



EngageOne Smart Pay Developers' Guide

**Document 100-00001-009
Published January 2023
US English Edition**

Copyright © 2018, 2023 Precisely. All rights reserved.

The information contained in the documentation and/or disk is proprietary and is subject to all relevant copyright, patent, and other laws protecting intellectual property, as well as any specific agreement protecting EngageOne Digital Self Service's rights in the aforesaid information. Neither this document nor the information contained in the documentation and/or disk may be published, reproduced, copied, modified, or disclosed to third parties, in whole or in part, without the express prior written permission. In addition, any use of this document, the documentation and/or the disk, or the information contained therein for any purposes other than those for which it was disclosed, is strictly forbidden. ALL RIGHTS NOT EXPRESSLY GRANTED ARE RESERVED.

Any representation(s) in the documentation and/or disk concerning performance of EngageOne Digital Self Service are for informational purposes only and are not warranties of product performance or otherwise, either express or implied. EngageOne Digital Self Service standard limited warranty, stated in its sales contract or order confirmation form, is the only warranty offered.

The documentation and/or disk is provided "AS IS" and may contain flaws, omissions, or typesetting errors. No warranty is granted, nor liability assumed in relation thereto, unless specifically undertaken in EngageOne Digital Self Service's sales contract or order confirmation. Information contained in the documentation and in the disk is periodically updated, and changes will be incorporated in subsequent editions. If you have encountered an error, please notify the contacts within. All specifications are subject to change without prior notice.

Portions of this document are © Copyright 2015-2023, Sorriso Technologies, Ltd.

Table of Contents

1	Introduction	4
2	Smart Pay Swagger API	4
3	Smart Pay Iframes	5
4	Smart Pay Callback CSS	5
5	Smart Pay Callback HTML	6
6	Customizing the Date Format	8
6.1	Changing the Date Format with the Development Kit	8
6.2	Changing the Date Format with the I18N files.....	9

1 Introduction

EngageOne Smart Pay is an advanced online payment services solution (for both small and large businesses) that allows you to access account information and make payments. Reliable, convenient, and easily customizable, this solution provides an intuitive digital experience.

The *EngageOne Smart Pay Developers' Guide* provides technical insight into the accessible areas of the Smart Pay server integration.

EngageOne Smart Pay consists of several primary components:

- Customer Portal
 - The online location where customers can pay bills, establish one-time and automatic payments, and view past payments. These integrations are performed with REST calls to the Payment Server.
- Payment Server
 - The payment server handles the REST calls defined in the Smart Pay Swagger API. It interacts with the Payment Encryption server to tokenize and detokenize payment source information and submits authentication and payments to a merchant gateway through a payment driver interface.
- Payment Encryption Server
 - The encryption server handles encryption and decryption of sensitive customer data, as defined by PCI specifications for handling secure data. Sensitive information captured in the add/edit source iframes are immediately sent for authentication to the merchant gateway, and then encrypted and tokenized by the server. This allows the Customer Portal to reference the payment source with the token, thus ensuring the portal does not have access to sensitive unencrypted data at any time. On making a payment, the token is decrypted, and the data is sent to the merchant gateway.

2 Smart Pay Swagger API

The Smart Pay API consists of a number of REST calls that are documented as Swagger YAML definition files. These are the REST calls used by the Smart Pay module to perform payment functions and are provided for informational purposes. Please refer to the latest **smartpay.yaml** and **smartpay_tokenization.yaml**.

3 Smart Pay Iframes

Configure credit card
Please fill in the details as printed on the card

Credit card number * ? Card type *

CVV number * ? Expiration date *
06 - Jun 2018

Name on card * Country *
United States

Billing address * State/province *
Alabama

address field 2 (optional) Postal code *

City *

Credit card nickname ?

Make default payment method

SAVE CANCEL

Figure 1 - Add Credit Card Iframe

In the Customer Portal, the forms for capturing sensitive data (add payment source and edit payment source) are embedded in the screen as iframes, served from the payment server URL. This means one cannot directly edit or override the iframe content on screen. This design is due to PCI Compliance security requirements, and the only way to customize the Smart Pay server interface is through the callback URLs set during the Smart Pay installation.

To allow for customizing the forms, the **callback CSS URL** is used to load the file as native CSS within the iframe context. To customize the application interaction with the server responses, the **callback HTML URL** is used, which the iframe redirects to upon success or error.

4 Smart Pay Callback CSS

Most users of Smart Pay will wish to override the CSS styling of the add source and edit source iframes. The default file provided is **paymentCustom.css**. This file can be edited to override the iframe CSS for

both add and edit source. If another file was used in the URL setup during installation, edit the corresponding file and override the iframe styling.

5 Smart Pay Callback HTML

Due to PCI Compliance, the only accessible code of the iframe is the callback HTML defined via a URL. The payment server redirects to this HTML file upon iframe success or error, and this file can be used to catch the response. The default file provided is `paymentCallback.html`

```

1 <html>
2   <head>
3     <title>Sorriso Payment Callback</title>
4     <script>
5       function getQueryParam(paramName) {
6         var queryString = window.location.search.substring(1);
7         var queryStringMap = queryString.split("&");
8         for (i = 0; i < queryStringMap.length; i++) {
9           var paramMap = queryStringMap[i].split("=");
10          if (paramMap[0] == paramName) {
11            return decodeURIComponent(paramMap[1]);
12          }
13        }
14
15        return null;
16      }
17
18      // get query string params and form into JSON object
19      var jsonData = {
20        "responseCode": getQueryParam("responseCode"),
21        "responseMessage": getQueryParam("responseMessage"),
22        "token": getQueryParam("o"),
23        "transactionId": getQueryParam("transactionId")
24      };
25
26      function loadCallback() {
27        // callback handlers
28        if (jsonData.responseCode == "-1") {
29          window.top.handleCancelResponseCallback(jsonData);
30        } else if (jsonData.responseCode == "10") {
31          window.top.handleAddSourceSuccessResponseCallback(jsonData);
32        } else if (jsonData.responseCode == "11") {
33          window.top.handleAddSourceErrorResponseCallback(jsonData);
34        } else if (jsonData.responseCode == "20") {
35          window.top.handleEditSourceSuccessResponseCallback(jsonData);
36        } else if (jsonData.responseCode == "21") {
37          window.top.handleEditSourceErrorResponseCallback(jsonData);
38        } else {
39          window.top.handleErrorResponseCallback(jsonData);
40        }
41      }
42    </script>
43  </head>
44  <body onLoad="loadCallback();">
45  </body>
46 </html>

```

Figure 2 - paymentCallback.html

The payment server will redirect to this callback file with the query string parameters of “responseCode”, “responseMessage”, “o” (payment source token), and “transactionId”. As shown in the sample HTML in Figure 2, the query parameters are extracted, and onLoad of the <body> tag, a JavaScript method is called to handle the response. This method calls individual callback methods for each response type, using “window.top” to refer to the customer portal scope. These methods perform whatever logic is desired, such as showing success and error messages on screen. You can view or

modify these functions in the **payment.js** found the application's linked-root, or add custom functions via a new file included in the settings.xml header.

6 Customizing the Date Format

The application has two formats for displaying dates. The first is "numeric" which is used to display date in a compact form using just numbers. The second is "text" this displays the date in a longer form where the month is text rather than numeric.

6.1 Changing the Date Format with the Development Kit

If you have the Development Kit, the date formats can be customized for each application by modifying their "validation.i18n" file. To modify the numeric format of the date you need to update the following two values:

- dateNumeric - Defines how to format the date for display.
- dateValidation - Defines the human readable text to display when the user is providing a date and doesn't use the correct format.

When defining the date format, you can provide any characters mixed with three keys:

- %m - Identifies the month which will be displayed as a number between 1 and 12.
- %d - Identifies the day of the month displayed as a number between 1 and 31.
- %y - Identifies the year displayed as 4 digits.

The default format is "%m/%d/%y" which means December 25th, 2021, would be displayed as "12/25/2021". Some other possible formats you can use would be:

- "%d-%m-%y" which would display the date as "25-12-2021".
- "%y.%m.%d" which would display the date as "2021-12-25".

To change how "text" format of the date you need to update the following values:

- dateText - How to format the date.
- dateMonth1 - The text for the month January.
- dateMonth2 - The text for the month February.
- dateMonth3 - The text for the month March.
- dateMonth4 - The text for the month April.
- dateMonth5 - The text for the month May.
- dateMonth6 - The text for the month June.
- dateMonth7 - The text for the month July.
- dateMonth8 - The text for month August.
- dateMonth9 - The text for month September.
- dateMonth10 - The text for month October.
- dateMonth11 - The text for month November.
- dateMonth12 - The text for month December.

Much like dateNumeric format you can provide text for the format along with the keys:

- %m - Identifies the month which will be displayed as defined in the dateMonth values.
- %d - Identifies the day of the month displayed as a number between 1 and 31.
- %y - Identifies the year displayed as 4 digits.

The default format is “%m %d %y” which means December 25th, 2021, would be displayed as “Dec 25 2021”. Some other possible formats you can use would be:

- “%d-%m-%y” which would display the date as “25-Dec-2021”.
- %y.%m.%d” which would display the date as “2021-Dec-25”.

If you do not want the month abbreviated, you just need to provide the full month text in the dateMonth values.

6.2 Changing the Date Format with the I18N files

If you are just customizing the dates via the I18N files you need to edit the “validation_en.properties”, replacing “en” with the language code for which ever language you want to customize. To modify the numeric format of the date you need to update the following two values:

- validation_dateNumeric - Defines how to format the date for display.
- validation_dateValidation - Defines the human readable text to display when the user is providing a date and doesn’t use the correct format.

When defining the date format, you can provide any characters mixed with three keys:

- %m - Identifies the month which will be displayed as a number between 1 and 12.
- %d - Identifies the day of the month displayed as a number between 1 and 31.
- %y - Identifies the year displayed as 4 digits.

The default format is “%m/%d/%y” which means December 25th, 2021, would be displayed as “12/25/2021”. Some other possible formats you can use would be:

- “%d-%m-%y” which would display the date as “25-12-2021”.
- %y.%m.%d” which would display the date as “2021-12-25”.

To change how “text” format of the date you need to update the following values:

- validation_dateText - How to format the date.
- validation_dateMonth1 - The text for the month January.
- validation_dateMonth2 - The text for the month February.
- validation_dateMonth3 - The text for the month March.
- validation_dateMonth4 - The text for the month April.
- validation_dateMonth5 - The text for the month May.
- validation_dateMonth6 - The text for the month June.
- validation_dateMonth7 - The text for the month July.
- validation_dateMonth8 - The text for month August.

- validation_dateMonth9 - The text for month September.
- validation_dateMonth10 - The text for month October.
- validation_dateMonth11 - The text for month November.
- validation_dateMonth12 - The text for month December.

Much like validation_dateNumeric format you can provide text for the format along with the keys:

- %m - Identifies the month which will be displayed as text as defined in the validation_dateMonth values.
- %d - Identifies the day of the month displayed as a number between 1 and 31.
- %y - Identifies the year displayed as 4 digits.

The default format is “%m %d %y” which means December 25th, 2021, would be displayed as “Dec 25 2021”. Some other possible formats you can use would be:

- “%d-%m-%y” which would display the date as “25-Dec-2021”.
- %y.%m.%d” which would display the date as “2021-Dec-25”.

If you do not want the month abbreviated, you just need to provide the full month text in the validation_dateMonth values.