TECHNICAL CARTRIDGE DOCUMENTATION

ScalaPay

LINK INTEGRATION (Pipelines & Controllers)

VERSION 19.1.0

Document version: 2.1.0

TABLE OF CONTENTS

1. SUMMARY

This document provides the guidelines for the implementation of the SCALAPAY "Site Genesis"(Pipelines and Controllers) enabled cartridge to SalesForce Commerce Cloud (SFCC). This document describes the various components: ScalaPay Payment Method, ScalaPay Widget.

2. COMPONENT OVERVIEW

2.1 Functional Overview

This cartridge extends store payment methods with ScalaPay payment method and payment method related functionality.

2.2 Use Cases

ScalaPay widget (Product Page):
Go to Product Detail Page(PDP), select product variation(color/size), find displayed ScalaPay text below the product pricing - find and click on "ⓘ" OR ScalaPay logo. Expected: Dialog window with ScalaPay content should be opened, displayed content is localised according to customer locale(if locale is supported by ScalaPay).

ScalaPay widget (Cart Page):
Go to Product Detail Page(PDP), select product variation(color/size) and add product to shopping cart. Open shopping cart page - find displayed ScalaPay text below the cart totals pricing - find and click on "ⓘ" OR ScalaPay logo. Expected: Dialog window with ScalaPay content should be opened, displayed content is localised according to customer locale(if locale is supported by the ScalaPay).

ScalaPay Payment Method: Open site storefront, go to any product details page, add product to shopping cart and proceed to checkout. On checkout payment page find and select ScalaPay payment method - place an Order with ScalaPay payment method. Expected result: Order was placed, order payment method is ScalaPay, Order contains all required address and products data.

Disable ScalaPay for Products in Category: Login to Business manager, go to: Merchant Tools -> Products and Catalogs -> Catalog -> find and open storefront catalog -> find and open category settings. Find and open tab "Category Attributes" -> find checkbox with name "Disable ScalaPay displaying for products in category:" -> check the checkbox. Expected result: ScalaPay widget and text should not be displayed for products where current category selected as "primary category"

Enable/Disable Checkout Summary page: Login to Business Manager, go to: Merchant Tools -> Site Preferences -> Custom Preferences -> Find and open custom preference group with ID scalapay_general -> find field "Enable Summary Page" -> set to "Yes" to enable summary page or "No" to disable it -> apply modified settings

Enable/Disable Payment Capture Delay: Login to Business Manager, go to: Merchant Tools -> Site Preferences -> Custom Preferences -> Find and open custom preference group with ID scalapay_general -> find field "Delay Capture Enabled" -> set to "Yes" to enable payment capture delay or "No" to use immediate payment capture.

2.3 Limitations, Constraints

1. To support ScalaPay payment method - SFCC instance should be configured with EUR currency, since ScalaPay supports this currency only.
2. Product Sets and Product Bundles are not supported in scope of ScalaPay integration cartridge.

2.4 Compatibility

Built on Commerce Cloud 19.1.0 (Compatibility Mode 18.10) and site genesis revision 105.0.0

3. IMPLEMENTATION GUIDE

3.1 Set-Up

The following ScalaPay Integration tasks are included within the LINK Cartridge:
1. Installation of the cartridge
2. Import custom metadata
3. Set the newly-created metadata values (Site Preferences)
4. Configure services
5. Make storefront cartridge edits

3.2 Configuration(Pipelines)

1. Import the integration pipelines cartridges into the Commerce Cloud Studio Workspace:
   a. Open Commerce Cloud Studio
   b. Go to: File -> Import -> General -> Existing Projects Into Workspace
   c. Browse to the integration cartridges folder, select “int_scalapay_pipelines” and “int_scalapay_core” directories
   d. Click Finish.
   e. Click OK when prompted to link the cartridge to the sandbox

2. Assign cartridge to the site
   a. Log into Business Manager
   b. Go to: Administration -> Sites -> Manage Sites -> RefArch or required site
   c. Select the “Settings” tab
   d. Add “int_scalapay_pipelines:int_scalapay_core:” to the beginning of the cartridge path (or ensure it is in front of app_storefront_core and app_storefront_pipelines
   e. Click Apply

3.2.1 Configuration(Controllers)

1. Import the integration controllers cartridges into the Commerce Cloud Studio Workspace:
   a. Open Commerce Cloud Studio
   b. Go to: File -> Import -> General -> Existing Projects Into Workspace
   c. Browse to the integration cartridges folder, select “int_scalapay_controllers” and “int_scalapay_core” directories
   d. Click Finish.
   e. Click OK when prompted to link the cartridge to the sandbox

2. Assign cartridge to the site
   a. Log into Business Manager
   b. Go to: Administration -> Sites -> Manage Sites -> RefArch or required site
   c. Select the “Settings” tab
   d. Add “int_scalapay_controllers:int_scalapay_core:” to the beginning of the cartridge path (or ensure it is in front of app_storefront_core and app_storefront_controllers
   e. Click Apply

3.2.2 Import Metadata

1. Metadata setup:
   a. Within integration cartridge source folder find and ZIP “metadata” folder
   b. Log in to Business Manager
   c. Go to: Administration -> Site Development -> Site Import & Export
   d. Using file selector form - select and upload archived “metadata” folder

e. After metadata archive uploaded -> select and import uploaded archive using file radio-button select and “import” button.
Note: Metadata package contains pre-configured ScalaPay payment method, payment processor, site preferences and services.

3.2.3 Site Preferences

1. Site Preferences Configuration:
   a. Log into Business Manager
   b. Go to: Merchant Tools -> Site preferences -> Custom Preferences
   c. Find and open each of custom site preference group: “scalapay_general”, “scalapay_cart”, “scalapay_product”
   d. Configure ScalaPay functionality using the following settings:
   ScalaPay Enabled - set Yes/No to enable disable ScalaPay functionality(payment method and widget displaying)
   Scalapay API Token(required) - Scalapay Payment Authorization API token, used to authorize and capture the payment.
   ScalaPay Applicable Countries List(required) - List of site country codes applicable for ScalaPay.
   ScalaPay widget API URLs(required) - ScalaPay widget API URLs list.
   Default URLs to add:
   https://cdn.scalapay.com/js/scalapay-widget/webcomponents-bundle.js
   https://cdn.scalapay.com/js/scalapay-widget/scalapay-widget.js
   ScalaPay Number of Instalments(required) - ScalaPay Number of Instalments
   Product Page Widget Price Color - ScalaPay widget product page price color
   Cart Page Widget Text Size - ScalaPay widget cart page text size
   Product Page Price Box Selector - ScalaPay widget product page price selector
   Show ScalaPay Logo on Product Page - Show ScalaPay widget logo on product page
   Product Page Widget Logo Color - ScalaPay product page widget logo color
   Cart Page Widget Logo Color - ScalaPay cart page widget logo color
   Product Page Custom CSS Classes - ScalaPay product page widget custom CSS classes
   Product Page Widget Logo Size - ScalaPay product page widget logo size
   Product Page Widget Text Size - ScalaPay widget product page text size
   Cart Page Widget Price Color - ScalaPay widget cart page price color
   ScalaPay minimum amount value(required) - Scalapay minimum applicable amount value
   Show ScalaPay Logo on Cart Page - Show ScalaPay widget logo on cart page
   ScalaPay maximum amount value(required) - ScalaPay maximum applicable amount value
   Cart Page Widget Logo Size - ScalaPay cart page widget logo size
   Cart Page Custom CSS Classes - ScalaPay cart page widget custom CSS classes
   Product Page Widget Logo Vertical Alignment - Scalapay product page widget logo vertical alignment
   Product Page Price Selectors(required) - ScalaPay product page widget price selectors array
   Cart Page Price Box Selector - ScalaPay widget cart page price selector
   Cart Page Price Selectors(required) - ScalaPay cart page widget price selectors array
   Product Page Widget Logo Vertical Alignment - Scalapay cart page widget logo vertical alignment
   Enable Summary Page - Enable/Disable Checkout Summary page for scalapay payment method
   Delay Capture Enabled - Enable/Disable payment capture delay

3.3 Services

1. Import integration cartridge metadata
2. Set up ScalaPay services configuration
   a. Login to Business manager
   b. Go to: Administration -> Operations -> Services
   c. Find and open tab “Credentials”
   d. Find and open service credentials with name “scalapay.payment.authorize.credentials”
   e. Set ScalaPay service URL(staging endpoint URL set by default)
   f. Apply changes
   g. Perform the same actions for payment service credentials with name “scalapay.payment.handle.credentials” (staging endpoint URL set by default) and “scalapay.payment.authorize.delay.credentials”
   Note: username and password for both credentials not required and should be empty

3.4 Payment Method Setup

1. Login to SFCC BM
2. Go to: Merchant Tools -> Ordering -> Payment Processors
3. Create new payment processor with ID “SCALAPAY”
4. Go to: Merchant Tools -> Ordering -> Payment Methods
5. Create new payment method with the following settings:
   Name: ScalaPay
   ID: ScalaPay
   Payment processor: SCALAPAY

3.5 Custom Code(General)

Use customization files list provided below to customize site storefront cartridge with ScalaPay functionality.

1. FIle: int_scalapay_core/cartridge/scss/default/_checkout.scss
   Changes: Line 240, added the following code block:
   .scalapay-widget-cart{ float: right; }

2. File: int_scalapay_core/cartridge/templates/default/checkout/billing/paymentmethods.isml
   Changes: Line 4, added the following code:
   <isset name="isScalaPayEnabled" value="${require('*/cartridge/scripts/helpers/scalapayHelper').isPaymentApplicable()}" scope="page"/>
   Line 17, added the following code block:
   <isif condition="${paymentMethodType.value.equals('ScalaPay') && !isScalaPayEnabled}" >
   <iscontinue/>
   </isif>
   Line 154, added the following code:
   <isinclude template="checkout/billing/scalapaymethod" />

3. File: int_scalapay_core/cartridge/templates/default/checkout/cart/cart.isml
   Changes: Line 821, add the following code block:
   <isinclude template="scalapay/widgetapi" />
   <isinclude url="${URLUtils.https('Scalapay-ShowWidget', 'type', ‘cart’)}"/>

4. File: int_scalapay_core/cartridge/templates/default/checkout/billing/billing.isml
   Changes: Line 25, added the following code block:
   <isif condition="${request.httpParameterMap.isParameterSubmitted('error')}">
   ** **
   ${Resource.msg(request.httpParameterMap.error.stringValue, 'checkout', null)}

```
        </div>
    </isif>
```
Line 56, replaced importScript("cart/CartUtils.ds"); with
importScript("app_storefront_core:cart/CartUtils.ds");
Line 220, replaced importScript("util/ViewHelpers.ds"); with
importScript("app_storefront_core:util/ViewHelpers.ds");

5. File: int_scalapay_core/cartridge/templates/default/product/product.isml
   Changes: Line 14, added the following code:
   var loadScalaPayAPI = require('dw/system/Site').getCurrent().getCustomPreferenceValue('scalaPayEnabled');
   Line 18, added the following code:
   loadScalaPayAPI = false;
   Line 22, added the following code block:
```
<isif condition="${loadScalaPayAPI}" >
    <isinclude template="scalapay/widgetapi" />
 </isif>
```

6. File: int_scalapay_core/cartridge/templates/default/product/productcontent.isml
   Changes: Line 78, added the following code block:
```
<isscript>
    var source = pdict.CurrentHttpParameterMap.source.stringValue;
    var format = pdict.CurrentHttpParameterMap.format.stringValue;
    var showScalapayWidget = require('dw/system/Site').getCurrent().getCustomPreferenceValue('scalaPayEnabled');
    if (source == 'search' || source == 'quickview' || source == 'giftregistry' || source == 'wishlist') {
        showScalapayWidget = false;
    }
</isscript>
<isif condition="${showScalapayWidget}" >
    <isinclude url="${URLUtils.https('Scalapay-ShowWidget', 'type', 'product', 'pid', pdict.Product.ID)}"/>
</isif>
```

3.5.1 Custom Code(Controllers)

1. File: app_storefront_controllers/cartridge/scripts/app.js
   Changes: Line 90, replaced
   return require('~/cartridge/controllers/' + controllerName);
   with
   return require('*/cartridge/controllers/' + controllerName);

2. File: int_scalapay_controllers/cartridge/controllers/COBilling.js
   Changes: replaced billing form handler "save" method with the following code:
```
save: function () {
    Transaction.wrap(function () {
        var cart = app.getModel('Cart').get();
        var handlePaymentResult;

        // Performs validation steps, based upon the entered billing address
        // and address options.
        // Performs payment method specific checks, such as credit card verification.
        if (!resetPaymentForms() || !validateBilling() || !handleBillingAddress(cart) ||
(handlePaymentResult = handlePaymentSelection(cart)).error) {
            returnToForm(cart);
        } else {
```

```
                if (customer.authenticated &&
app.getForm('billing').object.billingAddress.addToAddressBook.value) {

app.getModel('Profile').get(customer.profile).addAddressToAddressBook(cart.getBillingAddres
s());
                }
                // Mark step as fulfilled
                app.getForm('billing').object.fulfilled.value = true;
                if (handlePaymentResult && handlePaymentResult.redirectURL){
                   response.redirect(handlePaymentResult.redirectURL);
                   return;
                }
                // A successful billing page will jump to the next checkout step.
                app.getController('COSummary').Start();
                return;
            }
        });
    }
```

3. File: int_scalapay_controllers/cartridge/controllers/COPlaceOrder.js
   Changes: lines 68-70 within “handlePayments” method, replaced
   return {error: true};
   with the following code:

```
return {
   error: true,
   redirectURL: authorizationResult.redirectURL
};
```

   Line 159 or **after** “OrderMgr.failOrder(order);”, added the following code block:

```
if (handlePaymentsResult.redirectURL){
  return {
     error: true,
     redirectURL: handlePaymentsResult.redirectURL
  };
}
```

3.6 Custom Metadata

List of customized system object types

1. Type: **Site Preferences**
   Custom Attributes Added:
   scalaPayPaymentToken
   scalaPayEnabled
   scalaPayCountries
   scalaPayWidgetURLs
   scalaPayInstalmentsNumber
   scalaPayProductPriceColor
   scalaPayCartTextSize
   scalaPayProductPriceBox
   scalaPayProductShowLogo
   scalaPayProductLogoColor
   scalaPayCartLogoColor
   scalaPayProductCustomClass
   scalaPayProductLogoSize
   scalaPayProductTextSize
   scalaPayCartPriceColor
   scalaPayMinAmountVal

scalaPayCartShowLogo
scalaPayMaxAmountVal
scalaPayCartLogoSize
scalaPayCartCustomClass
scalaPayCartLogoAlignment
scalaPayProductPriceSelector
scalaPayCartPriceBox
scalaPayCartPriceSelector
scalaPayProductLogoAlignment
scalapayDelayCaptureEnabled
scalapayEnableSummaryPage

2. Type: **PaymentTransaction**
   Custom Attributes Added:
   a. authorizedAmount
   b. authAddressData
3. Type: **Category**
   Custom Attributes Added:
   a. scalaPayDisabled

4. INTEGRATION REFERENCES

Official ScalaPay API documentation can be found with links provided below:
WIdget API: http://www.cdn.scalapay.com/js/scalapay-widget/
Payment API: https://docs.api.scalapay.com/
Capture Delay API: https://documenter.getpostman.com/view/5023493/TVKFzwGq