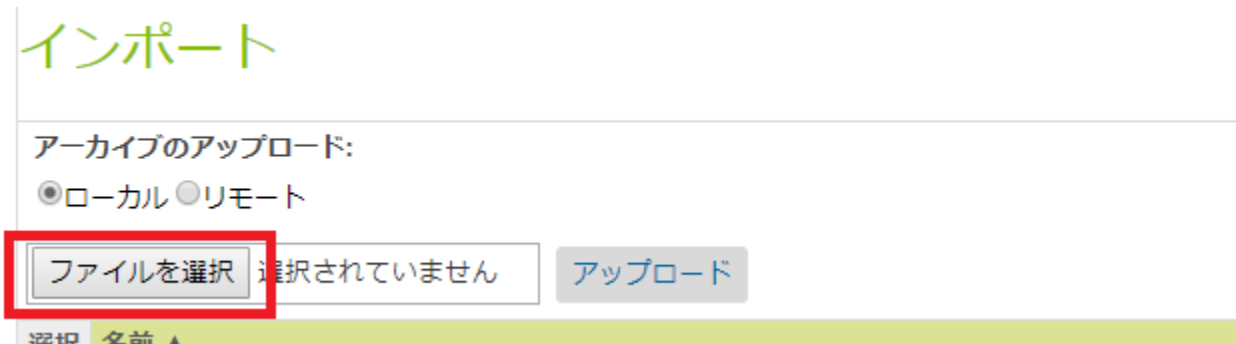
3-2-2. Importing metadata

Meta_data_22.1.0.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].

インポート

アーカイブのアップロード：
◉ローカル ◯リモート

| ファイルを選択 | 選択されていません | アップロード |

選択 名前 ▲

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

インポート
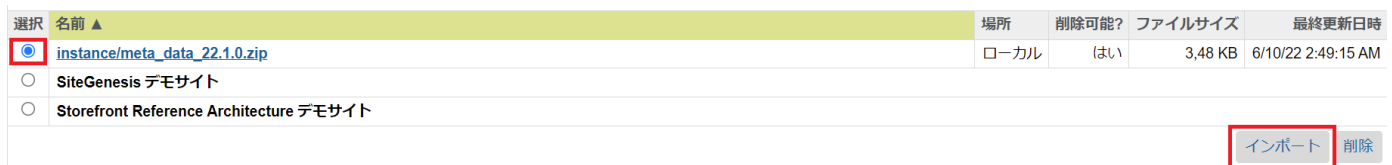
ファイルは、ローカルファイルである場合、またはファイルが作成されたインスタンスにログインしている場合にのみ削除できます。

アーカイブのアップロード：
◉ローカル ◯リモート

| ファイルを選択 | meta_data_22.1.0.zip | アップロード |

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

選択	名前 ▲	場所	削除可能?	ファイルサイズ	最終更新日時
◉	instance/meta_data_22.1.0.zip	ローカル	はい	3,48 KB	6/10/22 2:49:15 AM
◯	SiteGenesis デモサイト				
◯	Storefront Reference Architecture デモサイト				
				インポート	削除

Click [OK].

インポート

⚠ 選択したアーカイブをインポートしますか？   OK キャンセル

-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
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.

すべて選択	ID	名前
☐	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)

本番用APIキー

パブリックキー　　　　　　　　pk_live_

シークレットキー　　　　　　　sk_live_

テスト用APIキー

パブリックキー　　　　　　　　pk_test_

シークレットキー　　　　　　　sk_test_

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

すべて選択	ID	名前
☐	paidy_api_key	
☐	paidy_logo_url	
☐	paidy_secret_key	
☐	paidy_service_name	
☐	paidy_store_name	
☐	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.

すべて選択	ID	名前
☐	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.

マーチャントツール / サイト環境設定 /
カスタムサイト環境設定グループ ❓   サイト全体で表示 新規

Paidy 🔍 ▼

1～1 ▼ /1

ID	名前	説明	環境設定	サイト全体で表示
Paidy			6	表示

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

3-3. Modifying cartridges on the merchant

3-3-1. Add payment method

In order to display Paidy by payment method, write it in the template of the payment page.
  By writing this script, you can:
  ・Selection tab added to payment methods
  ・The explanation for consumers is displayed when Paidy payment is selected
  ・Tthe type of Paidy payment selected by the payment method is displayed on the order confirmation screen when selecting Paidy payment.
To embed a template, modify the following file.
 ・[app_storefront_base]/cartridge/templates/default/checkout/billing/paymentOptions/paymentOptionsTabs.isml
Add the below after </isif>.
....
```
<isif condition="${paymentOption.ID === 'PAIDY_STANDARD'}">
    <isinclude template="checkout/billing/paymentOptions/paidyStandardTab" />
</isif>
<isif condition="${paymentOption.ID === 'PAIDY_SUBSCRIPTION'}">
```

```
    <isinclude template="checkout/billing/paymentOptions/paidySubscriptionTab" />
</isif>
.........................
<isloop items="${pdict.order.billing.payment.applicablePaymentMethods}" var="paymentOption">
    <isif condition="${paymentOption.ID === 'CREDIT_CARD'}">
        <isinclude template="checkout/billing/paymentOptions/creditCardTab" />
    </isif>
    <isif condition="${paymentOption.ID === 'PAIDY_STANDARD'}">
        <isinclude template="checkout/billing/paymentOptions/paidyStandardTab" />
    </isif>
    <isif condition="${paymentOption.ID === 'PAIDY_SUBSCRIPTION'}">
        <isinclude template="checkout/billing/paymentOptions/paidySubscriptionTab" />
    </isif>
</isloop>
```


• [app_storefront_base]/cartridge/templates/default/checkout/billing/paymentOptions/paymentOpti
onsContent.isml
Add the below after </isif>.
....
```
<isif condition="${paymentOption.ID === 'PAIDY_STANDARD'}">
    <isinclude template="checkout/billing/paymentOptions/paidyStandardContent" />
</isif>
<isif condition="${paymentOption.ID === 'PAIDY_SUBSCRIPTION'}">
    <isinclude template="checkout/billing/paymentOptions/paidySubscriptionContent" />
</isif>
```
....
```
isloop items="${pdict.order.billing.payment.applicablePaymentMethods}" var="paymentOption">
    <isif condition="${paymentOption.ID === 'CREDIT_CARD'}">
        <isinclude template="checkout/billing/paymentOptions/creditCardContent" />
    </isif>
    <isif condition="${paymentOption.ID === 'PAIDY_STANDARD'}">
        <isinclude template="checkout/billing/paymentOptions/paidyStandardContent" />
    </isif>
    <isif condition="${paymentOption.ID === 'PAIDY_SUBSCRIPTION'}">
        <isinclude template="checkout/billing/paymentOptions/paidySubscriptionContent" />
    </isif>
</isloop>
```

・[app_storefront_base]/cartridge/templates/default/checkout/billing/paymentOptions/paymentOptionsSummary.isml

Add the below after </isif> and </div>.

....

(omission)

```
        <isif condition="${payment.paymentMethod === 'CREDIT_CARD'}">
            <isinclude template="checkout/billing/paymentOptions/creditCardSummary" />
        </isif>
        <isif condition="${payment.paymentMethod === 'PAIDY_STANDARD'}">
            <isinclude template="checkout/billing/paymentOptions/paidyStandardSummary" />
        </isif>
        <isif condition="${payment.paymentMethod === 'PAIDY_SUBSCRIPTION'}">
            <isinclude template="checkout/billing/paymentOptions/paidySubscriptionSummary" />
        </isif>
    </isloop>
</div>
<div class="payment-details-summary paidy-standard" style="display: none;">
    <isinclude template="checkout/billing/paymentOptions/paidyStandardSummary" />
</div>
<div class="payment-details-summary paidy-subscription" style="display: none;">
    <isinclude template="checkout/billing/paymentOptions/paidySubscriptionSummary" />
```

```
</div>
............................
<div class="payment-details">
    <isloop items="${pdict.order.billing.payment.selectedPaymentInstruments}" var="payment">
        <isif condition="${payment.paymentMethod === 'CREDIT_CARD'}">
            <isinclude template="checkout/billing/paymentOptions/creditCardSummary" />
        </isif>
        <isif condition="${payment.paymentMethod === 'PAIDY_STANDARD'}">
            <isinclude template="checkout/billing/paymentOptions/paidyStandardSummary" />
        </isif>
        <isif condition="${payment.paymentMethod === 'PAIDY_SUBSCRIPTION'}">
            <isinclude template="checkout/billing/paymentOptions/paidySubscriptionSummary" /
        </isif>
    </isloop>
</div>
<div class="payment-details-summary paidy-standard" style="display: none;">
    <isinclude template="checkout/billing/paymentOptions/paidyStandardSummary" />
</div>
<div class="payment-details-summary paidy-subscription" style="display: none;">
    <isinclude template="checkout/billing/paymentOptions/paidySubscriptionSummary" />
</div>
```

支払   編集

請求先住所:
100-0000
****
****
************* *************

██████████████████████
███████████████

支払:
Paidy翌月払い（コンビニ/銀行）

注文を確定する

・[app_storefront_base]/cartridge/scripts/checkout/checkoutHelpers.js
Add the below function.
....

```
/**
 * Selected payment instrument
 * @param {dw.order.Order} order - The order object to be placed
 * @returns {string|null} selected payment instrument
 */
function selectedPaymentInstrument(order) {
```

```
        if (order.billing.payment && order.billing.payment.selectedPaymentInstruments
            && order.billing.payment.selectedPaymentInstruments.length > 0) {
            return order.billing.payment.selectedPaymentInstruments[0].paymentMethod;
        } else {
            return null;
        }
    }
}
(omission)
    setGift: setGift,
    selectedPaymentInstrument: selectedPaymentInstrument
};
```

....

```
727
728  /**
729   * Selected payment instrument
730   * @param {dw.order.Order} order - The order object to be placed
731   * @returns {string|null} selected payment instrument
732   */
733  function selectedPaymentInstrument(order) {
734      if (order.billing.payment && order.billing.payment.selectedPaymentInstruments
735          && order.billing.payment.selectedPaymentInstruments.length > 0) {
736          return order.billing.payment.selectedPaymentInstruments[0].paymentMethod;
737      } else {
738          return null;
739      }
740  }
741
742  module.exports = {
743      getFirstNonDefaultShipmentWithProductLineItems: getFirstNonDefaultShipmentWithProductLineItems,
744      ensureNoEmptyShipments: ensureNoEmptyShipments,
745      getProductLineItem: getProductLineItem,
746      isShippingAddressInitialized: isShippingAddressInitialized,
747      prepareCustomerForm: prepareCustomerForm,
748      prepareShippingForm: prepareShippingForm,
749      prepareBillingForm: prepareBillingForm,
750      copyCustomerAddressToShipment: copyCustomerAddressToShipment,
751      copyCustomerAddressToBilling: copyCustomerAddressToBilling,
752      copyShippingAddressToShipment: copyShippingAddressToShipment,
753      copyBillingAddressToBasket: copyBillingAddressToBasket,
754      validateFields: validateFields,
755      validateCustomerForm: validateCustomerForm,
756      validateShippingForm: validateShippingForm,
757      validateBillingForm: validateBillingForm,
758      validatePayment: validatePayment,
759      validateCreditCard: validateCreditCard,
760      calculatePaymentTransaction: calculatePaymentTransaction,
761      recalculateBasket: recalculateBasket,
762      handlePayments: handlePayments,
763      createOrder: createOrder,
764      placeOrder: placeOrder,
765      savePaymentInstrumentToWallet: savePaymentInstrumentToWallet,
766      getRenderedPaymentInstruments: getRenderedPaymentInstruments,
767      sendConfirmationEmail: sendConfirmationEmail,
768      ensureValidShipments: ensureValidShipments,
769      setGift: setGift,
770      selectedPaymentInstrument: selectedPaymentInstrument
771  };
772
```

• [app_storefront_base]/cartridge/templates/default/checkout/billing/paymentOptions.isml
Add the below the first line of the file (after div.form-nav) and attribute in div.form-nav

....

```
    <isset name="paymentMethod" value="${require('*/cartridge/scripts/checkout/checkoutHelpers
').selectedPaymentInstrument(pdict.order)}" scope="page"/>
</isscript>
<div class="form-nav billing-nav payment-information"
    data-payment-method-id="CREDIT_CARD"
    data-selected-payment-method="${paymentMethod}"
    data-is-new-payment="${pdict.customer.registeredUser && pdict.customer.customerPaymentInstr
uments.length ? false : true}"
>
```

....

```
 1  <isset name="paymentMethod" value="${require('*/cartridge/scripts/checkout/checkoutHelpers').selectedPaymentInstrument(pdict.order)}" scope="page"/>
 2 ⊖<div class="form-nav billing-nav payment-information"
 3      data-payment-method-id="CREDIT_CARD"
 4      data-selected-payment-method="${paymentMethod}"
 5      data-is-new-payment="${pdict.customer.registeredUser && pdict.customer.customerPaymentInstruments.length ? false : true}"
 6  >
 7 ⊖    <ul class="nav nav-tabs nav-fill payment-options" role="tablist">
 8          <isinclude template="checkout/billing/paymentOptions/paymentOptionsTabs" />
 9      </ul>
10  </div>
11 ⊖<div class="credit-card-selection-new" >
12 ⊖    <div class="tab-content">
13          <isinclude template="checkout/billing/paymentOptions/paymentOptionsContent" />
14      </div>
15    </div>
```

3-3-2. CheckoutJS installation

In order to 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.

・[app_storefront_base]/cartridge/templates/default/common/layout/checkout.isml

Add the below after <head>

....

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <script type="text/javascript" src="https://apps.paidy.com/"></script>
        <!--[if gt IE 9]><!-->
```

....

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <script type="text/javascript" src="https://apps.paidy.com/"></script>

        <!--[if gt IE 9]><!-->
            <isinclude sf-toolkit="off" template="/common/scripts" />
```

3-3-3. Added paidy.js loading

Add the payid.js required for the payment process below.

・[app_storefront_base]/cartridge/templates/default/checkout/checkout.isml

Add the below after assets.addJs('/js/checkout.js');

....

```
<isscript>
    var assets = require('*/cartridge/scripts/assets.js');
    assets.addJs('/js/checkout.js');
    assets.addJs('/js/paidy.js');
    assets.addCss('/css/checkout/checkout.css');
</isscript>
```

....

3-3-4. Add attribute on order confirmation button

Define the information required for the payment process as the attribute of the settlement conf
irmation button.

・[app_storefront_base]/cartridge/templates/default/checkout/checkout.isml

Add the below in button.place-order

....

```
<button class="btn btn-primary btn-block place-order" data-action="${URLUtils.url('CheckoutServ
ices-PlaceOrder')}"
    data-get-paidy-standard-config="${URLUtils.url('PaidyStandard-GetPaidyConfig')}"
    data-paidy-standard-fail-order="${URLUtils.url('PaidyStandard-FailOrder')}"
    data-paidy-standard-place-order="${URLUtils.url('PaidyStandard-PlaceOrder')}"
    data-get-paidy-subscription-config="${URLUtils.url('PaidySubscription-GetPaidyConfig')}"
    data-paidy-subscription-set-token="${URLUtils.url('PaidySubscription-SetPaidyToken')}"
    type="submit" name="submit" value="place-order">${Resource.msg('button.place.order', 'check
out', null)}
</button>
```

....

3-3-5. CheckoutServices.js modifications

Select Paidy as the payment method, and do not process credit card payment in Ajax processing when transitioning to the order confirmation screen.

・[app_storefront_base]/cartridge/controllers/CheckoutServices.js

Add the below.

```
..........................
server.post(
    'SubmitPayment',
    server.middleware.https,
    csrfProtection.validateAjaxRequest,
    function (req, res, next) {
(omission)
        var validationHelpers = require('*/cartridge/scripts/helpers/basketValidationHelpers');
        var cartHelpers = require('*/cartridge/scripts/cart/cartHelpers');
        var paidyHelpers = require('*/cartridge/scripts/paidy/paidyHelpers');
        var currentBasket = BasketMgr.getCurrentBasket();
(omission)
        }
        paidyHelpers.resetPaymentForms(currentBasket);
        var billingAddress = currentBasket.billingAddress;
        var billingForm = server.forms.getForm('billing');
..........................
```



```
40  /**
41   * Handle Ajax payment (and billing) form submit
42   */
43  server.post(
44      'SubmitPayment',
45      server.middleware.https,
46      csrfProtection.validateAjaxRequest,
47      function (req, res, next) {
48          var PaymentManager = require('dw/order/PaymentMgr');
49          var HookManager = require('dw/system/HookMgr');
50          var Resource = require('dw/web/Resource');
51          var COHelpers = require('*/cartridge/scripts/checkout/checkoutHelpers');
52
                            (中略)
141          var validationHelpers = require('*/cartridge/scripts/helpers/basketValidationHelpers');
142          var cartHelpers = require('*/cartridge/scripts/cart/cartHelpers');
143          var paidyHelpers = require('*/cartridge/scripts/paidy/paidyHelpers');
144
145          var currentBasket = BasketMgr.getCurrentBasket();
146          var validatedProducts = validationHelpers.validateProducts(currentBasket);
147
148          var billingData = res.getViewData();
149
150          if (!currentBasket || validatedProducts.error) {
151              delete billingData.paymentInformation;
152
153              res.json({
154                  error: true,
155                  cartError: true,
156                  fieldErrors: [],
157                  serverErrors: [],
158                  redirectUrl: URLUtils.url('Cart-Show').toString()
159              });
160              return;
161          }
162
163          paidyHelpers.resetPaymentForms(currentBasket);
164
165          var billingAddress = currentBasket.billingAddress;
166          var billingForm = server.forms.getForm('billing');
167          var paymentMethodID = billingData.paymentMethod.value;
```

Add a process to be interrupted before proceeding the payment process when creating an order from Paidy with Ajax

Add the below.

....

```
server.post('PlaceOrder', server.middleware.https, function (req, res, next) {
    var BasketMgr = require('dw/order/BasketMgr');
(omission)
        var addressHelpers = require('*/cartridge/scripts/helpers/addressHelpers');
        var paidyHelpers = require('*/cartridge/scripts/paidy/paidyHelpers');
        var currentBasket = BasketMgr.getCurrentBasket();
(omission)
    if (handlePaymentResult.error) {
        res.json({
            error: true,
            errorMessage: Resource.msg('error.technical', 'checkout', null)
        });
        return next();
    }
    if (paidyHelpers.isPaidyStandard(currentBasket.paymentInstrument.paymentMethod)){
        var responseJson = require('*/cartridge/scripts/paidy/standard/authorize')
            .getConfirmationPaidyJSON(currentBasket.paymentInstrument.paymentMethod, order.getCustomer(), order);
        res.json(responseJson);
        return next();
    } else if (paidyHelpers.isPaidySubscription(currentBasket.paymentInstrument.paymentMethod)){
        var paidySubscriptionAuthorize = require(
            '*/cartridge/scripts/paidy/subscription/authorize')
            .Authorize(order, req.querystring.paidyToken);
        if (paidySubscriptionAuthorize.error) {
            Transaction.wrap(function () { OrderMgr.failOrder(order); });
            res.json({
                error: true,
                errorMessage: Resource.msg('error.technical', 'checkout', null)
            });
            return next();
        }
    }
    var fraudDetectionStatus = hooksHelper('app.fraud.detection', 'fraudDetection', currentBasket, require('*/cartridge/scripts/hooks/fraudDetection').fraudDetection);
```

....

```
540  server.post('PlaceOrder', server.middleware.https, function (req, res, next) {
541      var BasketMgr = require('dw/order/BasketMgr');
542      var OrderMgr = require('dw/order/OrderMgr');
543      var Resource = require('dw/web/Resource');
544      var Transaction = require('dw/system/Transaction');
545      var URLUtils = require('dw/web/URLUtils');
546      var basketCalculationHelpers = require('*/cartridge/scripts/helpers/basketCalculationHelpers');
547      var hooksHelper = require('*/cartridge/scripts/helpers/hooks');
548      var COHelpers = require('*/cartridge/scripts/checkout/checkoutHelpers');
549      var validationHelpers = require('*/cartridge/scripts/helpers/basketValidationHelpers');
550      var addressHelpers = require('*/cartridge/scripts/helpers/addressHelpers');
551
552      var paidyHelpers = require('*/cartridge/scripts/paidy/paidyHelpers');
553
554      var currentBasket = BasketMgr.getCurrentBasket();
555
```

────────────────────(中略)────────────────────

```
            errorMessage: Resource.msg('error.technical', 'checkout', null)
        });
        return next();
    }


    if (paidyHelpers.isPaidyStandard(currentBasket.paymentInstrument.paymentMethod)) {
        var responseJson = require('*/cartridge/scripts/paidy/standard/authorize')
            .getConfirmationPaidyJSON(currentBasket.paymentInstrument.paymentMethod, order.getCustomer(), order);
        res.json(responseJson);
        return next();
    } else if (paidyHelpers.isPaidySubscription(currentBasket.paymentInstrument.paymentMethod)) {
        var paidySubscriptionAuthorize = require('*/cartridge/scripts/paidy/subscription/authorize')
            .Authorize(order, req.querystring.paidyToken);
        if (paidySubscriptionAuthorize.error) {
            Transaction.wrap(function () { OrderMgr.failOrder(order); });
            res.json({
                error: true,
                errorMessage: Resource.msg('error.technical', 'checkout', null)
            });
            return next();
        }
    }

    var fraudDetectionStatus = hooksHelper('app.fraud.detection', 'fraudDetection', currentBasket, require('*/cartri
    if (fraudDetectionStatus.status === 'fail') {
```

・[app_storefront_base]/cartridge/scripts/cart/cartHelpers.js

Add the below function.

....

(omission)

```
/**
 * Deletes multiple payment instruments.
 *
 * @transactional
 * @alias module:models/CartModel~CartModel/removePaymentInstruments
 * @param {dw.order.Basket} basket - the target Basket object
 * @param {dw.util.Collection} paymentInstruments - The payment instruments to remove.
 */
function removePaymentInstruments(currentBasket, paymentInstruments) {
    collections.forEach(paymentInstruments, function(item) {
        currentBasket.removePaymentInstrument(item);
```

```
        });
    }
(omission)
    updateBundleProducts: updateBundleProducts,
    removePaymentInstruments: removePaymentInstruments
};
...........................

/**↓
 * Deletes multiple payment instruments.↓
 *↓
 * @transactional↓
 * @alias module:models/CartModel~CartModel/removePaymentInstruments↓
 * @param {dw.order.Basket} basket - the target Basket object↓
 * @param {dw.util.Collection} paymentInstruments - The payment instruments to remove.↓
 */↓
function removePaymentInstruments(currentBasket, paymentInstruments) {↓
    collections.forEach(paymentInstruments, function(item) {↓
        currentBasket.removePaymentInstrument(item);↓
    });↓
}↓
↓
module.exports = {↓
    addLineItem: addLineItem,↓
    addProductToCart: addProductToCart,↓
    checkBundledProductCanBeAdded: checkBundledProductCanBeAdded,↓
    ensureAllShipmentsHaveMethods: ensureAllShipmentsHaveMethods,↓
    getQtyAlreadyInCart: getQtyAlreadyInCart,↓
    getNewBonusDiscountLineItem: getNewBonusDiscountLineItem,↓
    getExistingProductLineItemInCart: getExistingProductLineItemInCart,↓
    getExistingProductLineItemsInCart: getExistingProductLineItemsInCart,↓
    getMatchingProducts: getMatchingProducts,↓
    allBundleItemsSame: allBundleItemsSame,↓
    hasSameOptions: hasSameOptions,↓
    BONUS_PRODUCTS_PAGE_SIZE: BONUS_PRODUCTS_PAGE_SIZE,↓
    updateBundleProducts: updateBundleProducts,↓
    removePaymentInstruments: removePaymentInstruments↓
};↓
```

3-3-6. Add payment method to email template

Add a payment method to the email sent when payment is completed.
・[app_storefront_base]/cartridge/templates/default/checkout/confirmation/confirmationPaymentInfo.isml
Add the below after </isif> in isloop of Payment information
....
(omission)

```
                        <span>${Resource.msg('msg.card.type.ending', 'confirmation', null)} ${payment.expirationMonth}/${payment.expirationYear}</span>
                    </div>
                </isif>
                <isif condition="${payment.paymentMethod === 'PAIDY_STANDARD'}">
                    <isinclude template="checkout/billing/paymentOptions/paidyStandardSummary" />
                </isif>
                <isif condition="${payment.paymentMethod === 'PAIDY_SUBSCRIPTION'}">
```

```
                    <isinclude template="checkout/billing/paymentOptions/paidySubscriptionSumm
ary" />
                </isif>
            </isloop>
        </div>
    </div>
...........................
```

```
<isloop items="${pdict.order.billing.payment.selectedPaymentInstruments}" var="payment">
    <isif condition="${payment.paymentMethod === 'CREDIT_CARD'}">
        <div>
            <span>${Resource.msg('msg.payment.type.credit', 'confirmation', null)}
                ${payment.type}</span>
        </div>
        <div>
            ${payment.maskedCreditCardNumber}
        </div>
        <div>
            <span>${Resource.msg('msg.card.type.ending', 'confirmation', null)} ${payment.expirationMonth}/${payment
        </div>
    </isif>
    <isif condition="${payment.paymentMethod === 'PAIDY_STANDARD'}">
        <isinclude template="checkout/billing/paymentOptions/paidyStandardSummary" />
    </isif>
    <isif condition="${payment.paymentMethod === 'PAIDY_SUBSCRIPTION'}">
        <isinclude template="checkout/billing/paymentOptions/paidySubscriptionSummary" />
    </isif>
</isloop>
```

3-4. Advice

- Notes on use of subscription purchases
  If you followed the steps to add a payment method as indicated in section 3-3-1, 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翌月払い(コンビニ/銀行)」.

- Notes on tokens
int_paidy_sfra does not use PaidyToken for customer information.
Tokens can be checked via order data.

PaidyPayments

paidyToken:

paidyPaymentId:

- Order confirmation e-mails for subscription purchases
int_paidy_sfra 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_sfra cartridge.

4. Cartridge deployment testing at merchant

4-1. Before beginning a test

4-1-1. Paidy merchant dashboard 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.

本番用APIキー	
パブリックキー	pk_live_
シークレットキー	sk_live_

テスト用APIキー	
パブリックキー	pk_test_
シークレットキー	sk_test_

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 dashboard screen.

名前	値	デフ
paidy_api_key	pk_test_	
paidy_logo_url		
paidy_secret_key	sk_test_	
paidy_store_name		
paidy_service_name		
paidy_token_description		

4-2. PaidyCheckout test data

-mail address and phone number

E-mail address and telephone usable for test cases

Test cases	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	Android4.4 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

Test as a member.

1. Success as a member
- When logging in as a member 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 as a member
- 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

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
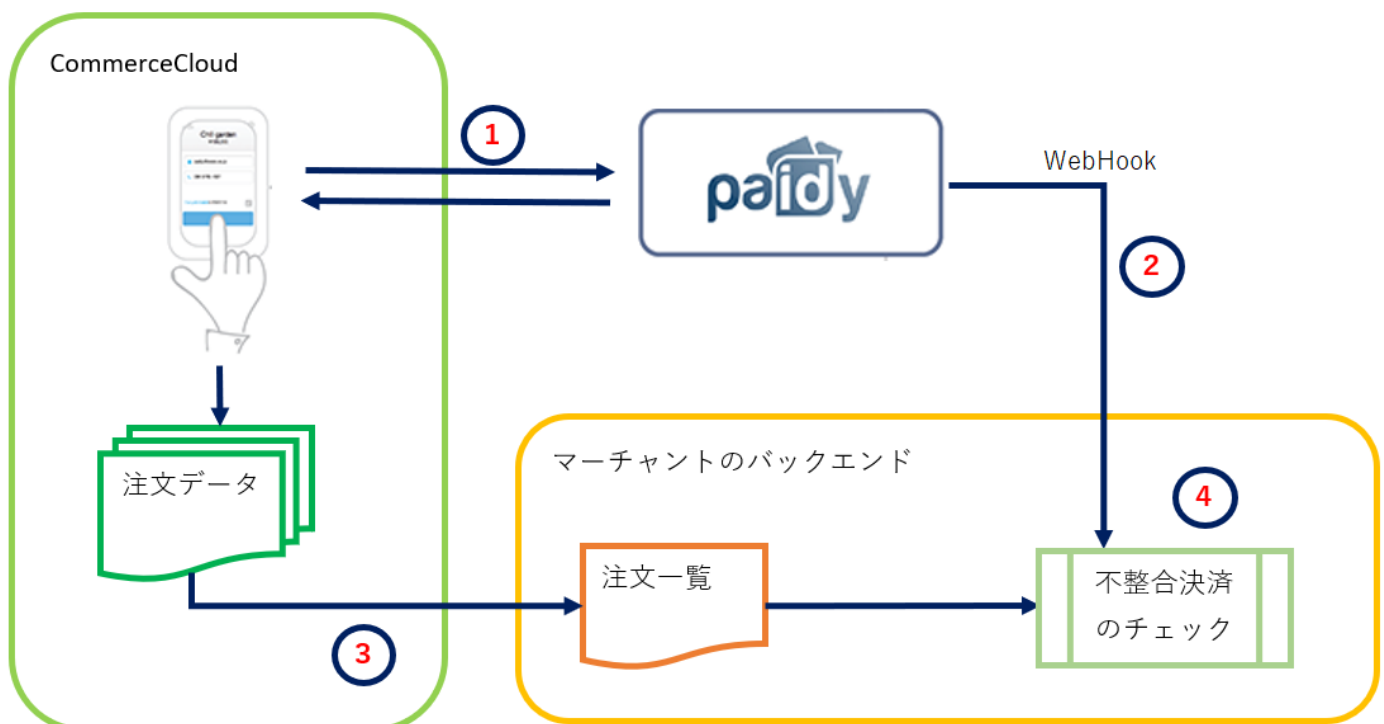
1. Payment
    Payment is made via Commerce Cloud using Paidy. A callback from Paidy's payment results returns one of the three order statuses below.
    - Payment success: New   Outputted to the order list. Payment possible status

- 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

Merchant Tools > Ordering > Orders > Order: WBSC00005100(Paidy_Master)

General　Attributes　Payment　Notes　History

Details for Order 'WBSC00005100'

Information:	Contains 1 line item to 1 shipping location The total price is ¥14,506.00.		
Date Received:	11/24/17 6:27:23 am Etc/UTC		
Site:	Paidy_Master		
Created By:	Customer		
Customer:	純 玉水		
Customer No.:	00033017		
IP Address:	n/a		
Email:	jun.tamamizu@wiredbeans.co.jp		
Phone:	022-380-8680		
Order Status:	Open	Confirmation Status:	Confirmed
Shipping Status:	Not Shipped	Export Status:	Ready for Export

The status of the above three points is changed per below

 Order Status : Created→Open

 Confirmation Status : Not Confirm→Confirmed

 Export Status:Not Exported→Ready for Export

・Attribute 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)

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	支払方法	支払金額合計	
日本 980-0014	Paidy翌月払い (コンビニ/銀行)	小計	￥ 1,808
宮城県 仙台市青葉区 本町	金額:￥ 2,492	配送 通常配送 ￥ 500	
玉水 純			

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

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
1.1.0	2019/12/06	First edition
20.1.0	2020/01/15	3-2-2. Importing metadata -Modify 「meta_data_19.1.0.zi

		p」to「meta_data_20.1.0.zip」
20.1.0	2020/4/5	-SFRAのバージョン情報を追記。 -Business Managerの互換モードを記載。 -Add Version of SFRA and Compatibility Mode -Add 6. What to do when the service is down
21.1.0	2021/6/21	-Update 1. Cartridges version update -Update 4.SFRA support version update -Update 16-17. int_paidy post script -Update 18. Import meta file name update Deleted description related to SiteGenesis and common meta file -Update 25. Corrected the description of function selected Payment Instrument - Update 30. Embedded code fix - Update 31. Embedded code fix - Update 32. File name Change confirmationEmail.isml to confirmationPaymentInfo.isml -Removed 35. Deleted the description of package.json
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 4.SFRA and compatibility mode support version update -Update 18. Import meta file name update - Update 37. Delete IE11 from Browser compatibility