Implementation Guide - Wallee Payment Solution for SAP Commerce Cloud

Table of contents

Index of Figures

Index of Tables

1 Introduction

The purpose of providing an extension for Wallee Payment Solution for SAP Commerce Cloud is to aid the integration of wallee into a SAP Commerce Cloud (previously known as SAP Hybris) implementation. SAP Commerce Cloud Store accelerator will use the newly developed extension to interact with wallee.

The Wallee Payment Solution for SAP Commerce Cloud extension provides a configurable spring component which helps shorten the implementation cycle and reduce system development, testing efforts and maintenance costs for payment integration. The extension package contains most of the backend code needed for creating transactions using wallee into the SAP Commerce Cloud platform. The Merchants can use the wallee Payment Solution for SAP Commerce Cloud extension with their SAP Commerce Cloud project to integrate wallee.

This document contains information about the functionalities and Payment Integration details of the wallee Payment Solution for SAP Commerce Cloud.

2 Functional Architecture

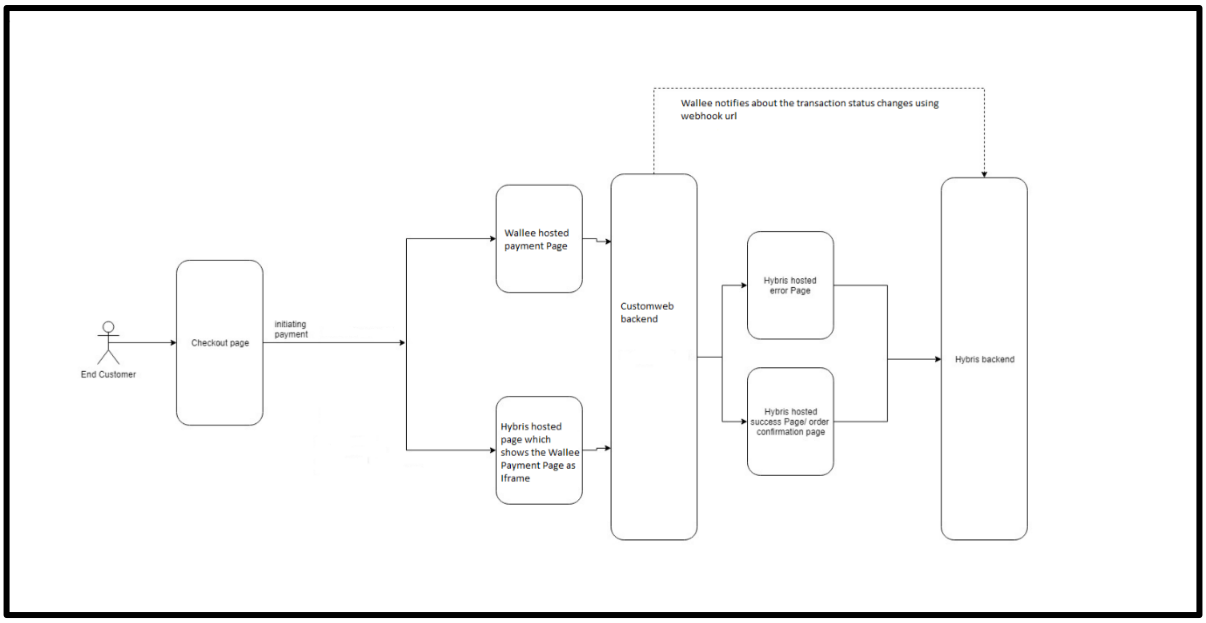

Wallee notifies about the transaction status changes using webhook url
End Customer
Checkout page
initiating payment
Wallee hosted payment Page
Hybris hosted page which shows the Wallee Payment Page as Iframe
Customweb backend
Hybris hosted error Page
Hybris hosted success Page/ order confirmation page
Hybris backend

Figure 2.1: Functional Diagram

3 Technical Overview

The Wallee Payment Solution for SAP Commerce Cloud extension is built on SAP Commerce Cloud (previously known as SAP Hybris) version 1905. Thus, make sure you have the project running with version 1905.

4 Technologies and Tools

The following table summarizes the proposed platform and software recommended for the system hosting.

Sl. No.	Component	Description
1	Operating Platform	Windows7+ /macOS/Linux
2	Web Server	Tomcat
3	Technology	SAP Commerce Cloud/SAP Hybris

Table 1: Application Software

Sl. No	Component	Description
1	Development Environment	Intellij IDE
2	Source Control	Git

Table 2: Development Tools

5 Plugin Requirements Scope

The following functionalities are covered as part of the “Wallee Payment Solution for SAP Commerce Cloud Plugin”. In order to create a payment in wallee, there is a choice between Payment Page Integration and iframe Integration.

5.1 Payment Page Integration

The Payment Page Integration helps in redirecting the customer to the wallee provided payment page where customers can enter the payment details. It allows the processing of all payment methods that are configured with wallee.

- The process begins with the creation of an order in your shop.
- Then you have to create Transaction object using the create method on Transaction Service.
- Use the buildPaymentPageUrl service to create the Payment Page Url.
- Redirect the customer to the Payment Page Url.
- When the transaction is processed or failed, the customer will be redirected to the successUrl or failedUrl that was defined when creating the transaction object.
- Listen to the notification via webhook url, to mark the order in the merchant system as authorized or failed.

5.2 IFrame Integration

The iframe Integration is used in implementing a custom checkout experience by building a payment form in an iframe using wallee provided JavaScript or the lightbox integration in order to achieve a seamless and PCI DSS compliant integration in your checkout.

- Create a transaction object with the Transaction service. To create a transaction object, you can provide all the information you have in this state.
- Once the transaction object is created the possible payment methods can be fetched by using fetchPossiblePaymentmethods on the Transaction Service by providing the transactionId returned by the initial request.
- To embed the iframe into a website a JavaScript URL has to be fetched. For this the service method buildJavaScriptUrl can be used. The form is then validated.
- The transaction is confirmed and the application reports the final state of transaction. The iframe is submitted with JavaScript.
- When the transaction is processed or failed, the customer will be redirected to the successUrl or failedUrl that was defined when creating the transaction object.

5.3 Webhook Configuration

A webhook can be used to notify another application about events. Webhooks are generally triggered by some event for example to notify your application about a status change of an entity. All configuration details can be set inside the space. You can define here the URL as well as chose from the different events for which you want to receive a HTTP call.

5.3.1 URLs

All URLs you want a webhook delivered to are configured centrally inside our space in wallee. Managing the URLs centrally helps you to easily manage changes to URLs so that you not have to touch every webhook listener configuration.

5.3.2 Webhook Listeners

A listener can listen on state changes of particular entities. Means when an entity changes into a specified state the URL will be invoked. Only certain entities are enabled to be subscribed by a webhook listener. A Webhook Listener can be created under Webhook Listener Configuration in wallee.

5.3.3 Invocations

A log of all triggered webhooks can be seen under invocations in webhooks in wallee. By clicking on the invocation, you can see the error in case the webhook failed including the response that we received from your server.

6 Additional Features Considered

- Plugin must work on most common browsers supported by the shop-system
- Error Handling
- Payment methods are configured in account space in wallee.
- Provide wallee full access to place it on its own portals: Can be provided as ZIP fie.
- A webhook url is configured in wallee, to get notifications of a successful transaction.

7 Implementation Guide

The purpose of providing an extension for wallee is to aid the integration of wallee payment service into a SAP Commerce Cloud implementation. This document describes how to install and configure the extension to work with any SAP Commerce Cloud implementation.

As SAP Commerce Cloud is built on the spring framework this makes it highly customizable and extensible. The extension also utilizes this framework, so it can also be easily extended to add specific behavior if required.

7.1 Prerequisite

1. Before starting with the integration of the plugin you should sign up with wallee and create a space.
2. This extension is built on SAP Commerce Cloud (previously known as SAP Hybris) version 1905. Thus, make sure you have the project running with version 1905.

7.2 Installation and Usage

The plugin is supplied as a zip file. Take the following steps to include the extension into your Hybris application:

1. Unzip the supplied zip file.
2. Copy the extracted folder to ***${HYBRIS_CUSTOM_DIR}*** of your Hybris installation.
3. Add the Customweb extensions to your ***localextensions.xml*** file.

```
<extension name='customweb' />
<extension name='customwebbackoffice' />
<extension name='customwebcheckoutaddon'/>
```

4. Install ***customwebcheckoutaddon*** to the storefront in order to make the changes applicable to the storefront.

```
ant addoninstall -Daddonnames="customwebcheckoutaddon" -
DaddonStorefront.yacceleratorstorefront="XXXstorefront"
```

5. Run the ‘ant clean updatesystem’ command from your ***${HYBRIS_PLATFORM_DIR}.***

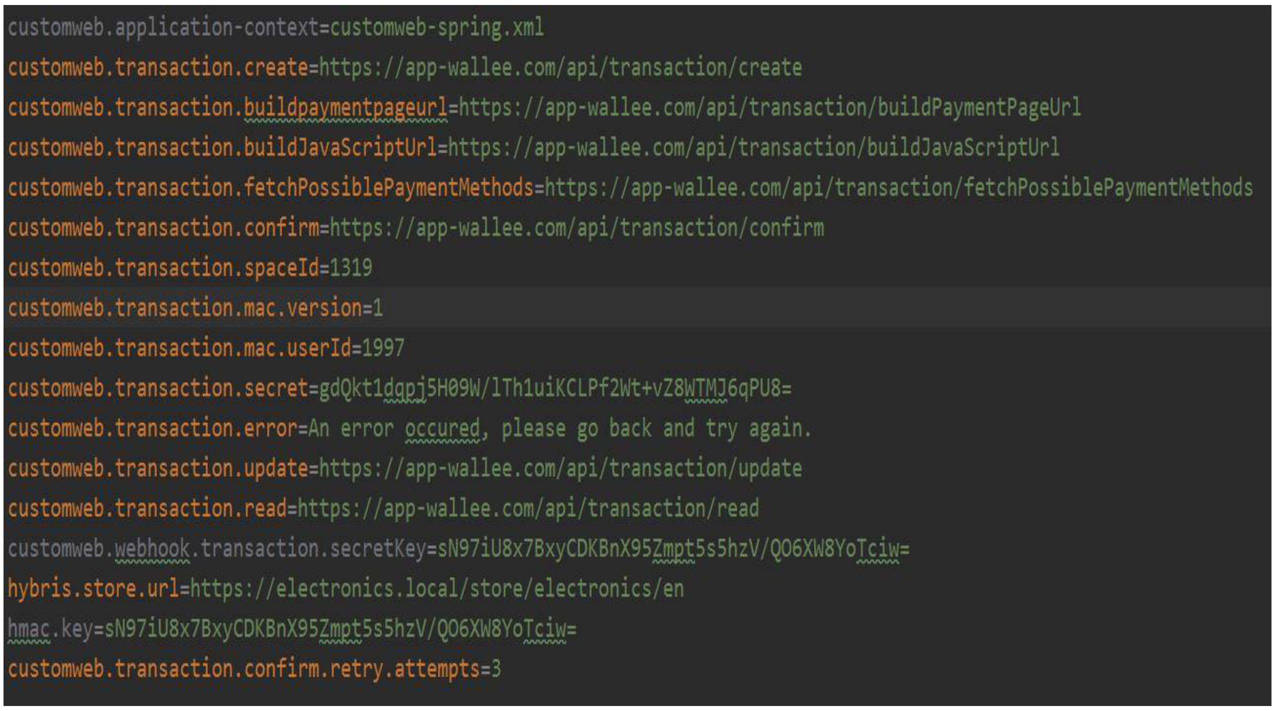
6. Run the hybrisserver.sh/bat to start up the hybris server.

7. Import the project data of the customweb extension from hac.

8. Or skip the steps from 5-7 and do an initialize and then start the server.

7.3 Configuring the Properties

Wallee Payment Solution for SAP Commerce Cloud uses some properties in order to manage the process. Those properties can be seen in the project properties. Some of the main properties are listed below:



```
customweb.application-context=customweb-spring.xml
customweb.transaction.create=https://app-wallee.com/api/transaction/create
customweb.transaction.buildpaymentpageurl=https://app-wallee.com/api/transaction/buildPaymentPageUrl
customweb.transaction.buildJavaScriptUrl=https://app-wallee.com/api/transaction/buildJavaScriptUrl
customweb.transaction.fetchPossiblePaymentMethods=https://app-wallee.com/api/transaction/fetchPossiblePaymentMethods
customweb.transaction.confirm=https://app-wallee.com/api/transaction/confirm
customweb.transaction.spaceId=1319
customweb.transaction.mac.version=1
customweb.transaction.mac.userId=1997
customweb.transaction.secret=gdQkt1dqpj5H09W/lTh1uiKCLPf2Wt+vZ8WTMJ6qPU8=
customweb.transaction.error=An error occured, please go back and try again.
customweb.transaction.update=https://app-wallee.com/api/transaction/update
customweb.transaction.read=https://app-wallee.com/api/transaction/read
customweb.webhook.transaction.secretKey=sN97iU8x7BxyCDKBnX95Zmpt5s5hzV/QO6XW8YoTciw=
hybris.store.url=https://electronics.local/store/electronics/en
hmac.key=sN97iU8x7BxyCDKBnX95Zmpt5s5hzV/QO6XW8YoTciw=
customweb.transaction.confirm.retry.attempts=3
```

Those properties have names that speak for itself. The transaction create request, build payment page url request etc. are configured here. The spaceId, mac values generated from wallee are also configured here.