

1. LINK Cartridge Documentation Updates	2
1.1 Adyen LINK Cartridge Integration Guide	3
1.2 Summary	4
1.3 Component Overview	5
1.3.1 Functional Overview	6
1.3.2 Use Cases	7
1.3.3 Limitations, Constraints	26
1.3.4 Compatibility	27
1.3.5 Privacy, Payment	28
1.4 Implementation Guide	29
1.4.1 Set Up	30
1.4.2 Configurations	33
1.4.2.1 Business Manager Configuration	34
1.4.2.2 Adyen Account Configuration	37
1.4.3 Custom Code	38
1.4.3.1 Integration Efforts	39
1.4.3.2 Common Changes	40
1.4.3.3 Controller Model: Changes Needed	66
1.4.3.4 Pipeline Model: Changes Needed	81
1.4.3.5 Additional Configuration	82
1.4.3.6 Custom Objects	83
1.4.4 External Interfaces	84
1.4.5 Testing	85
1.5 Operations, Maintenance	87
1.5.1 Data Storage	88
1.5.2 Availability	89
1.5.3 Support	90
1.5.4 Scheduled Jobs	91
1.6 User Guide	96
1.6.1 Roles and Responsibilities	97
1.6.2 Business Manager	98
1.6.3 Storefront Functionality	99
1.7 Known Issues	100
1.8 Release History	101
1.9 Version History	102

LINK Cartridge Documentation Updates

This section of pages represents the updates that will be made to the Adyen LINK Cartridge documentation.

Content in the following sections will either replace or be merged with the existing documentation upon certification/release.



Summary

This cartridge enables a Commerce Cloud store to use the Adyen payment service.

The developer has to install the cartridge and integrate it into online store following the instructions from this document. It is mandatory that the merchant has to open an account with Adyen and configure some items in Commerce Cloud Business Manager as well as in the Adyen back office to make the integration working properly.

Adyen offers a test environment, so the integration can be tested before switching to the Adyen production environment.

The integration consists of an archive, which contains the following contents:

- Cartridge called 'int_adyen', and optionally either "int_adyen_pipelines" or "int_adyen_controllers" depending on which architecture is to be employed.
- An extension cartridge to the Business Manager 'bm_adyen', which enables the display of Adyen specific information in order details
- A site-template archive containing new attributes and settings
- An example 'skin', which is used for styling the Adyen hosted payment page (HPP)
- An archive with standard storefront cartridges with Adyen integrated. They can be used as an example for the integration and testing the cartridge
- This document

The integration is based on the Site Genesis demo store provided by Commerce Cloud.

Component Overview

Functional Overview

Short description of the Adyen payment system

Adyen can receive payments in 2 ways:

1. via the Adyen API using SOAP or POST. A result is returned right away. This can only be used for Credit Cards. The cartridge uses POST via the Adyen REST API
2. via the 'redirect model' where a form containing certain fields is posted to Adyen and the shopper is redirected to Adyen's Hosted Payment Page (HPP) . After completing the payment, the shopper is returned to the shop's resultURL. This resultURL can be configured and contains the result of the payment (success or failed)

Compared to the API, the redirect model offers more payments methods as well as some other functionality, including reduced PCI SAQ exposure.

However, if you decide to handle credit card payments yourself, you can use API payments, which are implemented in the cartridge. The stored credit card data could be used for returning customers, saving them the trouble of entering their details again.

Payment results are sent asynchronously to a 'notification URL' at the Merchant (SCC) site. Payments can change status over the time (Authorised, Refunded, Cancelled and others) on Adyen side, and these statuses are updated automatically on the Commerce Cloud side through sending these notifications.

In both cases the 'two phase ordering' approach is used, where the order is first created but only 'placed' after a payment has been authorized (which is now standard behavior in Site Genesis). In case if the payment cannot be authorized, the order will fail.

Use Cases

Registered and unregistered shoppers will be able to follow the standard Site Genesis flow with the following changes:

- Adyen redirect, Adyen API and Gift Certificates (GC) will be the only payment options
- if Adyen redirect is enabled:
 - no entering of credit card details on the Billing page is possible when the user selects this payment method (as this is done on the Adyen HPP) and Directory Lookup feature is disabled
 - the possibility to choose one of the predefined payment methods on the Billing page with further redirection to Adyen HPP, if Directory Lookup is enabled and the user selected Adyen payment method
 - a redirect to Adyen HPP after clicking the 'SUBMIT ORDER' button on the Order Confirmation page (only if no GCs are used and/or they don't cover the whole amount)
 - a return to the SCC Order Summary page after successful payment authorization
 - a return to the SCC Order Confirmation page, if the user cancelled the payment on Adyen HPP
 - a return to the SCC Order Confirmation page with an error message displayed, if the payment was refused
- if Adyen API payments are enabled:
 - credit card details are entered and stored in your SCC shop
 - for the other payment methods you can still redirect to the Adyen HPP, but can disable credit cards via the configuration of your skin
 - if the AVS feature is enabled, the user will see 2 new fields 'Suite' and 'Street' (or 'Building number' and 'Street name') on all address forms of the site, instead of field 'Address 1'

Detailed Use Cases

Use Case: Redirect to Adyen HPP

This use case describes a redirect to the Adyen Payment page (HPP). Make sure the settings are correct (see screenshot below). Usually Payment Selection is set to 'one' to select the one-page flow on the Adyen HPP. If the debug flag is checked, you will see a button before the redirect, which gives you a possibility to inspect the contents of the form being sent to Adyen.

The screenshot shows the 'Merchant Tools' interface for 'Sandbox - adyen01 LyonsSiteGenesis'. The 'Instance Type' is set to 'Sandbox'. The configuration table is as follows:


Name	Value
merchantCode	LyonsCOM
Recurring Type	ONECLICK (ONECLICK)
skinCode	tQvp5MMX
AdyenMerchantCode	
Adyen Notifications Password	tester
	Adyen Notifications Password for basic Authentication
Adyen Notifications User	tester
	Adyen Notifications User for Basic Authentication
Url to the pal service	
Password for the pal service	
User for the pal service	


Select 'Adyen' as the payment method on Billing page of checkout process and click the 'CONTINUE' button:


SELECT PAYMENT METHOD • REQUIRED

☒ Adyen
☐ Credit Card
☐ Pay Pal
☐ Bill Me Later

- ☒


American Express
- ☐



MasterCard
- ☐


VISA

CONTINUE TO PLACE ORDER >

You will see the Order Confirmation page and can press the 'SUBMIT ORDER' button:

STEP 1: Shipping > STEP 2: Billing > **STEP 3: Place Order**

PRODUCT	QTY	TOTAL
 <div> Women's Hypertrail Low with Gore-Tex XCR® Membrane Item No.: 883239315158 Color Light Brown Width M Size 4 </div>	1 In Stock	\$76.00
Subtotal		\$76.00
Shipping Standard		\$6.99
Sales Tax		\$4.15
Order Total		\$87.14

« Edit Cart

PLACE ORDER

If debug is enabled, you will see an empty page with a small button

sa,amex,maestrouk,so
amex
USD
LyonsCOM
00002301
albR2w9jnwYtb21UnD1aa
8714
ONECLICK
https://adyen01.tech-prtnr-e
2016-10-24T17:08:00.966Z
grnwood@gmail.com
bc7938zieJxzmiAAe7VasW
tQvp5MMX

pay

Additionally you will be able to check the form that will be submitted to Adyen via Firebug or an equivalent tool, or just by checking the source code of the page


```

<form name="adyenForm" id="adyenForm" action="https://test.adyen.com/hpp/skipDetails.shtml" method="post">
  <input type="text" name="allowedMethods" value="mc,visa,amex,maestrouk,solo,ideal,elv,paypal,sepadirectdebit,bankTransfer_IBAN">
  <br>
  <input type="text" name="brandCode" value="amex">
  <br>
  <input type="text" name="currencyCode" value="USD">
  <br>
  <input type="text" name="merchantAccount" value="LyonsCOM">
  <br>
  <input type="text" name="merchantReference" value="00002301">
  <br>
  <input type="text" name="merchantsSig" value="aIBR2w9jnwYtb21UnD1aa64mK50B2K11IPPRJ9qj28">
  <br>
  <input type="text" name="paymentAmount" value="8714" => $0
  <br>
  <input type="text" name="recurringContract" value="ONECLICK">
  <br>
  <input type="text" name="resURL" value="https://adyen01.tech-prtnr-eu01.dw.demandware.net/on/demandware.store/Sites-LyonsSiteGenesis-Site/default/Adyen-ShowConfirmation">
  <br>
  <input type="text" name="sessionValidity" value="2016-10-24T17:08:00.966Z">
  <br>
  <input type="text" name="shopperEmail" value="grnwood@gmail.com">
  <br>
  <input type="text" name="shopperReference" value="bc7938zieJxzmiAAe7VasWRJ64">
  <br>
  <input type="text" name="skinCode" value="tQvp5MMX">
  <br>
  <input type="submit" value="pay" style="margin-left:200px">
</form>
<!-- Demandware Analytics code 1.0 (body_end-analytics-tracking-async.js) -->
<script type="text/javascript" src="/on/demandware.static/Sites-LyonsSiteGenesis-Site/-/default/v1476352373824/internal/js/script/dwanalytics-16.9.js" async="async" onload="trackPage()"></script>
<script type="text/javascript"></script>
<!-- Demandware Active Data (body_end-active.data.js) -->

```

If debug is not enabled, you will be redirected to Adyen HPP immediately, and will see a list of payment methods (the list depends on the chosen countries and the settings you applied)

Step 2: Enter your Payment Details

You are paying USD 87.14 with 

Card Number	<input type="text"/>	⚠
Card Holder Name	<input type="text"/>	⚠
Card Expiry Date	<input type="text"/> / <input type="text"/>	⚠
CID	<input type="text"/> What is CID?	⚠

Card Number invalid or missing
Card Holder Name missing
Expiry Month missing
Expiry Year missing
CID missing

Next Step: Review and Complete Your Payment

[previous](#)

[continue](#)

After the successful payment authorization, you will be redirected back to the SCC Order Summary page (should be configured in the Adyen back office):

You can check the result of the payment authorization in Business Manager. On the payment tab of the order detail screen, the payment status should be Paid and the Adyen Eventcode should be AUTHORISATION (see the screenshot below):

Sandbox - Instance Log

Sites: [\(4\)](#) **Site Genesis**

- Site - Site Genesis
 - Storefront
 - Products and Catalogs
 - Content
 - Search
 - Online Marketing
 - Customers
 - Custom Objects
 - Ordering**
 - Analytics
 - Site URLs
 - Site Preferences
 - Administration

[Site](#) > [Ordering](#) > [Orders](#) > Order: ad01-00006580

General Attributes **Payment** Notes History


Payment Information for Order 'ad01-00006580'

Order Total:	\$105.87
Amount Paid:	\$0.00
Balance Due:	\$105.87

Invoice Number:	00021604
Payment Status:	Paid

Payment Method: Adyen [Billing Address](#)

Processor: Adyen
Transaction:
Amount: \$105.87



Payment info	
PSP reference	7914204984750298
Payment Method	mc
Eventcode	AUTHORISATION
Amount	105.87

If the payment authorization was refused, you will be redirected back to the SCC Order Confirmation page and an error message will be shown:

salesforce commerce cloud 📍 👤 🛒

[NEW ARRIVALS](#) [WOMENS](#) [KIDS](#) [MENS](#) [ELECTRONICS](#) [TOP SELLERS](#)

STEP 1: Shipping > **STEP 2: Billing** > STEP 3: Place Order

SELECT OR ENTER BILLING ADDRESS • REQUIRED

• First Name

• Last Name

• Address 1

• Suite

ORDER SUMMARY [Edit](#)

3/4 Sleeve Button Front Cardigan
Color Multi
Size S
Qty: 1 \$54.99

Subtotal	\$54.99
Edit Shipping Standard	\$6.99
Sales Tax	\$3.10
Order Total:	\$65.08

SHIPPING ADDRESS [Edit](#)

In Business Manager, the order will be marked as failed in this case:

Order Search

Simple

Advanced

By Number

Order Number:

Find

Number	Order Date	Created By	Registration Status	Customer	Email	Total	Status
ad01-00006580	1/5/15 11:54:21 pm	Customer	Registered	Denis Developer	developer.12321@gmail.com	\$105.87	Open
ad01-00006579	1/5/15 11:51:52 pm	Customer	Registered	Denis Developer	developer.12321@gmail.com	\$105.87	Failed
ad01-00006578	1/5/15 11:50:26 pm	Customer	Registered	Denis Developer	developer.12321@gmail.com	\$105.87	Created
ad01-00006577	1/5/15 11:49:30 pm	Customer	Registered	Denis Developer	developer.12321@gmail.com	\$105.87	Failed

Orders can fail for many reasons, and in order to check that the payment was refused, you should check the notifications tab of the order details page. If the Adyen payment method was used, the notification should be similar to the one provided below:

General
Attributes
Payment
Notes
History

Notes for Order 'ad01-00006974'

Fields with a red asterisk (*) are mandatory. Click Add to save a new note.

Subject:

Text:

Select All	User	Time	Note
<input type="checkbox"/>	storefront	1/6/15 7:22:09 pm	Adyen Payment Notification AdyenHppRefusedPayment v 45 - Payment info <hr/> merchantReference : ad01-00006974 skinCode : byS01r6C shopperLocale : en_GB authResult : REFUSED pagReference : 8614205485294326 paymentMethod : mc

There may be a few notifications, you should check the newest one.

Refused payments always have the Eventcode equals to AUTHORISATION, regardless of the payment method used:

General Attributes **Payment** Notes History

Payment Information for Order 'ad01-00006974'

Order Total: \$105.87
 Amount Paid: \$0.00
 Balance Due: \$105.87

Invoice Number:
 Payment Status: Not Paid

Payment Method: Adyen
 Processor: Adyen
 Transaction:
 Amount: \$105.87



Payment info

PSP reference	8614205685296326
Payment Method	mc
Eventcode	AUTHORISATION
Amount	105.87

In the Adyen back office, refused payments always have the appropriate status:

8614207067403165	ad01-00007075	2015-01-08 10:45:40 EET	USD	105.87		Refused	1
8614207065791058	ad01-00007074	2015-01-08 10:42:59 EET	USD	105.87		Refused	1
8614205685296326	ad01-00006974	2015-01-06 20:22:09 EET	USD	105.87		Refused	1

Use Case: Cancel Payment on Adyen HPP

If you click on the button 'previous' on the Adyen HPP, you will be redirected back to the SCC order confirmation page. It is a standard order confirmation page, you can to continue work with the site as usually you may return to any step of checkout process or go back to Adyen HPP by clicking 'SUBMIT ORDER' button.



Step 1: Please select your payment method

Total payment amount USD 105.87

Card Number

Card Holder Name

Card Expiry Date

CVC/CVV/CID

Remember these details

Next Step: Enter your Payment Details

previous

Every time when you go to the Adyen HPP, a new order is created in BM, and when you return back to SCC, the order status will be failed. There is no any additional info in BM about the fact that the payment was cancelled. The payment does not appear in the Adyen back office if it was cancelled in this way.

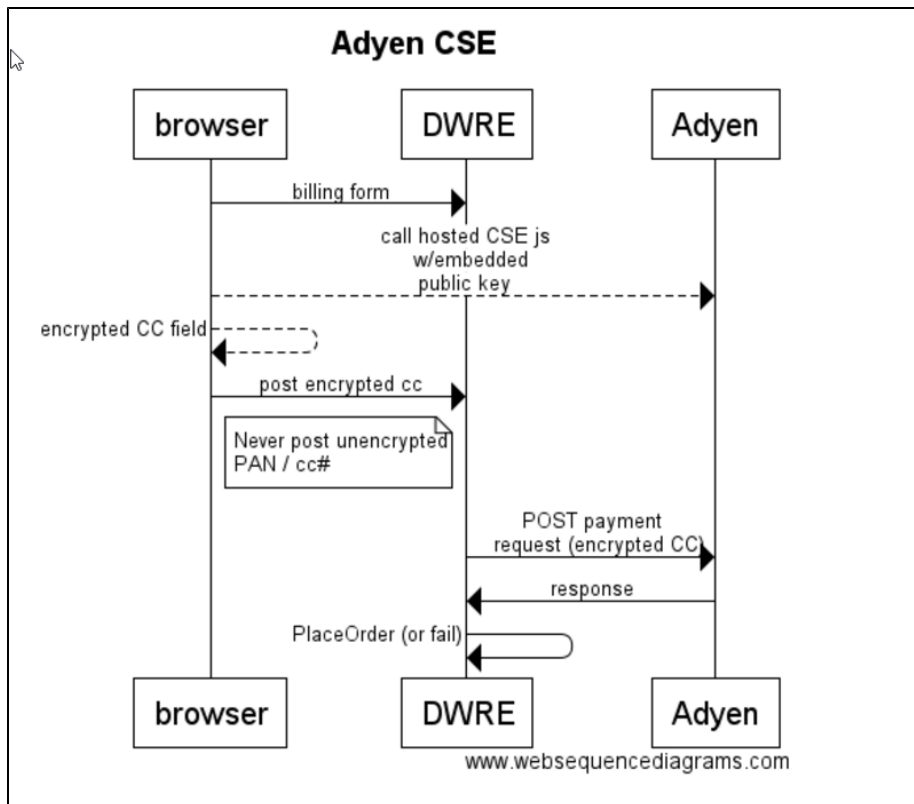
Use Case: Client Side Encryption (CSE)

Adyen offers CSE / Easy Encryption via Javascript on the client side to accept payments while the payment data is encrypted in the shopper's browser or application. This integration provides stricter security protocols and you do not need to manage the [PCI compliance](#).

There are three ways that CSE can be used:

1. **Adyen Hosted** ★
 - a. In this case an Easy Encryption library is hosted by Adyen with your public key injected into JavaScript code. You retrieve this library from the unique URL provided to you by Adyen, and thus you don't need to download and host the `adyen.encrypt.min.js` on your own servers.

To obtain this URL, log in the [Adyen Customer Area \(CA\)](#) using your company-level account, go to Settings / Users, filter the list of users by System in the drop-down box, and then open the Web Service (ws) user page. On this page you can find the Library location on the Easy Encryption pane, or (if the library is not available yet), click Generate and then Save (at the page bottom) to create a new URL.
2. **Merchant Hosted**
 - a. This integration binds to existing HTML in the page, adding a hidden input containing the encrypted card data to the form on the moment the form is submitted. The complete integration requires HTML markup to be present in the page, as well as accompanying JavaScript to enable the behaviour.
3. **Javascript Only**
 - a. In case the HTML integration does not fulfill your setup requirements, the library has been split up into two parts since release V0_1_11. The newly introduced part is an HTML-independent encryption.



★ The Adyen preferred method of integration is the Adyen Hosted which is the method assumed in the LINK cartridge. If you wish to utilize Merchant Hosted and/or Javascript only cartridge modifications will be necessary and are beyond the scope of this document. More information can be obtained here:

- Merchant Hosted: <https://docs.adyen.com/developers/easy-encryption#merchanthosted>
- Javascript only: <https://docs.adyen.com/developers/easy-encryption#javascriptonly>

Configuration

Please configure the following site preferences:

Name	Id	Type / Description
CSE Enabled	cseEnabled	Boolean, default to true.
CSE Test Javascript URL	cseTestJsURL	String, default to empty.
CSE Prod Javascript URL	cseProdJsURL	String, default to empty.

To obtain the CSE Hosted Javascript URL, go into the Adyen console Settings/Users, click on the user and copy the latest URL. This URL value should be configured into the Site Preference.

User Selection	.
Copy CSE URL:	.
Resolves to properly configured JS for client account:	.

Use Case: Credit Card (direct to API).

This use case describes the credit card payments, when the card details are entered on the SCC side and are sent to Adyen via the Adyen API.

Enter the credit card details on the billing page of the checkout process and click the 'CONTINUE' button:

SELECT PAYMENT METHOD • REQUIRED

☐ Adyen ☒ Credit Card ☐ Pay Pal ☐ Bill Me Later

• Name on Card

Joe Greenwood

• Type

Visa

• Number

4111111111111111

Example: 4111111111111111

• Expiration Date:

September

2020

• Security Code


737


What is this?




CONTINUE TO PLACE ORDER >

You will see the order confirmation page. The credit card details will be shown on the lower right of the page. You can press the Place Order button in order to complete the purchase.

If the payment authorization was successful, you will see the 'Order Summary' page with information about the payment:


 commerce cloud

Enter Keyword or Item No. 

NEW ARRIVALS WOMENS KIDS MENS ELECTRONICS TOP SELLERS

Thank you for your order.

 Order Number: 00002306

Order Placed: Oct 24, 2016

PAYMENT METHOD
Credit Card
joe greenwood
Visa
.....1111
Amount: \$65.08

BILLING ADDRESS
Joseph Greenwood
7713 eastwood ct. apt 1, apt 1
apt 1
Schiller Park, IL 60176
United States
Phone: 7739688560

SHIPMENT NO. 1

SHIPPING STATUS:

SHIPPING TO

PAYMENT TOTAL

Subtotal	\$54.99
Shipping Standard	\$6.99
Sales Tax	\$3.10
Order Total:	\$65.08

CREATE ACCOUNT

Creating an account is easy. Just fill out the form below and enjoy the benefits of being a registered customer.

• First Name

Joseph

• Last Name

Greenwood

• Email

grnwood@gmail.com


• Confirm Email

You can check the result of the payment authorization in Business Manager. On the payment tab of the order details screen, the payment status should be Paid and the Adyen Eventcode should be AUTHORISATION:

General Attributes **Payment** Notes History

Payment Information for Order 'ad01-00007374'


Order Total:	\$105.87
Amount Paid:	\$0.00
Balance Due:	\$105.87
Invoice Number:	00024104
Payment Status:	Paid
Payment Method:	CREDIT_CARD Processor: ADYEN_CREDIT Transaction: Amount: \$105.87



Payment info	
PSP reference	8514207360609279
Payment Method	bicard
Eventcode	AUTHORISATION
Amount	105.87

If payment authorization was refused, you will see the order confirmation page where the following error message will be displayed:


salesforce commerce cloud

Enter Keyword or Item No. 

NEW ARRIVALS WOMENS KIDS MENS ELECTRONICS TOP SELLERS

STEP 1: Shipping > STEP 2: Billing > **STEP 3: Place Order**

We're sorry that your order could not be placed. This probably happened due to a high order volume or temporary connection errors. Please wait a few minutes and resubmit your order. We won't process your payment until you successfully place your order. If you have further questions, please contact us.

PRODUCT	QTY	TOTAL
 Women's Hypertrail Low with Gore-Tex XCR® Membrane Item No.: 883239315158 Color Light Brown Width M Size 4	1 In Stock	\$76.00
Subtotal		\$76.00
Shipping Standard		\$6.99
Sales Tax		\$4.15
Order Total		\$87.14

« Edit Cart **PLACE ORDER**

In Business Manager, the order will be marked as failed in this case:

Number	Order Date	Created By	Registration Status	Customer	Email	Total	Status
ad01-00007375	1/8/15 6:07:12 pm	Customer	Registered	Denis Developer	developer.12321@gmail.com	\$105.87	Failed
ad01-00007374	1/8/15 5:54:19 pm	Customer	Registered	Denis Developer	developer.12321@gmail.com	\$105.87	Open

Refused payments always have an Eventcode that equals to AUTHORISATION, regardless of the payment method used:

General Attributes **Payment** Notes History

Payment Information for Order 'ad01-00006974'

Order Total:	\$105.87
Amount Paid:	\$0.00
Balance Due:	\$105.87
Invoice Number:	
Payment Status:	Not Paid
Payment Method:	Adyen Processor: Adyen Transaction: Amount: \$105.87

Payment info

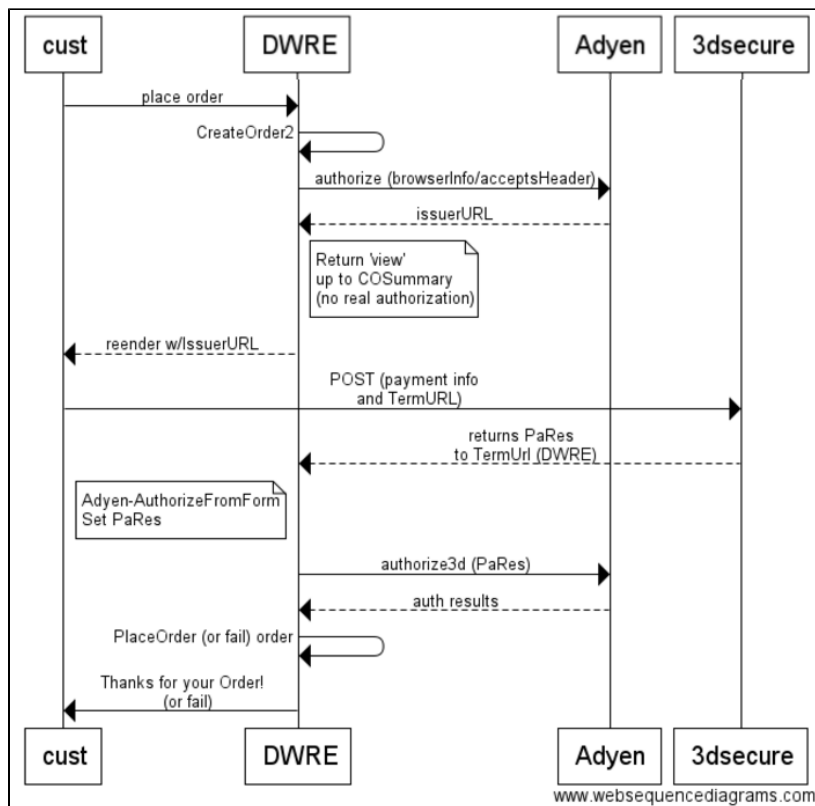
PSP reference	8614205685296326
Payment Method	mc
Eventcode	AUTHORISATION
Amount	105.87

In the Adyen back office, refused payments always have the appropriate status:

8614207067403165	ad01-00007075	2015-01-08 10:45:40 EET	USD	105.87		Refused	1
8614207065791058	ad01-00007074	2015-01-08 10:42:59 EET	USD	105.87		Refused	1
8614205685296326	ad01-00006974	2015-01-06 20:22:09 EET	USD	105.87		Refused	1

Use Case: 3-D Secure Credit Card

3-D Secure (Verified by VISA / MasterCard SecureCode™) is an additional authentication protocol that involves the shopper being redirected to their card issuer where they authenticate themselves before the payment can proceed to an authorization request.



You can find the 3-D Secure test credit card numbers here: <https://docs.adyen.com/support/integration#testcardnumbers>

For placing an order you have to follow the same steps like usual, the difference is that before you will be directed to the order confirmation page you will be prompted to enter your 3-D Secure credit card credentials.

The order payment details can be found on the "Payment" tab from the order details page in Business Manager > Site > Ordering > Orders >

Orders. Here you can find the information in the Adyen payment info section the PSP reference number with a link for the corresponding payment entry stored in the Adyen Customer Area (CA).

In the CA you can find the details about the payments in general, but also specific information about 3-D Secure payments like 3-D Secure Offered (yes or no) and 3-D Secure Authenticated (yes or no).

Note. For 3-D Secure, no additional changes of the storefront are required.

This use case describes the 3-D Secure credit card payments, when the card details are entered on SCC side and are sent to Adyen via the Adyen API and PAL adapter. Make sure the settings for Adyen PAL adapter are correct.

Navigate to the place order page following the usual credit card checkout steps. Click on the 'SUBMIT ORDER' button to be directed to the 3-D Secure Authentication Page.

salesforce

commerce cloud

Enter Keyword or Item No.

NEW ARRIVALS

WOMENS

KIDS

MENS

ELECTRONICS

TOP SELLERS

D3D

Demo page

Authenticate a transaction

Amount

USD 144.88

To be paid to

LyonsCOM

With card number

4212345678901237

User Name:

Password:

Submit

You have to enter your 3-D Secure credit card credentials there.

If you enter the correct credentials in the 3-D Secure authentication page, you will be directed to the order confirmation page. If these credentials are wrong you will be redirected back to the place order page and you will see an error message on top.

Good Order	Bad Order
------------	-----------

[NEW ARRIVALS](#)
[WOMENS](#)
[KIDS](#)
[MENS](#)
[ELECTRONICS](#)
[TOP SELLERS](#)

Thank you for your order.

Order Number: 00002402

Order Placed: Oct 24, 2016

PAYMENT METHOD
 Credit Card
 de greenwood
 /isa
 **** * 1237
 Amount: \$144.88

BILLING ADDRESS
 Jamen Semirero
 133 Hudson St
 apt 1
 New York, NY 10013
 United States
 Phone: 7735551212

PAYMENT TOTAL
 Subtotal \$130.99
 Shipping Standard \$6.99
 Sales Tax \$6.90
Order Total: \$144.88

SHIPMENT NO. 1

SHIPPING STATUS:
 Not Shipped
METHOD:
 Standard

SHIPPING TO
 Joseph Greenwood
 9913 eastwood ct, apt 1, apt 1
 Schiller Park, IL 60176
 United States

[NEW ARRIVALS](#)
[WOMENS](#)
[KIDS](#)
[MENS](#)
[ELECTRONICS](#)
[TOP SELLERS](#)

 STEP 1: Shipping > STEP 2: Billing > **STEP 3: Place Order**

We're sorry that your order could not be placed. This probably happened due to a high order volume or temporary connectivity errors. Please wait a few minutes and resubmit your order. We won't process your payment until you successfully place your order. If you have further questions, please contact us.

PRODUCT	QTY
<div> Women's Hypertrail Low with Gore-Tex XCR® Membrane Item No.: 883239315158 Color Light Brown Width M Size 4 </div>	1 In Stock

Subtotal
 Shipping Standard
 Sales Tax
Order Total

[Edit Cart](#)
[PLACE ORDER](#)

In Business Manager the order status will be failed in this case.

Orders can have the failed status for many reasons. In order to check that a payment was refused, you should check the notifications tab of order details page. If the credit card payment method was used, the notification should be similar to the one provided below:

[Site](#) > [Orders](#) > [Orders](#) > Order: os04-00003002

[General](#)
[Attributes](#)
[Payment](#)
[Notes](#)
[History](#)

Notes for Order 'os04-00003002'

Fields with a red asterisk (*) are mandatory. Click [Add](#) to save a new note.

Subject*

[Add](#)

Text:

Select All	User	Time	Note
<input type="checkbox"/>	storefront	1/13/16 3:55:22 pm	Adyen Payment Notification AdyenNotification v 48 - Payment info (Called from : 82.199.90.163) ===== reason : 3d-secure: Authentication Failed eventDate : 2015-01-13T15:55:22.002 merchantReference : os04-00003002 currency : USD paymentReference : 8514211634416123 merchantAccountCode : 08P01sma1COM2 eventCode : AUTHORIZATION value : 6045 operations : success : false paymentMethod : visa live : false

There may be a few notifications, you should check the newest one.

Refused payments always have the Eventcode that equals to AUTHORISATION, regardless of the payment method used.

In Adyen back office, you can find details about payments in general, but also specific information about 3-D Secure payments like 3-D Secure Offered (yes or no) and 3-D Secure Authenticated (yes or no).

Refused payments always have the appropriate status:

psp reference ▼	merchant reference	order reference ▲	date ▼	method ▼	status ▼	shopper reference	3d offered	3d authenticated
8614211647545776	os04-00003003		2015-01-13 17:59:22 EET	VISA	Authorised	00000001	Y	Y
8514211634416123	os04-00003002		2015-01-13 17:55:22 EET	VISA	Refused	00000001	Y	N

Use Case: AVS

How to enable/configure AVS on Business Manager

How to enable/disable AVS

1. Go to BM
2. Select a site
3. Go to Site Preferences > Custom Preferences > Adyen
4. Check/uncheck Enable AVS flag

Enable AVS: <input type="checkbox"/>	false
Enable AVS support. Requires Street Name and House Number fields to be sent to Adyen. The result code is returned by the service and should be handled separately	
<div> <input type="button" value="Apply"/> <input type="button" value="Reset"/> </div>	

Other configurations

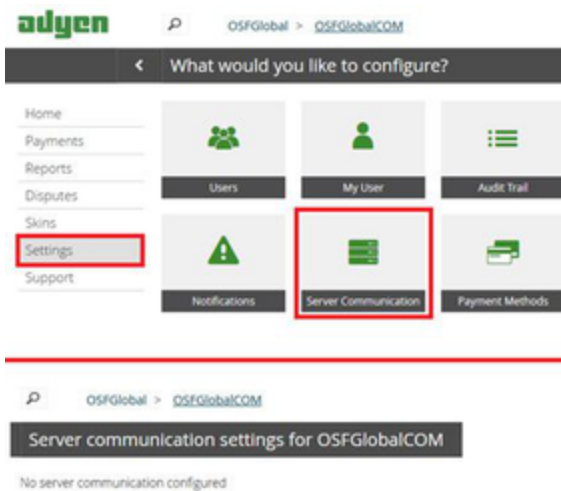
There is a field 'Delimiter for parts of Address1 field' that you can use for configuring the handling of address fields. There is a comment to this field in BM in which the meaning of the field is described, also this field is described in one of the next sections of this document.

Delimiter for parts of Address1 field:	<input type="text"/>
When AVS is enabled, Suite No and Street Name fields are used on the storefront instead of Address1 field. Address1 field is used behind the scenes as well, and it is populated automatically by concatenation of Suite No and Street Name values. The delimiter for parts of Address1 field can be defined here. If empty value is specified, space will be used as a delimiter.	

Adyen Back Office

When billing details are sent to Adyen, the AVS result code is defined, and stored internally in the payment record, without any extra configurations.

Adyen can also send the AVS result code back to SCC site, which may be needed for some site functionality. This possibility is disabled by default. In order to enable AVS, the server communication section has to be used and an appropriate flag has to be set there. This section is disabled by default and in order to enable it, you have to contact Adyen support. Also, when you use server communication, the notifications section has to be turned off, otherwise notifications will be used instead.



General explanation

1. In order to enable/disable AVS on the site, you have to select/deselect the flag in BM
2. In order to have an AVS code returned back from Adyen to the SCC site, you have to configure Adyen back office properly
3. The settings in the Adyen back office mean only that AVS is allowed/disallowed to be used, but they do not require to use it, only the flag in BM turns on/off AVS

AVS use cases

AVS is disabled

Storefront

Storefront works as usually, there are no visible changes for user

Business Manager

BM works as usually, there are no visible changes for merchant

Adyen Back Office

There is a field AVS Result on payment page, where you can check the result code.

There are 2 types of credit cards from an AVS standpoint: cards which support AVS and cards that do not support AVS.

1. If AVS is not supported for credit cards, an appropriate message will be shown in Adyen back office

Card BIN	411111
AVS Result	<i>i</i> AVS not supported for this card type (4)
3D Authenticated	<i>i</i> no

2. If AVS is supported for credit cards, the message will be the following

Card BIN	550000
AVS Result	<i>i</i> No AVS data provided (5)
3D Authenticated	<i>i</i> no

AVS is enabled

Storefront

When AVS is enabled, there should appear two new fields instead of one field called Address1 on all site forms: Suite and Street. These two new fields should appear on all site forms where the user enters his address data. In the current implementation of the Adyen cartridge, only the billing address form from the checkout process is updated to meet this requirement, the other forms are remained untouched

Checkout

STEP 1: Shipping > STEP 2: Billing >

SELECT OR ENTER A BILLING ADDRESS

Choose an Address

• First Name

• Last Name

• Suite

• Street

Address 2

The billing form should work normally with two new fields. The selection of saved addresses from a dropdown list, storing of new addresses and the other functions should normally work on the standard Site Genesis site.

Business Manager

When AVS is enabled, two new fields are used for storing address details: Suite No and Street Name. There are three different places where address data is stored: Customer addresses, order billing and shipping addresses. In the current cartridge implementation, only customer address and order billing address are configured to be used/updated by the cartridge.

1. Customer address

Manage Address "Berlin-5"

Fields with a red asterisk (*) are mandatory

Standard Address

Address ID*

Title:

Company:

Salutation:

First Name:

Second Name:

Last Name*:

Suffix:

Address 1:

Address 2:

Suite No:

Post Box:

City:

Postal Code:

Country*:

State:

Contact Phone:

Street Name:

2. Order billing address

Billing Address for Order 'ad01-00005977'

Fields with a red asterisk (*) are mandatory

Standard Address	
Title:	
Company:	
Salutation:	
First Name:	Denis
Second Name:	
Last Name:	Developer
Suffix:	
Address 1:	10 Downing Street
Address 2:	
Suite No:	10
Post Box:	
City:	Berlin
Postal Code:	12345
Country:	DE (Germany)
State:	OTHER
Contact Phone:	333-333-3333
Street Name:	Downing Street

The logic of the address fields usage and update is like this: the values of Suite and Street fields which the user updates on the storefront are stored in appropriate fields in BM, but they are also concatenated and stored in field Address1. The concatenation of the values is performed behind the scenes, the delimiter for concatenated strings can be changed in Site Preferences, the concatenation of the values is required in order to make the Adyen cartridge compatible with other site functionality and not to interfere with other site functionality.

Adyen Back Office

There is a field AVS Result on the payment page, where you can check the result code.

There are two different types of credit cards from an AVS standpoint: cards for which AVS is supported and for which AVS is not supported.

1. If AVS is not supported for credit card, an appropriate message will be shown in Adyen back office

Card BIN	411111
AVS Result	i AVS not supported for this card type (4)
3D Authenticated	i no

2. If AVS is supported for credit card, one of possible result codes will be shown. Here are a few examples:

- a. Successful address verification result

Card BIN	550000
AVS Result	i Both postal code and address match (7)
3D Authenticated	i no

- b. Not successful address verification result

Card BIN	550000
AVS Result	i Neither postal code nor address match (2)
3D Authenticated	i no

Summary

1. The AVS result code is viewable for the merchant only in Adyen back office.
2. The AVS feature can be enabled in Business Manager.
3. If AVS is enabled, new fields will be shown on the storefront, otherwise there will not be any viewable changes for the user.

4. If AVS is enabled, new fields will be used for storing addresses in BM, but the old field Address1 will be used as well. If AVS is disabled, there will not be any changes visible for the merchant in BM.
5. The AVS result code can be received by the cartridge from the Adyen server, if an appropriate configuration of Adyen back office is done. If the AVS result code is received by the Adyen cartridge, it will not be shown to the merchant in the current implementation of the feature, the code can be handled only programmatically:
 - a. The developer has a possibility to handle AVS result code programmatically if needed
 - b. The merchant can see the AVS result code only in Adyen back office

Limitations, Constraints

- The merchant will need a configured Adyen account
- The merchant can style the Adyen HPP (skin), to make it matching the look and feel of their store. See the Adyen skin manual or contact Adyen support for more info
- Successful payment authorizations lead to a 'Paid' order payment status and 'Ready for Export' export status
- Refused payment authorizations lead to a 'Not Paid' order payment status, and 'Not Exported' export status
- Cancels and refunds lead to a 'Not Paid' order payment status (even if the status was 'Paid' before), and 'Not Exported' export status
- Refused payments have the status 'Refused' in Adyen back office, but 'AUTHORISATION' Eventcode in Business Manager
- In the cartridge these API methods are implemented: 'Capture', 'Cancel Before Capture', and 'Cancel Or Refund'. The statuses require custom integration into the end site, they are optional functions and are not described in this document from a position of integration
- The AVS feature requires that all address forms of the site have two new fields 'Building number' and 'Street name' instead of the old field 'Address 1'. When AVS is enabled, all site forms should have two new fields, otherwise either the new or the old approach can be used. Also, 'Address 1' field has to be updated behind the scenes in order to not break existing site functionality. In this guide it is described how to change Billing address form, it's an example of how should work all the other address forms of your site
- The cartridge makes use of the "[Adyen Hosted](#)" Client Side Encryption library , other methods require additional customization.

Compatibility

Based on Commerce Cloud 17.1.0 and SiteGenesis 103.1.11

Privacy, Payment

- If only the redirect method is used, all payment data is entered into the Adyen HPP by the shopper and no Credit Card data will be stored inside Commerce Cloud (except the brand of used card)
- The shoppers email address and shopper ID used in Commerce Cloud are sent to Adyen to allow 'recurring' and/or 'one click' payments

Implementation Guide

Set Up

Installation

Integration Choice

The Adyen LINK Cartridge can be used with either a pipeline or controller based SiteGenesis. Depending on the scope and needs of the project the cartridge configuration is different.

Pipeline based integration

Install 'int_adyen', 'int_adyen_pipelines' and 'bm_adyen' cartridges using Commerce Cloud UX-studio.

Controller based integration

Install 'int_adyen' and 'bm_adyen', and 'int_adyen_controllers' cartridges using Commerce Cloud UX-studio.

Metadata Import

Find the site_import folder in the installation package, it contains an unzipped site import archive. Please review the archive, do the necessary modifications for your site ID, zip the archive and import it through BM Administration > Site Development > Site Import & Export section. The only required modification is renaming the root folder of the archive to the ID of your site and zipping the folder.

Before the import of the site_import archive you've created, check and save cartridge paths of your site and Business Manager, because they will be modified, so you will be able to configure them manually if something goes wrong. Also, check the fields which will be updated after importing file 'site-template.zip', and be sure that there won't be any conflict with the existing fields.

The following configuration items will be added after you import your site_import archive:

The screenshot displays the 'Adyen Account Settings' page within the Commerce Cloud UX-studio interface. The page is titled 'Adyen Account Settings' and includes a search bar, a 'Cancel' button, and an 'Apply to Other Sites' button. The 'Instance Type' is set to 'Sandbox'. The configuration fields are as follows:

Name	Value	Default Value
Ayden Notification Delay Minutes	1	
Ayden Payment Methods Service Prefix	adyen.http.payment.methods	adyen.http.payment.methods
Ayden Payment Send Service Prefix	adyen.http.payment.send	adyen.http.payment.send
Enable adyen debug mode	Yes	yes / no
HMAC payment setup (SHA-256)	Yes	Enable if you use an autogenerated HMAC SHA-256 key, it is also recommended.
HMACkey	373D8C678CE8895A2C2FC67A2A625967A5E02:	Key used to calculate a signature
Test/Production Mode	TEST	TEST

© 2016 salesforce.com, inc. LyonsSiteGenesis Time Zone: Coordinated Universal Time (Instance Time Zone: Eastern Standard Time) Version: 16.9, last updated Oct 24, 2016 (Compatibility Mo All Rights Reserved

Besides configuration parameters, the following order attributes will be added:

Object Type 'Order' - Attribute Definition Assignments

On this page you can assign existing attribute definitions to your attribute group.

Assign Attribute Definition

ID:





Select All	ID	Name	Type	Attribute Settings	Sorting
<input type="checkbox"/>	Adyen_pspReference	pspReference	String		
<input type="checkbox"/>	Adyen_eventCode	eventCode	String		<input data-bbox="1414 625 1458 657" type="button" value="↑"/>
<input type="checkbox"/>	Adyen_paymentMethod	paymentMethod	String		<input data-bbox="1414 667 1458 699" type="button" value="↓"/>
<input type="checkbox"/>	Adyen_value	Amount	String		

Also the new custom field 'streetName' to 'CustomerAddress' and 'OrderAddress' system objects will be added:

<input type="checkbox"/>		salutation	Salutation	String		0	Edit
<input type="checkbox"/>		secondName	Second Name	String		0	Edit
<input type="checkbox"/>		stateCode	State	String		0	Edit
<input type="checkbox"/>		streetName	Street Name	String		0	Edit
<input type="checkbox"/>		suffix	Suffix	String		0	Edit
<input type="checkbox"/>		suite	Suite No	String		0	Edit

Custom Object Definition:

After importing metadata there will be a site specific custom object to store Adyen notifications:

<input type="checkbox"/>		UUID	UUID	String
<input type="checkbox"/>		creationDate	Creation Date	Date+Time
<input type="checkbox"/>		currency	Currency	String
<input type="checkbox"/>		eventCode	Event Code	String
<input type="checkbox"/>		eventDate	Event Date	String
<input type="checkbox"/>		httpRemoteAddress	Remote Address	String
<input type="checkbox"/>		lastModified	Last Modified	Date+Time
<input type="checkbox"/>		live	Live	String
<input type="checkbox"/>		merchantAccountCode	Merchant Account Code	String
<input type="checkbox"/>		merchantReference	Merchant Reference	String
<input type="checkbox"/>		operations	Operations	String
<input type="checkbox"/>		orderId		String
<input type="checkbox"/>		paymentMethod	Payment Method	String
<input type="checkbox"/>		processedDate	Processed Date	Date+Time
<input type="checkbox"/>		processedStatus	Processed Status	String
<input type="checkbox"/>		pspReference	PSP Reference	String
<input type="checkbox"/>		reason	Reason	String
<input type="checkbox"/>		success	Success	String
<input type="checkbox"/>		updateStatus	Update Status	Enum of Strings
<input type="checkbox"/>		value	Reason	String
<input type="checkbox"/>		version	Version	String

Setup of Scheduled Jobs.

Configurations

Business Manager Configuration

Cartridge Path

If using the controller based integration, cartridge path should be setup as such under **Sites / Manage Sites /**

`app_storefront_controllers:app_storefront_core:int_adyen_controllers:int_adyen`

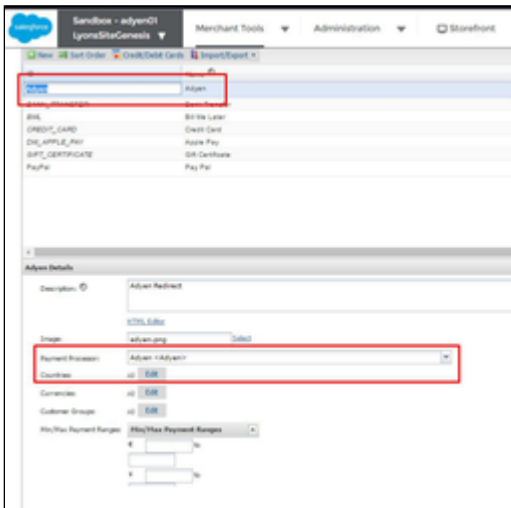
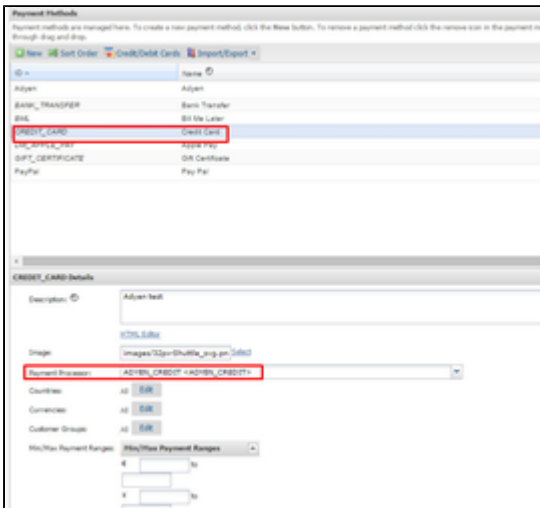
Instance Type:	<input type="text" value="Sandbox/Development"/>
Deprecated. The preferred way of configuring HTTP and HTTPS hostnames is by using new features of the site aliases configuration ("Site URLs/Aliases Configuration"). The HTTP/HTTPS hostnames values set in this section will be used if no ho configuration style.	
HTTP Hostname:	<input type="text"/>
HTTPS Hostname:	<input type="text"/>
Instance Type:	All
Cartridges:	<code>app_storefront_controllers:app_storefront_core:int_adyen_controllers:int_adyen</code>
Effective Cartridge Path:	<code>app_storefront_controllers:app_storefront_core:int_adyen_controllers:int_adyen:plugin_apple_pay:plugin_facebook:plugin_pinterest_commerce:bc_api:core</code>

If using pipelines, the cartridge path should be similar to:


`app_storefront_pipelines:app_storefront_core:int_adyen_pipelines:int_adyen`

Adyen Configurations

In order to use Adyen service for processing credit cards you have to enable 'Credit Card' payment method and set payment processor to 'ADYEN_CREDIT' for it. In order to use 'Adyen redirect' feature, 'Adyen' payment method must be enabled. Both can be enabled in Ordering > Payment Processors:

Adyen HPP Redirect Setup	Adyen CREDIT API Setup
	

Also, there is a page in Business Manager where you have to configure Adyen cartridge, you can find it in Site Preferences > Custom Preferences > Adyen:


Sandbox - adyen01
LyonsSiteGenesis
Merchant Tools
Administration
Storefront

[Merchant Tools](#) / [Site Preferences](#) / [Custom Site Preference Groups](#) /

AdyenNew

Search by IDs...

Instance Type: Sandbox

Name	Value
CSE Enabled	Yes
CSE Javascript Public Key	10001jC845F051414906292D40EAA5FD476B50
CSE Prod Javascript URL	
CSE Test Javascript URL	https://test.adyen.com/hpp/cse/js/75147672384
Adyen Notification Delay Minutes	1
Adyen Payment Methods Service Prefix	adyen.http.payment.methods
Adyen Payment 3D Secure Service Prefix	adyen.http.payment.3dsecure
Adyen Payment Send Service Prefix	adyen.http.payment.send
Enable adyen debug mode	Yes

Here is the description of the Site Preferences fields which have to be configured:

Parameter Name	Description
merchantCode	Adyen merchant account name. The account for which the payments will be processed.
skinCode	Code of the skin which you want to use. There can be defined few skins, each having a different configuration, style, language etc.
Adyen_HMAC256	Secret key used to calculate a signature over the payment fields sent to Adyen. The value must be the same in Site Preferences and in Adyen Account Settings
allowedMethods	Payment Methods that will be shown on the Adyen HPP
Test/Production Mode	Define the live (Production) or test mode
Enable Adyen debug mode	Enables debug mode which shows a 'pay' button before doing the redirect to Adyen. This allows you to check the parameters sent to the HPP
Recurring Type	Either RECURRING, or ONECLICK or RECURRING,ONECLICK
Adyen Notifications User	The username used by the Adyen notification system in order to authenticate the calls with Demandware. Should be the same as the one set in Adyen system
Adyen Notifications Password	The password by the Adyen notification system in order to authenticate the calls with Demandware. Should be the same as the one set in Adyen system
Enable CSE	Enable Client Side Encryption.
Adyen Js Public Key	The public key for the Adyen CSE encryption This should match the Adyen console configuraiton.
Adyen Test JS URL	CSE Test js Url on Adyen
Adyen Prod JS URL	CSE Prod js URL on Adyen
Adyen Notification Delay Minutes	How long to wait before Salesforce will update an order based on notification from Adyen.

Adyen Payment Methods Service Prefix	Prefix for multiple site configurations.
Adyen Payment 3D Secure Service Prefix	Prefix for multiple site configurations.
Adyen Payment Send Service Prefix	Prefix for multiple site configurations.
Enable AVS	Enabled AVS Features.

Services Configuration

Go to **Administration > Operations > Services** and configure Adyen services.

Make sure that you set up prefixes for services like so:

Ayden Payment Methods Service Prefix	adyen.http.payment.methods
Ayden Payment Send Service Prefix	adyen.http.payment.send

Integration will use this prefix + current site ID to create web service. Example:

prefix :adyen.http.payment.methods

site id: SiteGenesis

at the result will be call service with name : adyen.http.payment.methods.SiteGenesis (prefix + '.' + siteID)

Prefix field is required for integration or the Service Registry will error out.

Job Schedule Configuration

Adyen requires a custom job to run periododically, details are in [Scheduled Jobs](#).

Adyen Account Configuration

some items that need to be configured in the Adyen Customer Area (CA) for the Adyen Integration to work correctly. You can find the CA at <https://ca-test.adyen.com>.

Parameter Name	Description
HMAC256 key	Secret key used to calculate a signature over the payment fields sent to Adyen. This should match what is used in the Adyen configuration settings in your DW site
SkinCode	Code of the skin you want to use. You can use different skins, each having a different configuration, style, language etc.
Notification URL and test page	URL where Adyen should send notification messages to. The script at this URL should always reply with [accepted], otherwise Adyen will keep sending messages. The URL will point to your DW store e.g.: https://adyen01.tech-prtnr-eu01.dw.demandware.net/on/demandware.store/Demos-SiteGenesis-Site/default/Adyen-Notify The Notification URL and it's response can be tested from within the CA
Payment Methods	You can select which payment methods should be displayed by configuring your skin. This can also be influenced with the 'allowedMethods' parameter that can be sent to Adyen for each payment
Test HPP and skin	The skin and HPP can be tested from within the CA, to check the look and feel, translations, configured payment methods etc
Settings for the PAL adapter	To send Credit Card payments directly (without redirecting to the HPP) you can use the PAL adapter. The user, password and location can be configured here
Notifications User Name	The username used by the Adyen notification system in order to authenticate the calls with Demandware. Should be the same as the one set in Demandware system under "Adyen Notifications User" preference. (Found under : Settings -> Notifications -> User Name)
Notifications Password	The password used by the Adyen notification system in order to authenticate the calls with Demandware. Should be the same as the one set in Demandware system under "Adyen Notifications Password" preference. . (Found under : Settings -> Notifications -> Password)

Custom Code

Integration Efforts

The following steps are needed to complete the integration:

- Install the cartridges
- Change and import 'site-template.zip' (system objects and custom objects)
- Do the required code changes
- Configure your merchant's Adyen Account in the Adyen back office
- Configure Adyen parameters in Commerce Cloud
- Configure and Enable [Scheduled Jobs](#).
- Test
- Create Skin for Adyen HPP

These steps should not require more than a few hours (creating the Skin can require several hours depending on complexity of styling).

Note that the name of your storefront site ID should match the name in the 'site-template.zip', otherwise you will get import errors.

Also, note that the payment methods 'Adyen' and 'Credit Card' and the payment processors 'Adyen' and 'ADYEN_CREDIT' will be created or updated from the 'site-template.zip'.

Common Changes

The following changes should be made regardless of whether a controller or a pipeline integration approach are being used.

app_storefront_core

Form *billingaddress.xml*

Increase the max length of field address1 and add 2 new fields to the form, here is the expected result:

```
<?xml version="1.0"?>
<form xmlns="http://www.demandware.com/xml/form/2008-04-19">

  <field formid="firstName" label="profile.firstname" type="string"
mandatory="true" binding="firstName" max-length="50"
missing-error="address.firstname.missing"
range-error="address.field.invalid" />
  <field formid="lastName" label="profile.lastname" type="string"
mandatory="true" binding="lastName" max-length="50"
missing-error="address.lastname.missing"
range-error="address.field.invalid" />
  <field formid="address1" label="resource.address1" type="string"
mandatory="true" binding="address1" max-length="105"
missing-error="resource.addresserror"
range-error="address.field.invalid" />
  <field formid="address2" label="resource.address2" type="string"
mandatory="false" binding="address2" max-length="50"
range-error="address.field.invalid" />
  <field formid="suite" label="forms.suite" type="string" mandatory="true"
binding="suite" max-length="50" missing-error="forms.suiteerror"
range-error="forms.suite.field.invalid" />
  <field formid="streetName" label="forms.streetname" type="string"
mandatory="true" binding="custom.streetname" max-length="50"
missing-error="forms.streetnameerror"
range-error="forms.streetname.field.invalid" />
  <field formid="city" label="resource.city" type="string"
mandatory="true" binding="city" min-length="2" max-length="50"
missing-error="address.city.missing" range-error="address.field.invalid"
/>

  <!-- postal code -->
  <field formid="postal" label="resource.zipcode" type="string"
mandatory="true" min-length="5" max-length="10"
regexp="(^\d{5}(-\d{4})?$)|(^[abceghjklmnprstvxYABCEGHJKLMNPRSTVXY]{1}\d{1}[A-Za-z]{1} *\d{1}[A-Za-z]{1}\d{1}$)" binding="postalCode"
missing-error="resource.errorzip"/>

  <!-- use set of supported billing countries -->
  <field formid="country" label="resource.country" type="string"
default="us" mandatory="true" binding="countryCode"
```



```
missing-error="address.country.missing"
value-error="address.field.invalid">
  <options>
    <option optionid="us" label="country.unitedstates" value="us"/>
  </options>
</field>

<!-- use global state list -->
<include formid="states" name="states"/>

<!-- phone is optional for billing addresses -->
<field formid="phone" label="profile.phone"
description="address.phone.example" type="string" mandatory="true"
binding="phone" max-length="20" missing-error="address.phone.missing"
range-error="address.field.invalid" />

<!-- actions provided by the edit billing address dialog -->
<action formid="apply" valid-form="true"/>
```

```
<action formid="remove" valid-form="false"/>
</form>
```

To make use of some of Adyen's various payment providers, you may also likely want to add additional billing countries in this file.

Form creditcard.xml

Add fields for client-side card encryption and recurring payment options.

Here is the expected result with new lines at 56 and 64:

```
<!-- field for credit card owner -->
  <field formid="owner" label="creditcard.ownerlabel" type="string"
mandatory="true" max-length="40" binding="creditCardHolder"
    missing-error="creditcard.ownermissingerror"/>
  <field formid="encrypteddata" type="string" mandatory="false" />
  <!-- field for credit card security code -->
    <field formid="cvn" label="creditcard.cvnlabel" type="string"
mandatory="true" masked="0"
    missing-error="creditcard.cvnmissingerror"
value-error="creditcard.cvnrangeerror"/>

  <!-- optional flags -->
  <field formid="saveCard" label="creditcard.savecard" type="boolean"
mandatory="false" default-value="true" />
  <!-- field for credit card recurring payments / tokenization -->
  <field formid="selectedCardID" mandatory="false" type="string" />

  <!-- confirm action to save the card details -->
    <action formid="confirm" valid-form="true"/>

</form>
```

Client Javascript adyen_cse.js

The adyen-cse.js file included in the int_adyen cartridge needs to be copied into your STOREFRONT app_storefront_core/js folder for inclusion into the storefront javascript and gulp builder tools.

cartridge/js/adyen-cse.js

Client Javascript app.js

Add the supplied adyen-cse.js file to your app_storefront_core/cartridge/js folder, and then require in the file in your app.js file in the same folder.

Here is the expected result with new line on 21:

```
'use strict';

var countries = require('./countries'),
    dialog = require('./dialog'),
    minicart = require('./minicart'),
    page = require('./page'),
    rating = require('./rating'),
    searchplaceholder = require('./searchplaceholder'),
    searchsuggest = require('./searchsuggest'),
    tooltip = require('./tooltip'),
    util = require('./util'),
    validator = require('./validator'),
    tls = require('./tls'),
    adyenCse = require('.adyen-cse');
```

Client Javascript pages/account.js

Require in the cse module first:

```
'use strict';

var giftcert = require('../giftcert'),
    tooltip = require('../tooltip'),
    util = require('../util'),
    dialog = require('../dialog'),
    page = require('../page'),
    login = require('../login'),
    validator = require('../validator'),
    adyenCse = require('.adyen-cse');
```

Add code to initialize Adyen's custom client side code on payment forms.

Here is the expected result with new lines in red:

```

function initializePaymentForm() {
    $('#CreditCardForm').on('click', '.cancel-button', function (e) {
        e.preventDefault();
        dialog.close();
    });
    if (SitePreferences.ADYEN_CSE_ENABLED {
        adyenCse.initAccount();
    }
}
/**
 * @private

```

Client Javascript pages/checkout/billing.js

Several code changes to support various different Adyen payment options.

Here are the expected results with new lines at line 9 and 25:

First require in the cse module

```

'use strict';

var ajax = require('../..//ajax'),
    formPrepare = require('../formPrepare'),
    giftcard = require('../..//giftcard'),
    util = require('../..//util'),
    adyenCSE = require('../..//adyen-cse');

/**
 * @function

```

```

/**
 * @function
 * @description Fills the Credit Card form with the passed
data-parameter and clears the former cvn input
 * @param {Object} data The Credit Card data (holder, type, masked
number, expiration month/year)
 */
function setCCFields(data) {
    var $creditCard = $(' [data-method="CREDIT_CARD"] ');

    $creditCard.find('input[name$="creditCard_owner"]').val(data.holder).trigger('change');

    $creditCard.find('select[name$="_type"]').val(data.type).trigger('change');

    $creditCard.find('input[name$="_creditCard_number"]').val(data.maskedNumber).trigger('change');

    $creditCard.find('input[name$="_month"]').val(data.expirationMonth).trigger('change');

    $creditCard.find('input[name$="_year"]').val(data.expirationYear).trigger('change');

    $creditCard.find('input[name$="_cvn"]').val('').trigger('change');
    $creditCard.find('input[name$="creditCard_selectedCardID"]').val(data.selectedCardID).trigger('change');
}

```

change (two new functions added)

```

// ensure checkbox of payment method is checked
    $('input[name$="_selectedPaymentMethodID"]').removeAttr('checked');
    $('input[value=' + paymentMethodID + ']').prop('checked',
'checked');

    formPrepare.validateForm();
}

/**
 * @function
 * @description Changes the payment type or issuerId of the selected
payment method
 * @param {String, Boolean} value of payment type or issuerId and a test
value to see which one it is. to which the payment type or issuerID
should be changed to
 */
function upgradePaymentType(selectedPayType, test) {
    if (!test) {
        $('input[name="brandCode"]').removeAttr('checked');
    } else {
        $('input[name="issuerId"]').removeAttr('checked');
    }
    $('input[value=' + selectedPayType + ']').prop('checked', 'checked');
    formPrepare.validateForm();
}

/**
 * @function
 * @description Adyen - Initializes the visibility of HPP fields
 */
function initializeHPPFields () {
    if ($('[name="brandCode"]:checked').hasClass('openInvoice')) {
        $('.additionalfield').hide().find('input').val('');
        $('.additionalfield.' +
$('.checkout-billing').find('select.country').val()).show();
    } else {
        $('.additionalfield').hide().find('input').val('');
    }
}

```

changes on line 110 and line 126 (some additions to the billing init)

```

/**
 * @function
 * @description loads billing address, Gift Certificates, Coupon and
Payment methods
 */
exports.init = function () {
    var $checkoutForm = $('#.checkout-billing');
    var $addGiftCert = $('#add-giftcert');
    var $giftCertCode = $('input[name$="_giftCertCode"]');
    var $addCoupon = $('#add-coupon');
    var $couponCode = $('input[name$="_couponCode"]');
    var $selectPaymentMethod = $('.payment-method-options');
    var selectedPaymentMethod =
$selectPaymentMethod.find(':checked').val();
    var $payType = $('[name="brandCode"]');
    var $issuerId = $('[name="issuerId"]');
    var $issuer = $('ul#issuer');
    var selectedPayType = $payType.find(':checked').val();
    var selectedIssuerId = $issuerId.find(':checked').val();

    formPrepare.init({
        formSelector: 'form[id$="billing"]',
        continueSelector: '[name$="billing_save"]'
    });

    // default payment method to 'CREDIT_CARD'
    updatePaymentMethod((selectedPaymentMethod) ? selectedPaymentMethod
: 'CREDIT_CARD');
    $selectPaymentMethod.on('click', 'input[type="radio"]', function ()
{
        updatePaymentMethod($(this).val());
        if ($(this).val() == 'Adyen' && $payType.length > 0) {
            //set payment type of Adyen to the first one
            updatePaymentType((selectedPayType) ? selectedPayType :
$payType[0].value, false);
        } else {
            $payType.removeAttr('checked');
            $issuerId.removeAttr('checked');
        }
    });
    $issuerId.on('click', function () {
        updatePaymentType($(this).val(), true);
    })

    // select credit card from list
    $('#creditCardList').on('change', function () {
        var cardUUID = $(this).val();
        if (!cardUUID) {return;}
        populateCreditCardForm(cardUUID);
    });
}

```

change some more additions at the end of the main exports.init function

```
$giftCertCode.on('keydown', function (e) {
    if (e.which === 13) {
        e.preventDefault();
        $addGiftCert.click();
    }
});

if (SitePreferences.ADYEN_CSE_ENABLED) {
    adyenCse.initBilling();
}

// Adyen - Click event for payment methods
$payType.on('click', function () {
    updatePaymentType($(this).val(), false);
    //if the payment type contains issuerId fields, expand form with
the values
    if ($(this).siblings('#issuer').length > 0) {
        $issuer.show();
        updatePaymentType(
(selectedIssuerId) ? selectedIssuerId : $issuerId[0].value, true);
    } else {
        $issuer.hide();
        $('input[name="issuerId"]').removeAttr('checked');
    }
    initializeHPPFields();
});

var currentDate = new Date();
var currentYear = currentDate.getFullYear();
var initYear = currentYear - 100;
$('.openinvoiceInput input[name$="_dob"]').datepicker({
    showOn: 'focus',
    yearRange: initYear + ':' + currentYear,
    changeYear: true
});

};
```

Script cart/CartUtils.ds

Find method CartUtils.getAddressList(), find function getAddressObject() inside of it, and add these 2 new fields to the returned object:

suite: addy.suite, // Added for Adyen AVS support

streetName: addy.custom.streetName, // Added for Adyen AVS support

Here is the expected result with additions on line 27/28:

```
CartUtils.getAddressList = function(basket, currentCustomer,
includeLineItems) {
  function getAddressObject(addy, plist) {
    var display = addy.ID;
    var key = addy.ID;
    var addyType = "customer";
    if (!empty(plist)) {
      addyType = plist.type;
      var rsc = plist.type==ProductList.TYPE_WISH_LIST ? "wishlist" :
"giftregistry";
      display = dw.web.Resource.msgf("singleshipping."+rsc, 'checkout',
null, plist.owner.profile.firstName);
      key = plist.owner.profile.credentials.login+"??"+addy.ID;
    }

    return {
      UUID : addy.UUID,
      ID: addy.ID,
      key: key,
      firstName: addy.firstName,
      lastName: addy.lastName,
      address1: addy.address1,
      suite: addy.suite, // Added for Adyen AVS support
      streetName: addy.custom.streetName, // Added for Adyen AVS support
      address2: addy.address2,
      postalCode: addy.postalCode,
    }
  }
}
```

Script checkout/AddAddressToAddressBook.ds

Replace the line of code:

```
address.setAddress1( usedAddress.address1 );
```

on the following code:

```
// Added for Adyen AVS support. If AVS is enabled, 2 new fields are used on the site
```

```
if (dw.system.Site.getCurrent().getCustomPreferenceValue("Adyen_enableAVS")) {
  address.setSuite( usedAddress.suite );
  address.custom.streetName = usedAddress.custom.streetName;

  address.setAddress1( usedAddress.suite +
(!empty(dw.system.Site.getCurrent().getCustomPreferenceValue("Adyen_address1Delimiter")) ?
dw.system.Site.getCurrent().getCustomPreferenceValue("Adyen_address1Delimiter") : " ") + usedAddress.custom.streetName );
} else {
  address.setAddress1( usedAddress.address1 );
}
```

The result should look like:

```

}

address = addressBook.createAddress( addressID );
address.setFirstName( usedAddress.firstName );
address.setLastName( usedAddress.lastName );
// Added for Adyen AVS support. If AVS is enabled, 2 new fields are
used on the site
if
(dw.system.Site.getCurrent().getCustomPreferenceValue("Adyen_enableAVS")
) {
    address.setSuite( usedAddress.suite );
    address.custom.streetName = usedAddress.custom.streetName;
    address.setAddress1( usedAddress.suite +
(!empty(dw.system.Site.getCurrent().getCustomPreferenceValue("Adyen_addr
ess1Delimiter")) ?
dw.system.Site.getCurrent().getCustomPreferenceValue("Adyen_address1Deli
miter") : " ") + usedAddress.custom.streetName );
    } else {
        address.setAddress1( usedAddress.address1 );
    }

    address.setAddress2( usedAddress.address2 );
    address.setCity( usedAddress.city );
    address.setPostalCode( usedAddress.postalCode );
    address.setStateCode( usedAddress.stateCode );

```

Script checkout/UpdateBillingAddress.ds

Replace the line of code:

```
billingAddress.setAddress1( addressFields.address1.value );
```

on the following code:

```

// Added for Adyen AVS support. If AVS is enabled, 2 new fields are used on the site
if (dw.system.Site.getCurrent().getCustomPreferenceValue("Adyen_enableAVS")) {
    billingAddress.setSuite( addressFields.suite.value );
    billingAddress.custom.streetName = addressFields.streetName.value;
    billingAddress.setAddress1( addressFields.suite.value +
(!empty(dw.system.Site.getCurrent().getCustomPreferenceValue("Adyen_address1Delimiter")) ?
dw.system.Site.getCurrent().getCustomPreferenceValue("Adyen_address1Delimiter") : " ") + addressFields.streetName.value );
} else {
    billingAddress.setAddress1( addressFields.address1.value );
}

```

Resulting in:

```

function execute( pdict : PipelineDictionary ) : Number
{
    var billingAddress : OrderAddress = pdict.BillingAddress;
    var addressFields : FormGroup = pdict.AddressForm.addressFields;

    // copy the address details
    billingAddress.setFirstName( addressFields.firstName.value );
    billingAddress.setLastName( addressFields.lastName.value );
    // Added for Adyen AVS support. If AVS is enabled, 2 new fields are
    used on the site
    if
(dw.system.Site.getCurrent().getCustomPreferenceValue("Adyen_enableAVS")
) {
        billingAddress.setSuite( addressFields.suite.value );
        billingAddress.custom.streetName = addressFields.streetName.value;
        billingAddress.setAddress1( addressFields.suite.value +
(!empty(dw.system.Site.getCurrent().getCustomPreferenceValue("Adyen_addr
ess1Delimiter")) ?
dw.system.Site.getCurrent().getCustomPreferenceValue("Adyen_address1Deli
miter") : " ") + addressFields.streetName.value );
    } else {
        billingAddress.setAddress1( addressFields.address1.value );
    }
    billingAddress.setAddress2( addressFields.address2.value );
    billingAddress.setCity( addressFields.city.value );
    billingAddress.setPostalCode( addressFields.postal.value );
    billingAddress.setStateCode( addressFields.states.state.value );
    billingAddress.setCountryCode( addressFields.country.value );
    billingAddress.setPhone( addressFields.phone.value );
}

```

Script util/Resrouce.ds

Require in the AdyenHelper and add a few lines.

Here is the expected result with new lines at 10/11, 127-129, 206/207 222/223:

```

/**
 * Resource helper
 *
 */
var Currency = require('dw/util/Currency');
var Site = require('dw/system/Site');
var ContentMgr = require('dw/content/ContentMgr');
var ProductAvailabilityModel =
require('dw/catalog/ProductAvailabilityModel');

/* Script Modules */
var AdyenHelper = require
("int_adyen/cartridge/scripts/util/AdyenHelper");

function ResourceHelper() {}

```

```

VALIDATE_CREDITCARD          : Resource.msg('validate.creditcard',
'forms', null),
    VALIDATE_EQUALTO          :
Resource.msg('validate.equalTo', 'forms', null),
    VALIDATE_MAXLENGTH        :
Resource.msg('validate.maxlength', 'forms', null),
    VALIDATE_MINLENGTH        :
Resource.msg('validate.minlength', 'forms', null),
    VALIDATE_RANGELENGTH      :
Resource.msg('validate.rangelength', 'forms', null),
    VALIDATE_RANGE            :
Resource.msg('validate.range', 'forms', null),
    VALIDATE_MAX              : Resource.msg('validate.max',
'forms', null),
    VALIDATE_MIN              : Resource.msg('validate.min',
'forms', null),

    ADYEN_CC_VALIDATE         : Resource.msg('adyen.creditcard',
'adyen', null)
    };

// additional resources
resources[ProductAvailabilityModel.AVAILABILITY_STATUS_IN_STOCK] =
Resource.msg('global.instock', 'locale', null);

```

```

setSessionCurrency          :
URLUtils.url('Currency-SetSessionCurrency').toString(),
    addEditAddress          :
URLUtils.url('COShippingMultiple-AddEditAddressJSON').toString(),
    cookieHint               : URLUtils.url('Page-Show', 'cid',
'cookie_hint').toString(),
    rateLimiterReset        :
URLUtils.url('RateLimiter-HideCaptcha').toString(),
    csrffailed               :
URLUtils.url('CSRF-Failed').toString(),
    adyenpaymentmethods      :
URLUtils.url('Adyen-GetPaymentMethodsJSON').toString()
    };
    return urls;
}
/**
 * Get the client-side preferences of a given page
 * @returns {Object} An objects key key-value pairs holding the
preferences
 */
ResourceHelper.getPreferences = function(pageContext) {
    var cookieHintAsset = ContentMgr.getContent('cookie_hint');
    return {
        LISTING_INFINITE_SCROLL:
(Site.getCurrent().getCustomPreferenceValue('enableInfiniteScroll') ?
true : false),
        LISTING_REFINE_SORT: true,
        STORE_PICKUP:
Site.getCurrent().getCustomPreferenceValue('enableStorePickUp'),
        COOKIE_HINT: (cookieHintAsset && cookieHintAsset.online) ||
false,
        CHECK_TLS:
Site.getCurrent().getCustomPreferenceValue('checkTLS'),
        ADYEN_CSE_ENABLED : AdyenHelper.getAdyenCseEnabled()
    };
}
/**
 * Get the client-side preferences of a given page
 * @returns {Object} An objects key key-value pairs holding the
preferences
 */
ResourceHelper.getSessionAttributes = function(pageContext) {

```

Scss styling update

To support the date picker required for the DOB for some payment options, you may need to add some code in the `_checkout.scss` file. Here is a sample

```

.ui-datepicker {
  background-color: #e3e3e3;
  .ui-icon {
    text-indent: 0;
  }
}

.checkout-progress-indicator {
  overflow: hidden;
  div {
    display: inline;
    padding: 0 .83em 0 0;
  }
}

```

Template account/payment/paymentinstrumentdetails.isml

Changes to support saved credit cards.

Here is the expected result with new lines on line 27-32:

```

<form id="CreditCardForm" name="CreditCardForm"
action="${URLUtils.httpsContinue()}" class="form-horizontal"
method="post" id="newcreditcard">
  <fieldset>
    <isscript>
      var ownerAttributes = {
        maxlength: 40,
        size: 40
      };
      var numberAttributes = {
        maxlength: 16,
        size: 17
      };
    </isscript>

    <isset name="AdyenHelper"
value="${require('int_adyen/cartridge/scripts/util/AdyenHelper')}"
scope="pdict" />
    <isset name="AdyenCseEnabled"
value="${pdict.AdyenHelper.getAdyenCseEnabled()}" scope="page" />
    <isif condition="${AdyenCseEnabled}">
      <isinclude
template="account/payment/adyenpaymentinstrumentdetails"/>
    <elseif/>

    <isinputfield
formfield="${pdict.CurrentForms.paymentinstruments.creditcards.newcredit

```

```

card.owner}" type="input" attributes="${ownerAttributes}" />
    <isinputfield
formfield="${pdict.CurrentForms.paymentinstruments.creditcards.newcredit
card.type}" type="select"/>
    <isinputfield
formfield="${pdict.CurrentForms.paymentinstruments.creditcards.newcredit
card.number}" dynamicname="true" type="input"
attributes="${numberAttributes}"/>
    <div class="form-label-text">
        <span class="required-indicator">• </span>

${Resource.msg('account.paymentinstrumentdetails.expires','account',null
)}
    </div>
    <isscript>
        var currentCountry =
require('~cartridge/scripts/util/Countries').getCurrent(pdict);
    </isscript>

    <isdynamicform
formobject="${pdict.CurrentForms.paymentinstruments.creditcards.newcredi
tcard.expiration}"
formdata="${currentCountry.dynamicForms.expirationInfo}" />

    <div class="form-row form-row-button">
        <button id="applyBtn" type="submit"
name="${pdict.CurrentForms.paymentinstruments.creditcards.create.htmlNam
e}" value="${Resource.msg('global.apply','locale',null)}">
            ${Resource.msg('global.apply','locale',null)}
        </button>
        <button class="cancel cancel-button simple" type="submit"
name="${pdict.CurrentForms.paymentinstruments.creditcards.cancel.htmlNam
e}" value="${Resource.msg('global.cancel','locale',null)}">
            ${Resource.msg('global.cancel','locale',null)}
        </button>
    </isif>
</div>
<input type="hidden" name="${dw.web.CSRFProtection.getTokenName()}"

```

```

value="{dw.web.CSRFProtection.generateToken()}" />
</fieldset>
</form>

```

Template checkout/billing/billing.isml

More changes to support the various Adyen payment methods.

Here is the expected result:

```

<iscomment>payment method area</iscomment>
    <isinclude template="checkout/billing/paymentmethods"/>
    <isbonusdiscountlineitem
p_alert_text="{Resource.msg('billing.bonusproductalert','checkout',null
)}" p_discount_line_item="{pdict.BonusDiscountLineItem}" />

        <div class="form-row form-row-button">
            <isif condition="{(pdict.AdyenHelper &&
pdict.AdyenHelper.getAdyenCseEnabled()) || pdict.AdyenCseEnabled ==
true}">
                <button class="button-fancy-large" type="hidden"
id="billing-submit-hidden" style="display:none"
name="{pdict.CurrentForms.billing.save.htmlName}"
value="{Resource.msg('global.continueplaceorder','locale',null)}"><span
>{Resource.msg('global.continueplaceorder','locale',null)}</span></butt
on>
                <button class="button-fancy-large" type="submit" id="billing-submit"
name="{pdict.CurrentForms.billing.save.htmlName}"
value="{Resource.msg('global.continueplaceorder','locale',null)}"><span
>{Resource.msg('global.continueplaceorder','locale',null)}</span></butt
on>
            <iselse/>
                <button class="button-fancy-large" type="submit"
name="{pdict.CurrentForms.billing.save.htmlName}"
value="{Resource.msg('global.continueplaceorder','locale',null)}"><span
>{Resource.msg('global.continueplaceorder','locale',null)}</span></butt
on>
            </isif>
        </div>
        <input type="hidden"
name="{dw.web.CSRFProtection.getTokenName()}"
value="{dw.web.CSRFProtection.generateToken()}" />
    </form>

```

Template checkout/billing/creditcardjson.isml

Add one line as shown below at line 13:

```
<iscontent type="application/json" charset="UTF-8" compact="true"/>
<isinclude template="util/jsonmodule"/>
<iscomment>
  This template renders the attributes of a customer credit card payment
  instrument as JSON response.
</iscomment>
<isscript>
  var cc = {
    maskedNumber:pdict.SelectedCreditCard.maskedCreditCardNumber,
    holder:pdict.SelectedCreditCard.creditCardHolder,
    type:pdict.SelectedCreditCard.creditCardType,
    expirationMonth:pdict.SelectedCreditCard.creditCardExpirationMonth,
    expirationYear:pdict.SelectedCreditCard.creditCardExpirationYear ,
    selectedCardID:pdict.SelectedCreditCard.UUID
  }
  var json = JSON.stringify(cc);
</isscript>
<isprint value="{json}" encoding="off"/>
```

Template checkout/billing/paymentmethods.isml

Around line 42, in the if condition checking for CREDIT_CARD add the following:

```
<iscomment>
  Credit card block
  -----
</iscomment>

  <div class="payment-method <isif
condition="{empty(pdict.selectedPaymentID) ||
pdict.selectedPaymentID=='CREDIT_CARD'}">payment-method-expanded</isif>"
data-method="CREDIT_CARD">
    <isset name="AdyenHelper"
value="{require('int_adyen/cartridge/scripts/util/AdyenHelper')}"
scope="pdict"/>
    <isset name="AdyenCseEnabled"
value="{pdict.AdyenHelper.getAdyenCseEnabled()}" scope="page" />
    <iscomment>display select box with stored credit cards if customer is
authenticated</iscomment>
    <isif condition="{pdict.CurrentCustomer.authenticated &&
!empty(pdict.ApplicableCreditCards)}">

      <div class="form-row">
        <label
class="label">${Resource.msg('billing.selectcreditcard','checkout',null)}
```

```

} </label>
    <div class="field-wrapper">

        <isif condition="{${AdyenCseEnabled}}">
            <select
name="{${pdict.CurrentForms.billing.paymentMethods.creditCardList.htmlName}" id="adyenCreditCardList" class="input-select">
                <iselse/>
                <select
name="{${pdict.CurrentForms.billing.paymentMethods.creditCardList.htmlName}" id="creditCardList" class="input-select">
                    </isif>
                    <option value=""
selected="selected">${Resource.msg('billing.creditcardlistselect','checkout',null)}</option>
                    <isloop items="{${pdict.ApplicableCreditCards}}"
var="creditCardInstr">
                        <option value="{${creditCardInstr.UUID}}">(<isprint
value="{${creditCardInstr.creditCardType}}"/>) <isprint
value="{${creditCardInstr.maskedCreditCardNumber}}"/> -
${Resource.msg('billing.creditcardlistexp','checkout',null)} <isprint
value="{${creditCardInstr.creditCardExpirationMonth}" formatter="00"
/>.<isprint value="{${creditCardInstr.creditCardExpirationYear}"
formatter="0000" /></option>
                    </isloop>
                </select>
            </div>
        </div>

        <div class="form-row form-row-button">
            <button id="credit-card-select-go"
name="{${pdict.CurrentForms.billing.creditCardSelect.htmlName}"
type="submit" value="Go" class="simple-submit">Select</button>
        </div>

        <iscomment>
            <isloop items="{${pdict.ApplicableCreditCards}}"
var="creditCardInstr">
                <a href="{${URLUtils.https('COBilling-UpdateCreditCardSelection',
'creditCardUUID', creditCardInstr.UUID)}"
                    (<isprint value="{${creditCardInstr.creditCardType}}"/>)
                    <isprint value="{${creditCardInstr.maskedCreditCardNumber}}"/>
                    - ${Resource.msg('billing.creditcardlistexp','checkout',null)}
                    <isprint value="{${creditCardInstr.creditCardExpirationMonth}"
formatter="00" />
                    .<isprint value="{${creditCardInstr.creditCardExpirationYear}"
formatter="0000" />
                </a>
            </isloop>
        </iscomment>

```

```

</isif>

<isif condition="${AdyenCseEnabled}">
  <isinclude template="checkout/billing/adyenpaymentmethods"/>
  <isinputfield
formfield="${pdict.CurrentForms.billing.paymentMethods.creditCard.selectedCardID}" type="hidden"/>
  <iselse/>

  <isinputfield
formfield="${pdict.CurrentForms.billing.paymentMethods.creditCard.owner}"
" type="input"/>

  <isinputfield
formfield="${pdict.CurrentForms.billing.paymentMethods.creditCard.type}"
type="select"/>

  <isinputfield
formfield="${pdict.CurrentForms.billing.paymentMethods.creditCard.number}"
" type="input" dynamicname="true"/>

  <div class="form-row required">
    <label>
      <span
class="required-indicator">${Resource.msg('billing.requiredindicator','checkout',null)}</span>
      <span>${Resource.msg('billing.creditcardlistexpdate','checkout',null)}</span>
    </label>
    <isscript>
      var currentCountry =
require('~/cartridge/scripts/util/Countries').getCurrent(pdict);
    </isscript>

    <isdynamicform
formobject="${pdict.CurrentForms.billing.paymentMethods.creditCard.expiration}"
formdata="${currentCountry.dynamicForms.expirationInfo}"/>

  </div>

  <isscript>
    var help = {
      label: Resource.msg('billing.linkcvn','checkout',null),
      cid: 'checkout-security-code'
    };
  </isscript>
  <isinputfield
formfield="${pdict.CurrentForms.billing.paymentMethods.creditCard.cvn}"
type="input" rowclass="cvn" dynamicname="true" help="${help}"/>

```



```
</isif>
```

```
<isif condition="${pdict.CurrentCustomer.authenticated}">
```

Don't forget to close 'if' statement after you move all other default SG code that was moved to 'else' statement.

Add the following code right before the closing </fieldset> tag, near line 147::

```
<div class="payment-method <isif condition="${!empty(pdikt.selectedPaymentID) &&  
pdikt.selectedPaymentID=='Adyen'}">payment-method-expanded</isif>" data-method="Adyen">
```

```
<isinclude template="hpp"/>
```

```
</div>
```

```
<iscomment>
```

```
Custom processor
```

```
-----
```

```
</iscomment>
```

```
<div class="payment-method <isif  
condition="${!empty(pdikt.selectedPaymentID) &&  
pdikt.selectedPaymentID=='PayPal'}">payment-method-expanded</isif>"  
data-method="Custom">
```

```
<!-- Your custom payment method implementation goes here. -->  
${Resource.msg('billing.custompaymentmethod','checkout',null)}
```

```
</div>
```

```
<div class="payment-method <isif  
condition="${!empty(pdikt.selectedPaymentID) &&  
pdikt.selectedPaymentID=='Adyen'}">payment-method-expanded</isif>"  
data-method="Adyen">
```

```
<isinclude template="hpp"/>
```

```
</div>
```

```
</fieldset>
```

```
<iselse/>
```

```
<div class="gift-cert-used form-indent">
```

Template checkout/summary/summary.isml

```

Add the following code right before the closing </fieldset> tag:
<iscomment>
Set the brandcode and issuerId in session for when user hits the back
button on Adyen hpp
</iscomment>
<isif condition="${!empty(pdikt.CurrentHttpParameterMap.brandCode.value)
|| !empty(session.custom.brandCode)}">
<isset name="brandCode"
value="${!empty(pdikt.CurrentHttpParameterMap.brandCode.value) ?
pdikt.CurrentHttpParameterMap.brandCode.value :
session.custom.brandCode}" scope="session"/>
<input type="hidden" name="brandCode"
value="${session.custom.brandCode}" />
<isif condition="${!empty(pdikt.CurrentHttpParameterMap.issuerId.value)
|| !empty(session.custom.issuerId)}">
<isset name="issuerId"
value="${!empty(pdikt.CurrentHttpParameterMap.issuerId.value) ?
pdikt.CurrentHttpParameterMap.issuerId.value : session.custom.issuerId}"
scope="session"/>
<input type="hidden" name="issuerId" value="${session.custom.issuerId}"
/>
</isif>
</isif>

```

```

<form action="${URLUtils.https('COSummary-Submit')}}" method="post"
class="submit-order">
  <fieldset>
    <div class="form-row">
      <a class="back-to-cart" href="${URLUtils.url('Cart-Show')}}">
        <isprint
value="${Resource.msg('summary.editcart','checkout',null)}"
encoding="off" />
      </a>
      <button class="button-fancy-large" type="submit" name="submit"
value="${Resource.msg('global.submitorder','locale',null)}">
        ${Resource.msg('global.submitorder','locale',null)}
      </button>
    </div>
    <input type="hidden"
name="${dw.web.CSRFPProtection.getTokenName()}"
value="${dw.web.CSRFPProtection.generateToken()}" />

    <iscomment>
      Set the brandcode and issuerId in session for when user hits the
      back button on Adyen hpp
    </iscomment>
  </fieldset>

```

```

        </iscomment>
        <isif
condition="${!empty(pdikt.CurrentHttpParameterMap.brandCode.value) ||
!empty(session.custom.brandCode)}">
            <isset name="brandCode"
value="${!empty(pdikt.CurrentHttpParameterMap.brandCode.value) ?
pdikt.CurrentHttpParameterMap.brandCode.value :
session.custom.brandCode}" scope="page"/>
                <input type="hidden" name="brandCode" value="${brandCode}" />
            <isif
condition="${!empty(pdikt.CurrentHttpParameterMap.issuerId.value) ||
!empty(session.custom.issuerId)}">
                <isset name="issuerId"
value="${!empty(pdikt.CurrentHttpParameterMap.issuerId.value) ?
pdikt.CurrentHttpParameterMap.issuerId.value : session.custom.issuerId}"
scope="session"/>
                    <input type="hidden" name="issuerId"
value="${session.custom.issuerId}" />
                </isif>
            <isset name="brandCode"
value="${!empty(pdikt.CurrentHttpParameterMap.brandCode.value) ?
pdikt.CurrentHttpParameterMap.brandCode.value :
session.custom.brandCode}" scope="session"/>
                <isset name="dob"
value="${!empty(pdikt.CurrentHttpParameterMap.dob.value) ?
pdikt.CurrentHttpParameterMap.dob.value : ''}" scope="session"/>
                    <isset name="gender"
value="${!empty(pdikt.CurrentHttpParameterMap.gender.value) ?
pdikt.CurrentHttpParameterMap.dob.value : ''}" scope="session"/>
                </isif>

        </fieldset>
    </form>

```

```
</div>

</isdecorate>
```

Template components/header/htmlhead.isml

Add the following code after including file style.css:

```
<link rel="stylesheet" href="{URLUtils.staticURL('/css/checkout.css')}" />
```

```
<isinclude template="components/header/htmlhead_UI" />

<!-- UI -->
<link rel="stylesheet" href="{URLUtils.staticURL('/css/style.css')}" />
<link rel="stylesheet" href="{URLUtils.staticURL('/css/checkout.css')}" />

<!--[if lte IE 8]>
<script
src="//cdnjs.cloudflare.com/ajax/libs/respond.js/1.4.2/respond.js"
type="text/javascript"></script>
<script
src="https://cdn.rawgit.com/chuckcarpenter/REM-unit-polyfill/master/js/rem.min.js" type="text/javascript"></script>
<![endif]-->
```

File resources/forms.properties

Add the following lines to the file:

forms.suite=Suite

forms.streetName=Street

forms.suiteerror=Please enter Suite number

forms.streetnameerror=Please enter Street name

forms.suite.field.invalid=Please enter a valid value

forms.streetname.field.invalid=Please enter a valid value

adyentest.missedordernumber=Missed order number

adyentest.wrongordernumber=Wrong order number

adyentest.ordernumber=Order number


```
#####
# Validation messages
#####
validate.required=This field is required.
validate.remote=Please fix this field.
validate.email=Please enter a valid email address.
validate.url=Please enter a valid URL.
validate.date=Please enter a valid date.
validate.dateISO=Please enter a valid date ( ISO ).
validate.number=Please enter a valid number.
validate.digits=Please enter only digits.
validate.creditcard=Please enter a valid credit card number.
validate.equalTo=Please enter the same value again.
validate.maxlength=Please enter no more than {0} characters.
validate.minlength=Please enter at least {0} characters.
validate.rangelength=Please enter a value between {0} and {1} characters
long.
validate.range=Please enter a value between {0} and {1}.
validate.max=Please enter a value less than or equal to {0}.
validate.min=Please enter a value greater than or equal to {0}.

forms.suite=Suite
forms.streetName=Street
forms.suiteerror=Please enter Suite number
forms.streetnameerror=Please enter Street name
forms.suite.field.invalid=Please enter a valid value
forms.streetname.field.invalid=Please enter a valid value
adyentest.missedordernumber=Missed order number
adyentest.wrongordernumber=Wrong order number
adyentest.ordernumber=Order number
```

Controller Model: Changes Needed

If using a controller method SiteGenesis integration, additionally follow the instructions on this page. If integrating via the pipeline method please see [Pipeline Model: Changes Needed](#).

COBilling.js

In the top script modules section add Adyen helpers:

```
var AdyenController =
require("int_adyen_controllers/cartridge/controllers/Adyen");
var AdyenHelper =
require("int_adyen/cartridge/scripts/util/AdyenHelper");
```

Around line 100, add the Adyen Helper as a parameter to the app.getView calls:

```
if (params) {
    app.getView(require('~/cartridge/scripts/object').extend(params,
    {
        Basket: cart.object,
        AdyenHelper : AdyenHelper,
        ContinueURL: URLUtils.https('COBilling-Billing')
    })).render('checkout/billing/billing');
} else {
    app.getView({
        Basket: cart.object,
        AdyenHelper : AdyenHelper,
        ContinueURL: URLUtils.https('COBilling-Billing')
    }).render('checkout/billing/billing');
}
```

Around line 160, adjust this line that gets the applicableCreditCards:

```
        if (customer.authenticated) {
            var profile = app.getModel('Profile').get();
            if (profile) {
                applicableCreditCards =
profile.getWallet().getPaymentInstruments(PaymentInstrument.METHOD_CREDIT_CARD);
            }
        }

        return {
            ApplicablePaymentMethods: applicablePaymentMethods,
            ApplicableCreditCards: applicableCreditCards
        };
    }
}
```

In the `publicStart` function around line 174, make the following changes around line 182 and 199:

```

/**
 * Starting point for billing. After a successful shipping setup, both
COShipping
 * and COShippingMultiple call this function.
 */
function publicStart() {
    var cart = app.getModel('Cart').get();
    if (cart) {

        // Initializes all forms of the billing page including: -
address form - email address - coupon form
        initAddressForm(cart);
        initEmailAddress(cart);

        // Get the Saved Cards from Adyen to get latest saved cards

require('int_adyen/cartridge/scripts/UpdateSavedCards').updateSavedCards
({CurrentCustomer : customer});

        var creditCardList = initCreditCardList(cart);
        var applicablePaymentMethods =
creditCardList.ApplicablePaymentMethods;

        var billingForm = app.getForm('billing').object;
        var paymentMethods = billingForm.paymentMethods;
        if (paymentMethods.valid) {

paymentMethods.selectedPaymentMethodID.setOptions(applicablePaymentMetho
ds.iterator());
        } else {
            paymentMethods.clearFormElement();
        }

        app.getForm('billing.couponCode').clear();
        app.getForm('billing.giftCertCode').clear();

        var AdyenHppPaymentMethods =
AdyenController.GetPaymentMethods(cart);

        start(cart, {ApplicableCreditCards:
creditCardList.ApplicableCreditCards, AdyenHppPaymentMethods :
AdyenHppPaymentMethods});
        } else {
            app.getController('Cart').Show();
        }
    }
}

```

Around line 751, add the following conditional code to the validatePayment function:

```
function validatePayment(cart) {
    var paymentAmount, countryCode, invalidPaymentInstruments, result;
    if (AdyenHelper.getAdyenCseEnabled()) {
        result = true;
        return result;
    }
    if (app.getForm('billing').object.fulfilled.value) {
        paymentAmount = cart.getNonGiftCertificateAmount();
        countryCode = Countries.getCurrent({
            CurrentRequest: {
                locale: request.locale
            }
        }).countryCode;

        invalidPaymentInstruments =
        cart.validatePaymentInstruments(customer, countryCode,
        paymentAmount.value).InvalidPaymentInstruments;

        if (!invalidPaymentInstruments &&
        cart.calculatePaymentTransactionTotal()) {
            result = true;
        } else {
            app.getForm('billing').object.fulfilled.value = false;
            result = false;
        }
    } else {
        result = false;
    }
    return result;
}
```

Around line 787, make the following changes to the saveCreditCard function. Don't forget the closing curly bracket right after the return statement near the end of the function:

```

function saveCreditCard() {
    if (AdyenHelper.getAdyenRecurringPaymentsEnabled()) {
        // saved credit cards are handling in COPlaceOrder
        and Login for Adyen - saved cards are synced with
        Adyen ListRecurringDetails API call
        return true;
    } else {
        var i, creditCards, newCreditCard;
        if (customer.authenticated &&
app.getForm('billing').object.paymentMethods.creditCard.saveCard.value)
        {
            creditCards =
customer.getProfile().getWallet().getPaymentInstruments(PaymentInstrumen
t.METHOD_CREDIT_CARD);
            Transaction.wrap(function() {
                newCreditCard =
customer.getProfile().getWallet().createPaymentInstrument(PaymentInstrum
ent.METHOD_CREDIT_CARD);
                // copy the credit card details to the
                payment instrument

newCreditCard.setCreditCardHolder(app.getForm('billing').object.paymentM
ethods.creditCard.owner.value);

newCreditCard.setCreditCardNumber(app.getForm('billing').object.paymentM
ethods.creditCard.number.value);

newCreditCard.setCreditCardExpirationMonth(app.getForm('billing').object
.paymentMethods.creditCard.expiration.month.value);

newCreditCard.setCreditCardExpirationYear(app.getForm('billing').object.
paymentMethods.creditCard.expiration.year.value);

newCreditCard.setCreditCardType(app.getForm('billing').object.paymentMet
hods.creditCard.type.value);
                for (i = 0; i < creditCards.length; i++) {
                    var creditcard = creditCards[i];
                    if (creditcard.maskedCreditCardNumber ===
newCreditCard.maskedCreditCardNumber && creditcard.creditCardType ===
newCreditCard.creditCardType) {

customer.getProfile().getWallet().removePaymentInstrument(creditcard);
                    }
                }
            });
        }
        return true;
    }
}

```

COPlaceOrder.js

At the top of the file in the Script Module requires, add in a line to require in the Adyen Helper:

```
var AdyenHelper =  
require('int_adyen/cartridge/scripts/util/AdyenHelper');
```

In the handlePayments function around line 38, make the following changes at line 56, 72-76, 78-80 to the handlePayments code:

```
function handlePayments(order) {  
  
    if (order.getTotalNetPrice() !== 0.00) {  
  
        var paymentInstruments = order.getPaymentInstruments();  
  
        if (paymentInstruments.length === 0) {  
            return {  
                missingPaymentInfo: true  
            };  
        }  
        /**  
        * Sets the transaction ID for the payment instrument.  
        */  
        var handlePaymentTransaction = function () {  
  
paymentInstrument.getPaymentTransaction().setTransactionID(order.getOrderNo());  
        };  
  
        var show3dSecureForm : Boolean = false;  
        for (var i = 0; i < paymentInstruments.length; i++) {  
            var paymentInstrument = paymentInstruments[i];  
  
            if  
(PaymentMgr.getPaymentMethod(paymentInstrument.getPaymentMethod()).getPaymentProcessor() === null) {  
  
                Transaction.wrap(handlePaymentTransaction);  
  
            } else {  
                var authorizationResult =  
PaymentProcessor.authorize(order, paymentInstrument);  
                if (authorizationResult.not_supported ||  
authorizationResult.error) {  
                    return {  

```

```

        error: true
    };
}

if
(PaymentMgr.getPaymentMethod(paymentInstrument.getPaymentMethod()).getPa
ymentProcessor().ID === 'ADYEN_CREDIT'
    && authorizationResult.authorized3d === true) {
    var show3dSecureForm : Boolean = true;
    var view : Object = authorizationResult.view;
}
}
}
if (show3dSecureForm) {
    return {view : view};
}
}

```



```
    return {};  
}
```

Around line 160, near the end of the start function, make the following changes at line 162, 166-168, 189-206:

```
if (!order) {  
    // TODO - need to pass BasketStatus to Cart-Show ?  
    app.getController('Cart').Show();  
  
    return {};  
}  
  
var skipSubmitOrder : Boolean = false;  
  
var handlePaymentsResult = handlePayments(order);  
  
if (!empty(handlePaymentsResult.view)) {  
    skipSubmitOrder = true;  
}  
  
if (handlePaymentsResult.error) {  
    return Transaction.wrap(function () {  
        OrderMgr.failOrder(order);  
        return {  
            error: true,  
            PlaceOrderError: new Status(Status.ERROR,  
'confirm.error.technical')  
        };  
    });  
  
    } else if (handlePaymentsResult.missingPaymentInfo) {  
        return Transaction.wrap(function () {  
            OrderMgr.failOrder(order);  
            return {  
                error: true,  
                PlaceOrderError: new Status(Status.ERROR,  
'confirm.error.technical')  
            };  
        });  
    }  
  
    if (order.paymentInstrument.paymentMethod == "Adyen") {  
        return {  
            Order: order,  
            order_created: true  
        };  
    }  
}
```

```
} else {
  if (skipSubmitOrder) {
    return {
      Order: order,
      order_created: true,
      view : handlePaymentsResult.view,
      skipSubmitOrder : skipSubmitOrder
    };
  } else {
    var orderPlacementStatus = Order.submit(order);
  }
}
if (!orderPlacementStatus.error) {
  clearForms();
}
```

```
        return orderPlacementStatus;
    }
}
```

COSummary.js:

At the top of the file add controller include:

```
var AdyenController = require("int_adyen_controllers/cartridge/controllers/Adyen");
```

in function submit replace SG default code to include a Redirect

The resulting code should look like this:

```
/**
 * This function is called when the "Place Order" action is triggered by
the
 * customer.
 */
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) {
        if (placeOrderResult.Order.paymentInstrument.paymentMethod ==
"Adyen") {
            AdyenController.Redirect(placeOrderResult.Order);
        } else {
            if (placeOrderResult.skipSubmitOrder === true) {
                placeOrderResult.view.render('adyenform');
            } else {
                showConfirmation(placeOrderResult.Order);
            }
        }
    }
}
}
```

PaymentInstruments.js:

At the top of the file, add the following API and Script Module includes:

```

/* API includes */
var PaymentInstrument = require('dw/order/PaymentInstrument');
var PaymentMgr = require('dw/order/PaymentMgr');
var PaymentStatusCodes = require('dw/order/PaymentStatusCodes');
var Status = require('dw/system/Status');
var Transaction = require('dw/system/Transaction');
var URLUtils = require('dw/web/URLUtils');

/* Script Modules */
var app = require('~/cartridge/scripts/app');
var guard = require('~/cartridge/scripts/guard');
var AdyenHelper =
require('int_adyen/cartridge/scripts/util/AdyenHelper');
var Logger = require('dw/system/Logger');

```

In the list() function around line 33, add the following line to get saved cards from Adyen:

```

function list() {

    // Get the Saved Cards from Adyen to get latest saved cards

    require('int_adyen/cartridge/scripts/UpdateSavedCards').updateSavedCards
    ({CurrentCustomer : customer});

    var wallet = customer.getProfile().getWallet();
    var paymentInstruments =
wallet.getPaymentInstruments(dw.order.PaymentInstrument.METHOD_CREDIT_CA
RD);
    var pageMeta = require('~/cartridge/scripts/meta');
    var paymentForm = app.getForm('paymentinstruments');

    paymentForm.clear();

    paymentForm.get('creditcards.storedcards').copyFrom(paymentInstruments);

    pageMeta.update(dw.content.ContentMgr.getContent('myaccount-paymentsetti
ngs'));

    app.getView({
        PaymentInstruments: paymentInstruments
    }).render('account/payment/paymentinstrumentlist');
}

```

In the create() function around line 133, make the following changes:

```
function create() {
    if (!verifyCreditCard()) {
        return false;
    }

    var paymentForm = app.getForm('paymentinstruments');
    var newCreditCardForm =
paymentForm.get('creditcards.newcreditcard');
    var ccNumber = newCreditCardForm.get('number').value();

    var wallet = customer.getProfile().getWallet();
    var paymentInstruments =
wallet.getPaymentInstruments(dw.order.PaymentInstrument.METHOD_CREDIT_CA
RD);
    if (AdyenHelper.getAdyenRecurringPaymentsEnabled()) {
        var createRecurringPaymentAccountResult =
AdyenHelper.createRecurringPaymentAccount({
            Customer: customer
        });

        if (createRecurringPaymentAccountResult.error) {
            return false;
        }

        pspReference = 'PspReference' in
createRecurringPaymentAccountResult &&
!empty(createRecurringPaymentAccountResult.PspReference) ?
createRecurringPaymentAccountResult.PspReference : '';
        tokenID = 'TokenID' in createRecurringPaymentAccountResult &&
!empty(createRecurringPaymentAccountResult.TokenID) ?
createRecurringPaymentAccountResult.TokenID : '';
        /*if (empty(TokenID) || empty(PspReference)) {
            return false;
        }*/
        try {
            Transaction.wrap(function() {
                /* var newCreditCard =
customer.getProfile().getWallet().createPaymentInstrument(PaymentInstrum
ent.METHOD_CREDIT_CARD);

                * // copy the credit card details to the
payment instrument
                * newCreditCard.setCreditCardHolder(
                    newCreditCard.setCreditCardNumber(
                    newCreditCard.setCreditCardType(

                    newCreditCard.setCreditCardToken(tokenID);

                    newCreditCard.custom.AdyenPspReference =
pspReference; */
```

```
require('int_adyen/cartridge/scripts/UpdateSavedCards').updateSavedCards
({
    CurrentCustomer: customer,
    PaymentsMap:
createRecurringPaymentAccountResult.PaymentsMap
    });
});
} catch (e) {
    Logger.error('{0}: {1}', e, e.stack);
    return false;
}
return true;
}
```

```
var isDuplicateCard = false;
var oldCard;
```

In the Delete() function around line 219, make the following changes:

```
function Delete() {
    var paymentForm = app.getForm('paymentinstruments');
    paymentForm.handleAction({
        remove: function (formGroup, action) {
            Transaction.wrap(function () {
                var wallet = customer.getProfile().getWallet();
                //wallet.removePaymentInstrument(action.object);
                var paymentInstrument = action.object;
                if (!empty(paymentInstrument)) {

                    if (AdyenHelper.getAdyenRecurringPaymentsEnabled() &&
                        !empty(paymentInstrument.getCreditCardToken())) {
                        var result =
require('int_adyen/cartridge/scripts/adyenDeleteRecurringPayment').delet
eRecurringPayment({

                            Customer: customer,

                            RecurringDetailReference:
paymentInstrument.getCreditCardToken()

                        });
                        if (result == PIPELET_NEXT) {

wallet.removePaymentInstrument(paymentInstrument);
                        }
                    }
                    else {
                        wallet.removePaymentInstrument(paymentInstrument);
                    }
                }
            });
        },
        error: function () {
            // @TODO When could this happen
        }
    });
    response.redirect(URLUtils.https('PaymentInstruments-List'));
}
```

Add one "if" statement in the verifyCreditCard() function around line 255:

```
function verifyCreditCard() {  
    var newCreditCardForm =  
app.getForm('paymentinstruments.creditcards.newcreditcard');  
    if (AdyenHelper.getAdyenCseEnabled()) {  
        return true;  
    }  
  
    var expirationMonth =  
newCreditCardForm.get('expiration.month').value();  
    var expirationYear =  
newCreditCardForm.get('expiration.year').value();  
    var cardNumber = newCreditCardForm.get('number').value();  
}
```


Pipeline Model: Changes Needed

If using a pipeline method SiteGenesis integration, additionally follow the instructions on the dw_integration_adyen version v16.1.0.docx document which can be obtained from Adyen. Follow starting at section 3.5:

3.5 Changes to pipelines, templates and others

Otherwise, it's highly recommended to use the newer [Controller Integration Model](#).

Additional Configuration

You may need to change the ID of the storefront application. This can be defined in the file `int_adyen.properties` in the root directory of the `int_adyen` cartridge.

Example:

```
# defines what ID the storefront application cartridge has - default is app_tdc  
demandware.core.cartridge=SiteGenesisNew
```

Please open the '`int_adyen.properties`' file and edit the value to your storefront cartridge name to make the adyen cartridge work.

```
## cartridge.properties for cartridge int_adyen  
#Thu Apr 14 17:37:49 CEST 2011  
demandware.cartridges.int_adyen.id=int_adyen  
demandware.cartridges.int_adyen.multipleLanguageStorefront=true  
  
# defines what ID the storefront application cartridge has - default is  
app_tdc  
demandware.core.cartridge=app_storefront_core
```

Client Side Encryption

For client side encryption to work properly, the following site preferences need to be configured:

Site Pref	Value
CSE Enabled	true
Javascript URL	not empty
Javascript Public Key	Proper value set from Adyen Portal.

Custom Objects

A Custom object is used in the cartridge to support Adyen notifications back to Commerce Cloud for payment status.

The custom object definition is imported as part of the site metadata import. See [Data Storage](#) for more information.

External Interfaces

Testing

Make sure you have an Adyen test account. Configure your account and then configure the Adyen settings in the BM.

Test Credit Cards

In case you want to test Credit Cards payment method, Adyen provides some test numbers you can use. Please check the following link as these numbers may change:

<https://docs.adyen.com/support/integration#testcardnumbers>

Business Manager Configuration

Please read the Business Manager Configuration section for the info about configuring BM.

The Debug Setting set to 'yes' shows a 'pay' button before doing the redirect to Adyen. This allows you to check the parameters sent to the HPP.

Testing API methods

There is a storefront cartridge in the distributive package, where there are added additional pages for testing API methods implemented in the cartridge. The current implementation of the cartridge includes the following APIs:

- • Capture
- • Cancel before the capture
- • Cancel or refund

Here is the instruction for testing API methods.

Capture & Cancellation Payment

Storefront

1. In both cases you can see the form with the field where you should enter the order number
2. Once you provide the order number, you can click the submit button
3. You'll be redirected to the result page. On this page, in the grey area, you'll see the result of the operation you asked to perform
4. There are only 2 possible results of operation which you may see on result page:
 - a. 'Ok' – everything was ok, the data were processed correctly. This status means that the data were handled properly and no any unexpected situation happened, but it does not necessary means that the payment will be captured, see below the info about event codes which are available in BM
 - b. 'Error' – either the order was not found in DWRE system or some error occurred. If you see this message, nothing should be changed in BM, but theoretically something could be changed on Adyen side. See below the info about event codes which are available in BM

Business Manager

Once you see some result on the storefront, you may check the BM:

1. Find the order which you modified
2. Open it and go to Payment tab
3. The default value of Eventcode field is AUTHORISATION
4. If order payment was requested to be captured the following event codes are possible:
 - a. CAPTURING – the data were successfully handled and the request was sent to Adyen
 - b. CAPTURING REFUSED – Adyen refused payment capturing
5. If order payment was requested to be canceled the following event codes are possible:
 - a. CANCELLATION – the data were successfully handled and the request was sent to Adyen
 - b. CANCELLATION REFUSED – Adyen refused payment cancellation

6. If order payment was requested to be canceled or refunded the following event codes are possible:
 - a. CANCELLATION OR REFUND – the data were successfully handled and the request was sent to Adyen
 - b. CANCELLATION OR REFUND REFUSED – Adyen refused payment cancellation
7. After you send a request to Adyen, you see immediately only the result of this request, but not the result of operation which will be performed on Adyen side. Adyen works not in real time, so your request won't be handled immediately on Adyen side, but when the request will be handled, Adyen will change the status of payment in its back office and also the Eventcode will be changed on Demandware site. So you should check after some time which is the end result of the operation, in BM and in Adyen back office

Adyen back office

1. Once you created new order it'll be placed into Payment list in Adyen back office
2. When you perform some operations on payment, the status of payment is changed in an appropriate way. But this works not in real time, so when you send some request, wait a minute and only then check the result of operation. Also, when payment modification request is made from Adyen back office, an appropriate status of payment is shown in Adyen back office, but when I was working with the API, couldn't see some statuses (Sent for Settlement for example, or Sent for Cancellation)
3. Possible statuses of payment in Adyen back office should be searched for in Adyen documentation

Operations, Maintenance

Data Storage

For each order which results in a payment attempt on the Adyen HPP, the payment results are available in the following fields:

- PSP reference - unique Adyen ID of this payment. It's a link between Demandware order and Adyen payment
- Payment Method
- Eventcode - AUTHORISED, CANCELLATION, REFUND etc.
- Amount – Amount paid

They are handled by the `Adyen.notify()` controller method or Adyen-Notify pipeline, which receives the notification messages from Adyen. In one case, when a payment is refused on Adyen HPP, these fields are updated in `Adyen.showConfirmatino()` controller or Adyen-ShowConfirmation pipeline.

Notifications are queued in Commerce Cloud for a short duration, either until Adyen provides a "final" status for the order or the custom object retention period is reached.

Commerce Cloud stores a custom object briefly until notification has been received from Adyen. As the final status is obtained the CO is deleted. On the off chance that Adyen never responds to the order with a notification, Commerce Cloud will automatically remove the custom object instance after 72 hrs.

On top of that, all the communication to/from Adyen is logged in the log file directory available via webDAV. Please check this first in case you have any problems.

Also, all the notifications sent by Adyen are logged on the order level and can be found on Notifications tab of Order Details page in BM.

Availability

In case of problems with the connection to Adyen please contact the Adyen support department.

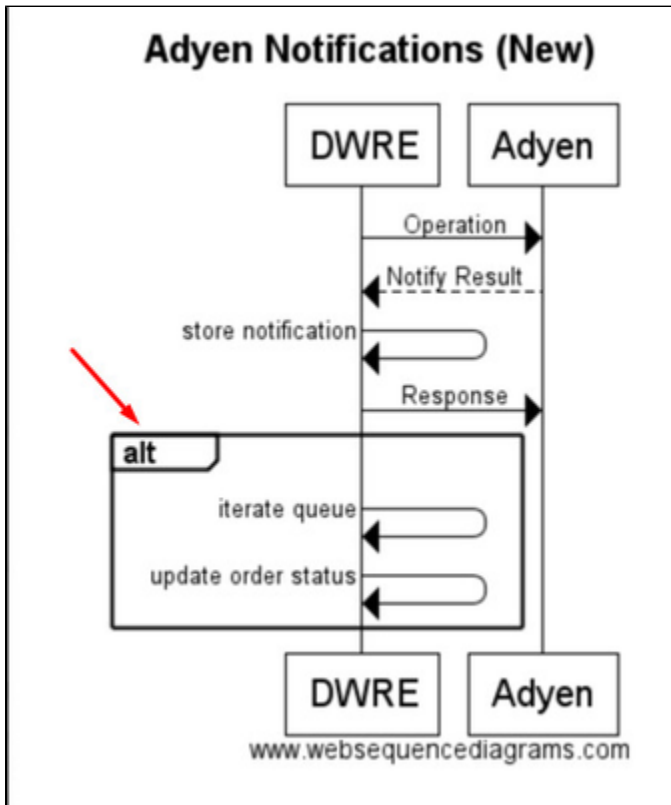
Please supply as much information as possible (Merchant account, skin code, time, order number, PSP reference etc.). Also check the log file mentioned above.

Support

In case of problems with the integration, missing features, etc. please contact the Adyen support department or your Adyen account manager.

Scheduled Jobs

The Adyen integration to the notification system requires a scheduled job to be configured and run on a regular basis.



The Adyen servers will call the Commerce Cloud endpoint with updates to payment transactions. Commerce Cloud will store these as custom object instances queued for processing. The scheduled job is responsible for iterating through the received notifications and updating the order payment transactions and order status appropriately.

Setup a site level job for each site using a Adyen configuration to run the Adyen notifications process. Each client will need to determine the schedule and frequency on which this runs based on their individual needs.

Step 1: Create new job ID with Priority Normal

Edit Job AdyenProcessNotifications2 [?]

General Schedule and History Resources Step Configuration

ID*
AdyenProcessNotifications2

Description
Execute adyen notification related handling

Priority
☒ Normal ☐ High

Step 2: Setup two steps for the job in the Step Configuration:

Administration / Operations / Jobs /

Edit Job AdyenProcessNotifications2 [?]

Run Now

General Schedule and History Resources Step Configurator Notification Failure Handling

Scope: LyonsSiteGenesis2

Process

+

Scope: LyonsSiteGenesis2

Clean

+

The contents of these steps are to be configured as follows:

Edit Job AdyenProcessNotifications2 ?

General Schedule and History Resources Step Configurator Notification Failure

Scope: LyonsSiteGenesis2

Process

Scope: LyonsSiteGenesis2

Clean

Select and configure step

ID*

Process

Description

ExecuteScriptModule.Module*

int_adyen/cartridge/scripts/job/notifications.js

ExecuteScriptModule.FunctionName

processNotifications

☒ ExecuteScriptModule.Transactionnal

ExecuteScriptModule.TimeoutInSeconds

3600

☐ Restart Enforced

Custom Parameters

ID*

Value*

Error Handling

On

ERROR

Action

Stop Job

← Back

Assign

Administration / Operations / Jobs /

Edit Job AdyenProcessNotifications2

General Schedule and History Resources Step Configurator Notification Failure

Scope: LyonsSiteGenesis2

Process

LyonsSiteGenesis2

Clean

Select and configure step

ID*
Clean

Description

ExecuteScriptModule.Module*
int_adyen/cartridge/scripts/job/notifications.js

ExecuteScriptModule.FunctionName
clearNotifications

☒ ExecuteScriptModule.Transactional

ExecuteScriptModule.TimeoutInSeconds
3600

☐ Restart Enforced

Custom Parameters

ID* Value*

Error Handling

On
ERROR

Action
Stop Job

© 2017 salesforce.com, LyonsSiteGenesis2 Time Zone: Coordinated Universal Time|Instance Time Zone: Eastern Standard Time|Version: 17.2 , last updated Jan 30, 2017 (Compatibility Mode: 16.2)

Testing

Notifications should result in orders having their payment transaction information stored in the order and, if complete, order submitted and "ready for export".



Sites: (4)

Site Genesis

▼ Site - Site Genesis

- Storefront
- Products and Catalogs
- Content
- Search
- Online Marketing
- Customers
- Custom Objects

Ordering

- Analytics
- Site URLs
- Site Preferences

► Administration

[Site](#) > [Ordering](#) > [Orders](#) > Order: ad01-00006580[General](#) [Attributes](#) [Payment](#) [Notes](#) [History](#)

Payment Information for Order 'ad01-00006580'

Order Total: \$105.87

[Amount Paid:](#) \$0.00

Balance Due: \$105.87

Invoice Number: 00021604

[Payment Status:](#) Paid

Payment Method: Adyen
Processor: Adyen
Transaction:
Amount: \$105.87

[Billing Address](#)

Payment info

PSP reference	7914204984750298
Payment Method	mc
Eventcode	AUTHORISATION
Amount	105.87

User Guide

Roles and Responsibilities

The store administrator should check the correct configuration of the Adyen Merchant account in the Adyen back office (CA) and should check the receiving of the Adyen notification messages on a regular basis.
Also, check the payments received to see if they contain the expected data (currencyCode, amount, shopperEmail, etc)

Business Manager

Please configure the settings relevant to the Adyen integration as described in section 3.2.

After an authorized payment, the Payments tab of the order should look like this:

General

Attributes

Payment

Notes


History

Payment Information for Order 'ad01-00005482'

Order Total:	\$105.87
Amount Paid:	\$0.00
Balance Due:	\$105.87

Invoice Number:	00016606
Payment Status:	Paid

Payment Method:	Adyen Processor: Adyen Transaction: Amount: \$105.87
-----------------	---------------------------------------------------------------



Payment info

PSP reference	8814199514951011
Payment Method	visa
Eventcode	AUTHORISATION
Amount	105.87

All possible Eventcodes are described in a separate document from the distributive package and partially described on section 2.2. Use Cases.

Note that you can click on the PSP reference to go directly to the payment details in the Adyen back office (if you have logged in first).

In the Adyen back office you can also link back to DW by setting Merchant Settings > Merchant Reference Link to:

https://adyen01.tech-prtnr-eu01.dw.demandware.net/on/demandware.store/Sites-Site/default/ViewOrder_52-FindByNumber?OrderNo= (change to your own site name)

Storefront Functionality

New storefront functionality is described in section [Use Cases](#)

Known Issues

Version	Issue	Description	Work Around
3	Session validity' and 'Shipbeforedate' are not used		
3	Partial payments are not supported		

Release History

Version	Date	Changes
1.0.0	April 27, 2011	Initial release
1.0.1	May 20, 2011	GC changes
2	June 06,2013	Plug and play, API payments
2	August 16,2013	Minor changes to documentation
3	January 8, 2015	Full refactoring and improvement of the document, fixed a lot of mistakes. Added description of new API methods, AVS, 3-D Secure
4	October, 2016	Updated cartridge for the following: <ul style="list-style-type: none">• Controllers Support (js)• Notification Handling Updates• Support for Multiple Service Frameworks• Fixed for 3D-Secure handling• Support for currencies with fraction digits• SHA-2 support• CSE support
5	January, 2018	Updated cartridge for the following: <ul style="list-style-type: none">• (HPP) Open Invoice Payments<ul style="list-style-type: none">• Klarna• AfterPay• RatePay• Saved Cards (OneClick, Recurring)• Additional Order Payment Transaction attributes stored with order• Additional code cleanup

Version History

Version Number	Change Description	Date	Author