

Public Cloud Custom Form Design

30 July 2024 | Document Version 0.01

Berk Saldır

Contents

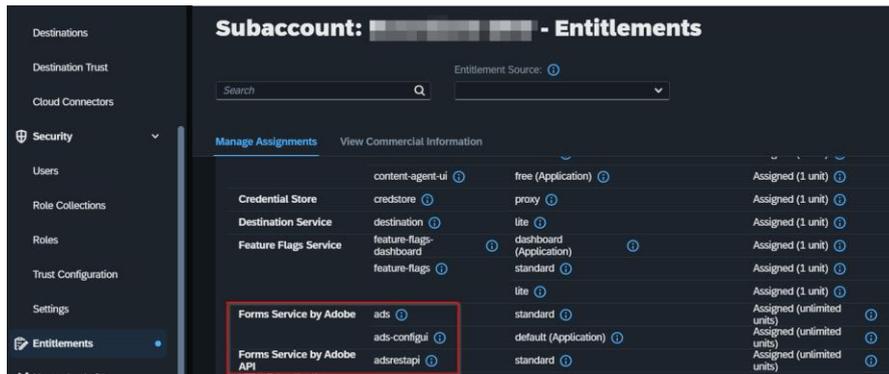
SAP Forms Service by Adobe.....	3
1. Service Adjustments on the BTP Account.....	3
2. Custom Form Creation and Upload.....	5
3. Forms Service by Adobe API Configurations in the System.....	8
4. Call Forms Service by Adobe from ABAP Code.....	12
5. Send Rendered PDF to Print Queue.....	14

SAP Forms Service by Adobe

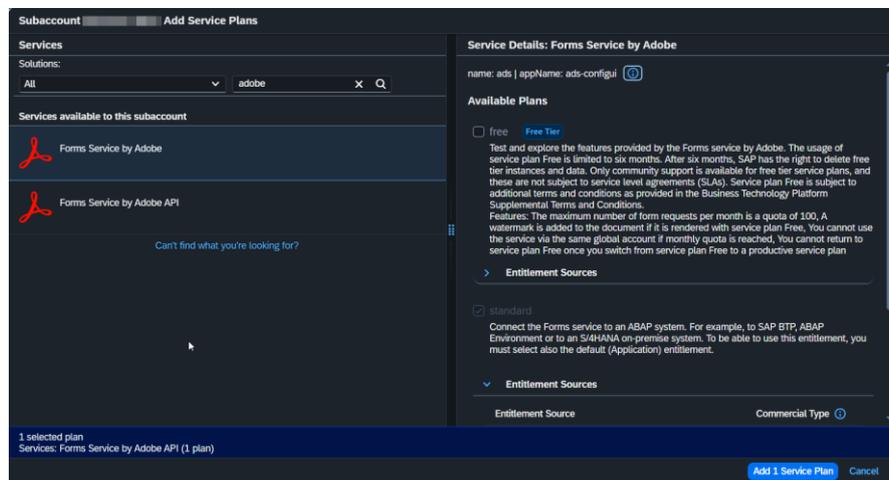
SAP Forms service by Adobe lets you generate print and interactive forms using Adobe Document Services (ADS). Calling the service from cloud application using a REST API for rendering documents and for managing form templates in the template store.

1. Service Adjustments on the BTP Account

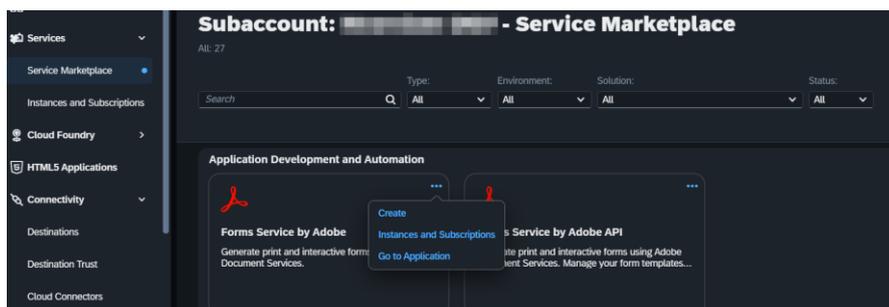
Firstly, it should be checked whether there are services in Entitlements part of subaccount.



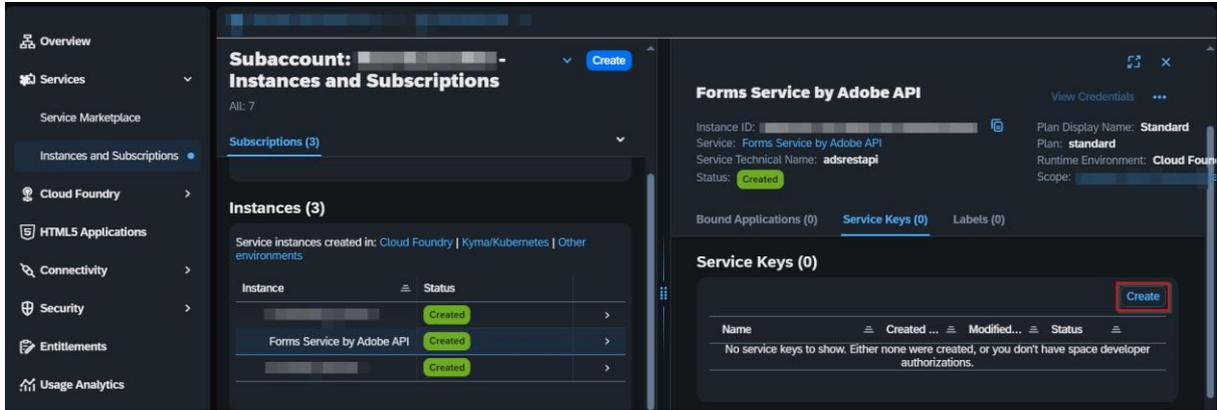
If services are not available, they should be added. To add them, edit mode should be opened and clicked "Add Service Plans" button. The services listed on the pop-up screen must be added.



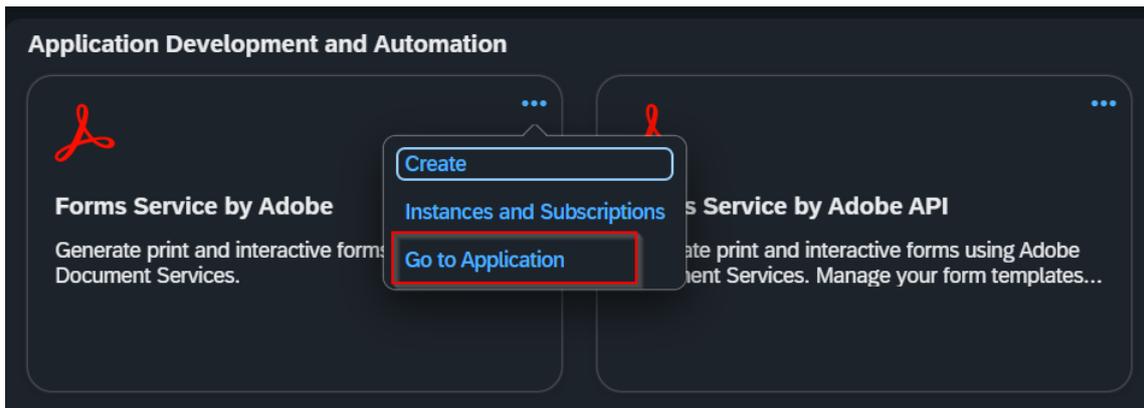
Default plan should be selected for "Forms Service by Adobe" service. Standard plan should be selected for "Forms Service by Adobe API" service. After that part, some adjustments will be made in the Services part of subaccount. Both services need to be created with selected plans.



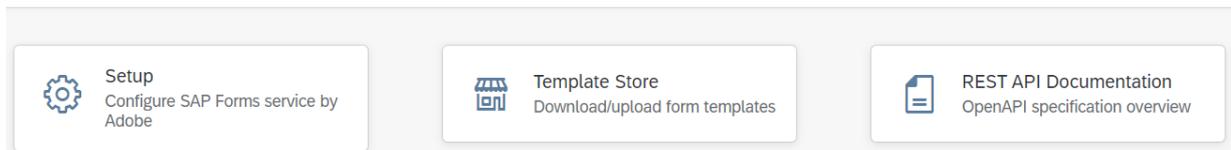
When creating “Forms Service by Adobe API”, “Runtime Environment” will be asked. “Cloud Foundry” should be selected. If “Cloud Foundry” is not enabled, it should be enabled by contacting an authorized person. After the “Forms Service by Adobe API” is created, a service key is generating from “Instances and Subscriptions”.



After completing adjustments, an app and a service key is created. The app is used to store form templates and to make configurations of SAP forms service by Adobe. The service key is used for creating custom forms from ABAP codes. The app can be opened as follows.

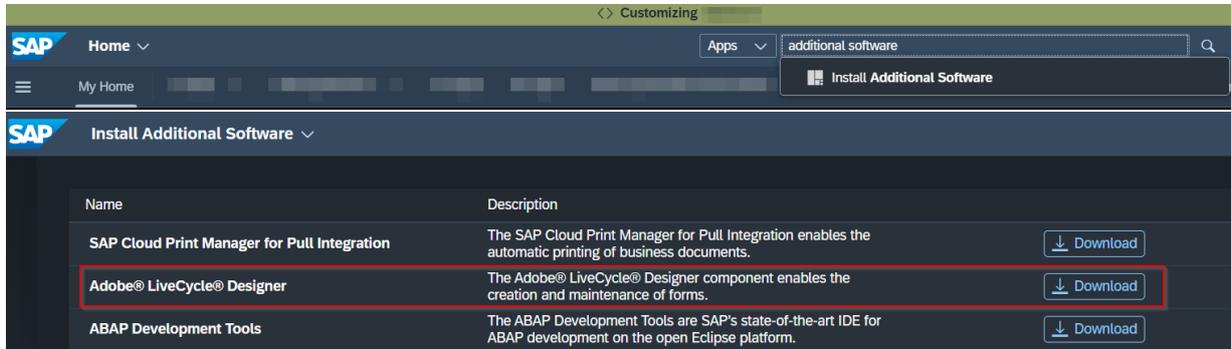


SAP Forms Service by Adobe Overview

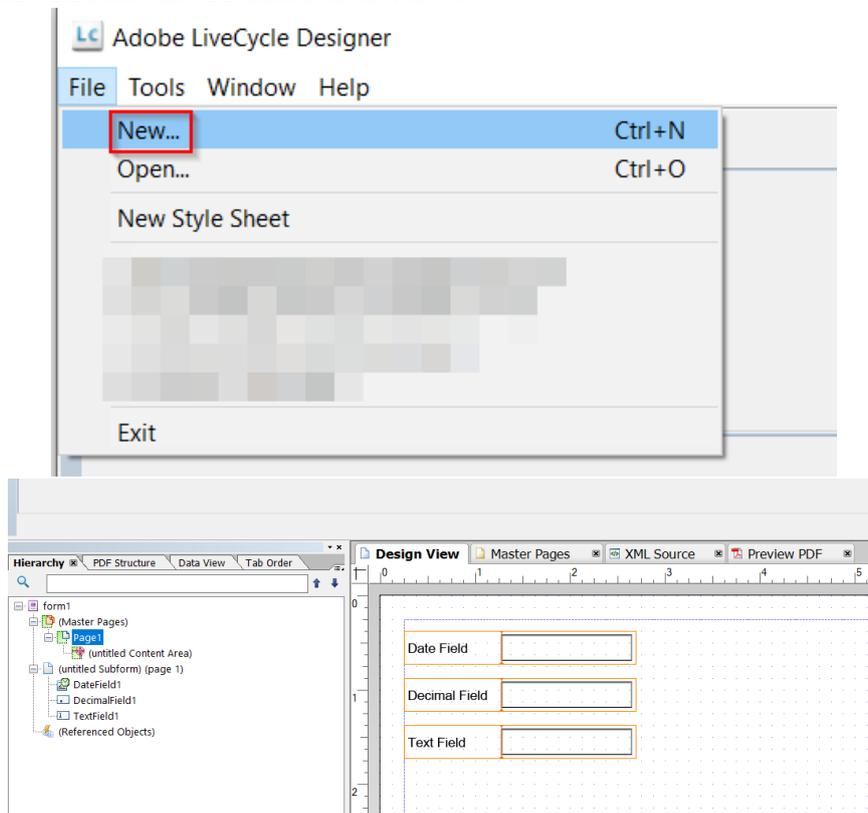


2. Custom Form Creation and Upload

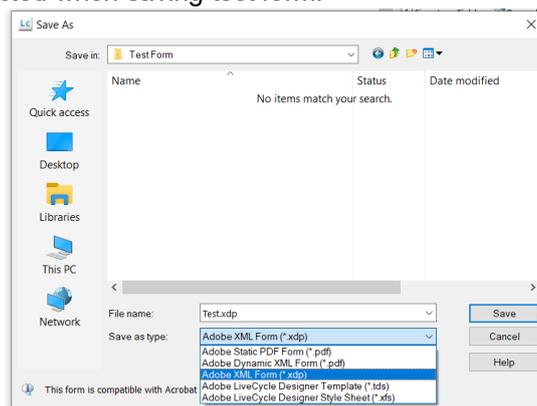
“Adobe LiveCycle Designer” is used to create custom forms. If the designer isn’t installed, it should be installed first. The installation file can be downloaded from the cloud system as follows.



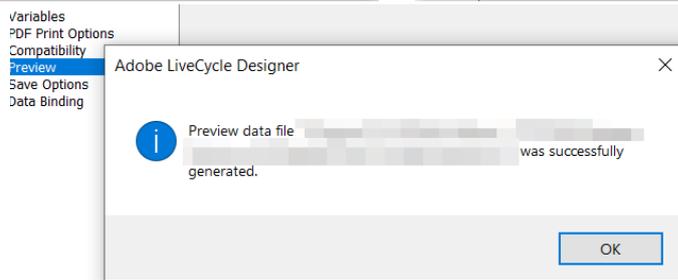
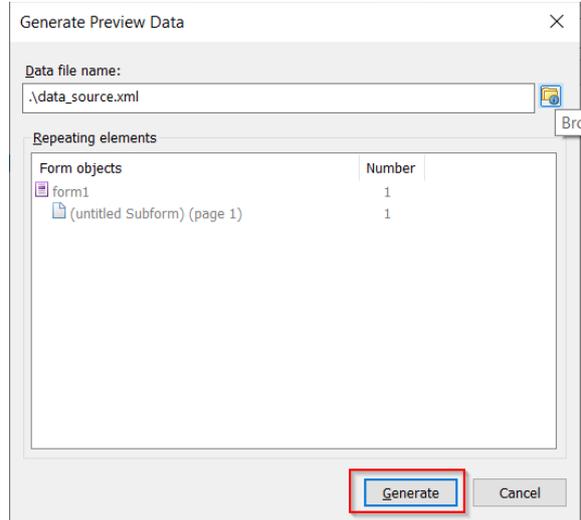
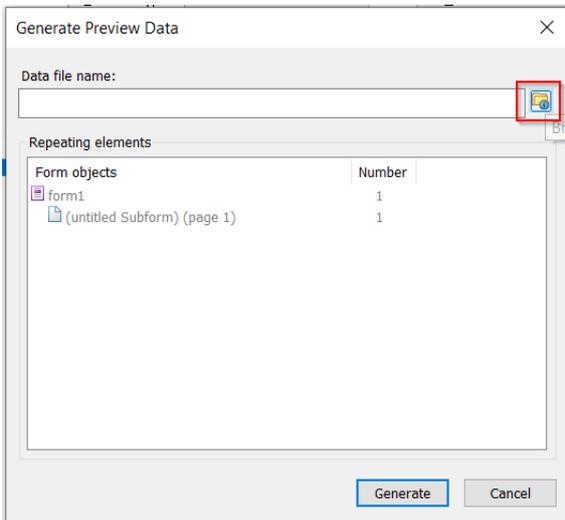
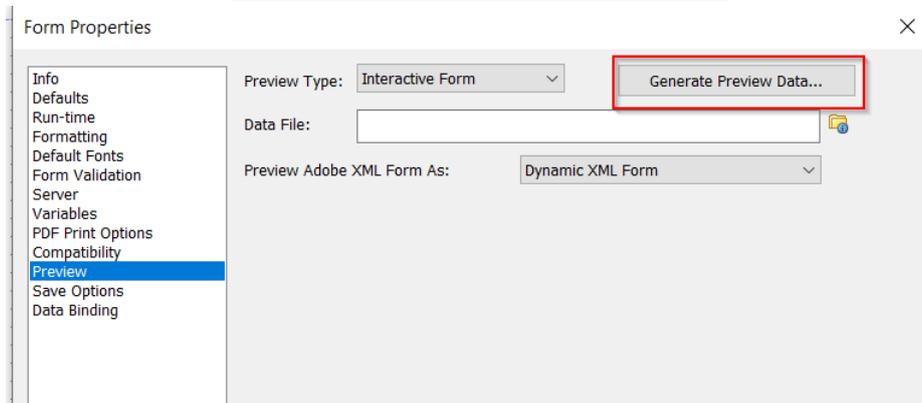
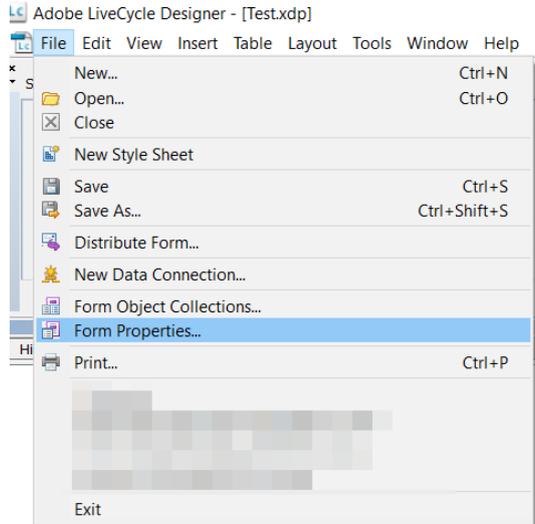
After installation test form can be created for uploading template store. Some fields which are text fields, decimal fields, date fields etc. can be added to test form.

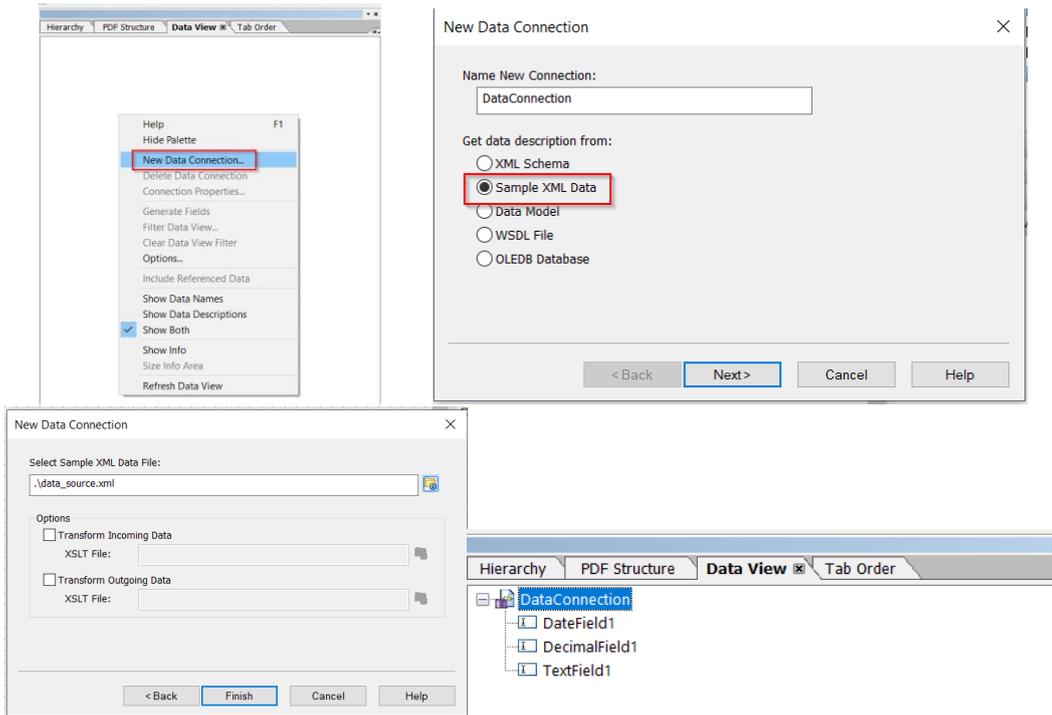


Below selection should be selected when saving test form.

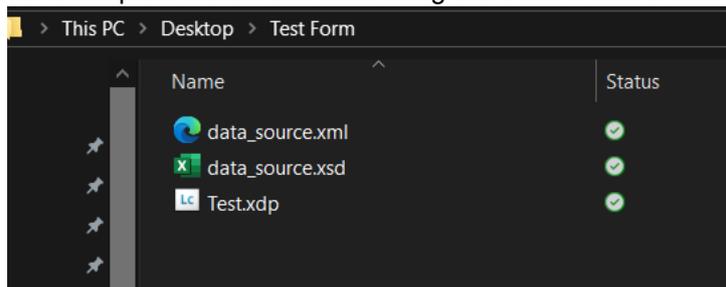


There is a “Data View” tab in left side of designer. Binding values are seen there. Following steps should be followed first for them to be seen.

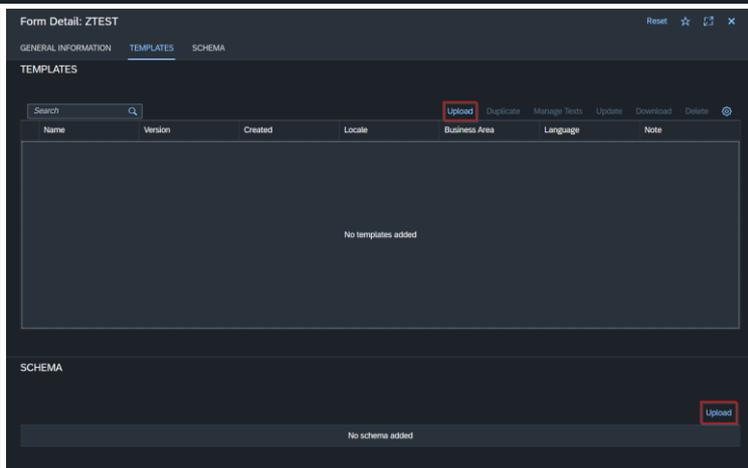
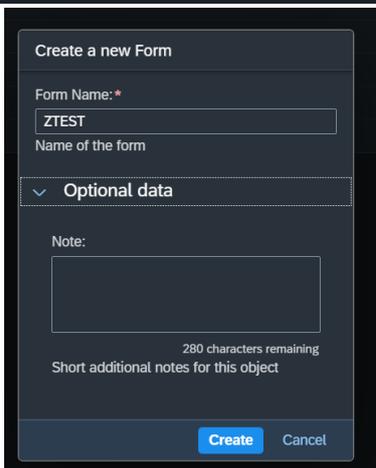
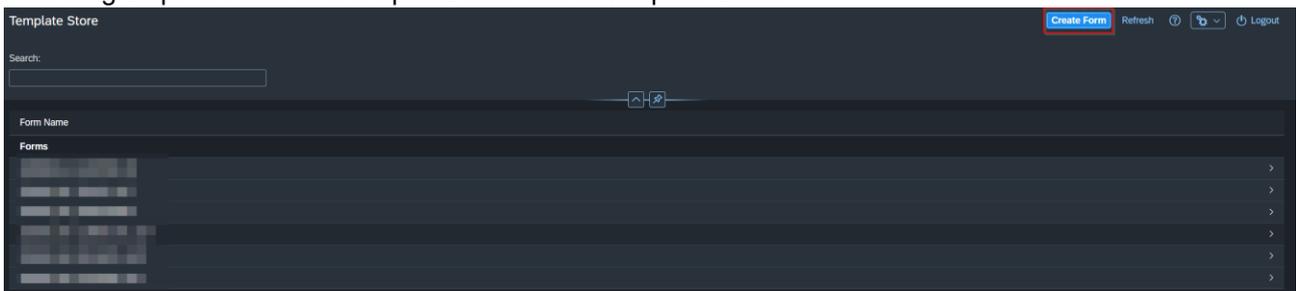




Now, binding values can be used for fields added to the form. XML file should be converted to XSD file for uploading binding schema to template store. After converting there are 3 files in test form folder.

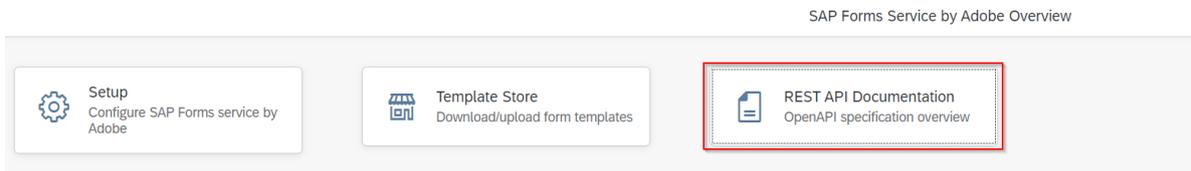


Following steps are followed to upload test form to template store.



3. Forms Service by Adobe API Configurations in the System

API document is opened using the app.

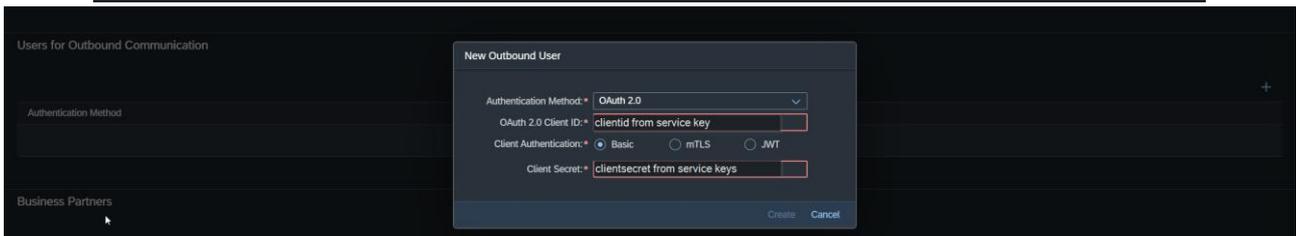
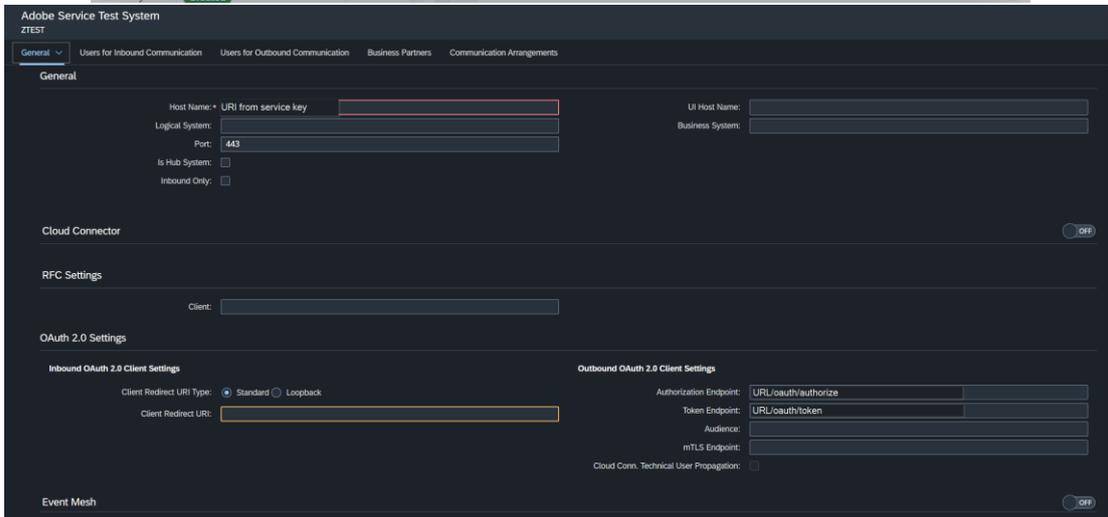
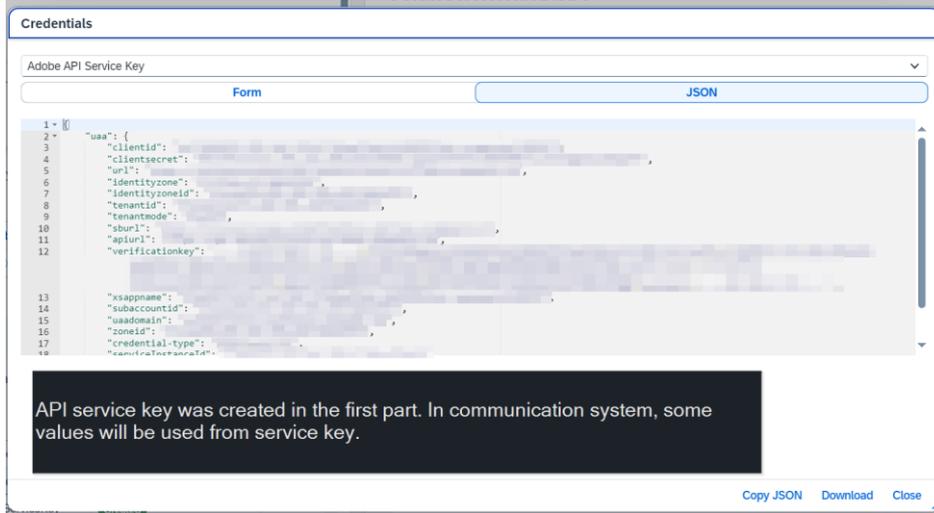
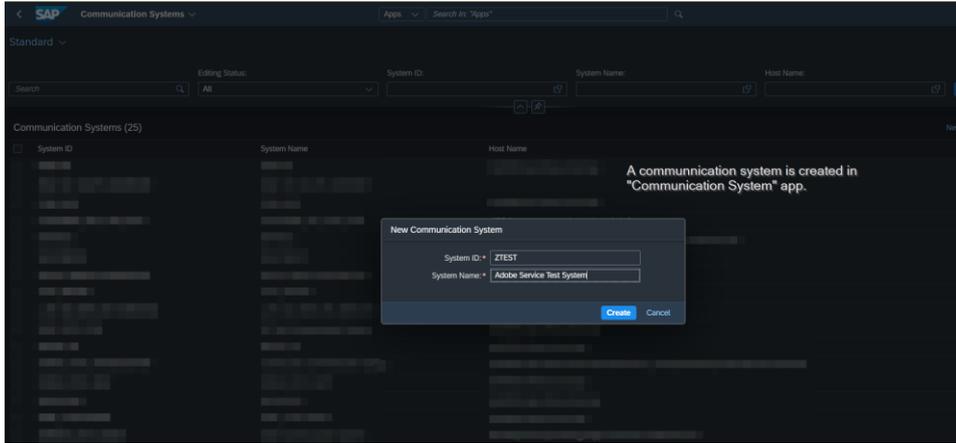


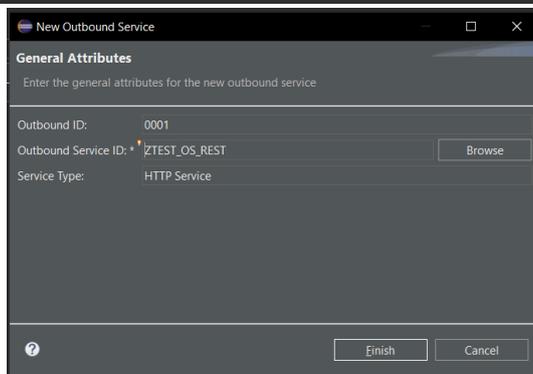
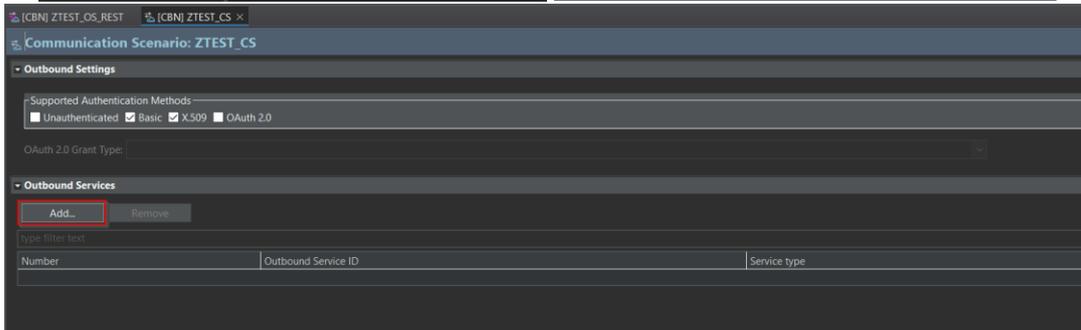
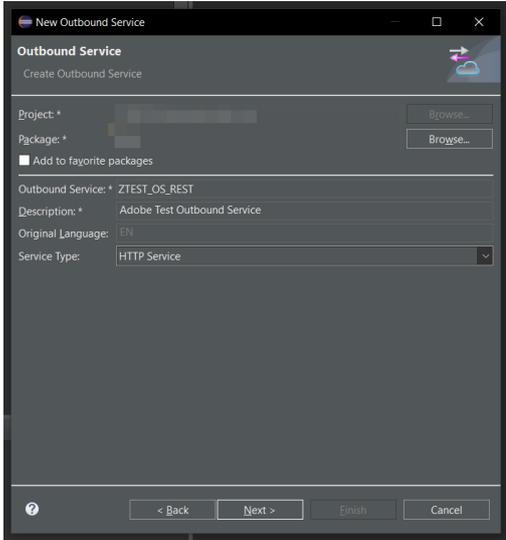
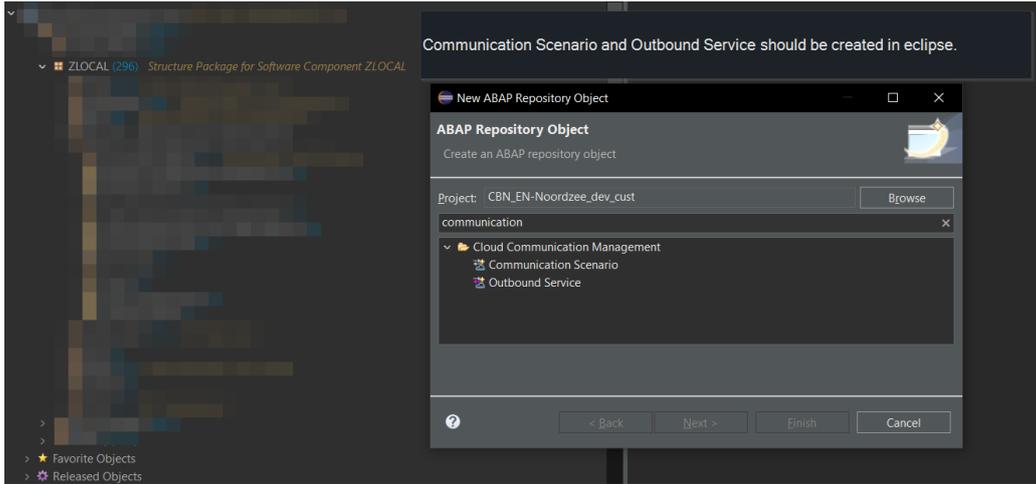
“ADS – Render Request” method is used to render pdf. Method responses can be examined on this page. Inputs of method is as follows.

```
{
  "xdpTemplate": "Base64 String",
  "xmlData": "Base64 String",
  "formType": "print",
  "formLocale": "en_US",
  "taggedPDF": true,
  "embedFont": true,
  "changeNotAllowed": false,
  "printNotAllowed": false
}
```

Parameters can be filled with fixed values in the service link. There are 8 input values in the request body. “xdpTemplate” is used for template name. “xmlData” is used to transfer bind values. Type of this field is base64. Bind values should be created in XML format. Then it should be converted to Base64 format in ABAP code. Transformation object can be used to convert a structure to XML. Other input values can be filled in as in the example value.

To use the service, the following steps must be made.





The image displays a sequence of four screenshots from the SAP system, illustrating the configuration and management of a communication scenario.

- Top Screenshot:** Shows the configuration for a Communication Scenario named 'ZTEST_CS'. Under 'Outbound Settings', 'Supported Authentication Methods' includes 'OAuth 2.0'. The 'Outbound Services' section lists 'Adobe Test Outbound Service' with ID '0001'. The 'REST Service Settings' are configured for 'HTTP 1.0' and 'Compress Response' is set to 'Inactive'.
- Second Screenshot:** Shows the 'General' properties of the scenario, including 'Communication Scenario Type' (Customer Managed) and 'Status' (Unpublished).
- Third Screenshot:** Shows the 'Communication Arrangements' list. A 'New Communication Arrangement' dialog box is open, with 'Scenario' set to 'ZTEST_CS' and 'Arrangement Name' set to 'ZTEST_CARR'.
- Bottom Screenshot:** Shows the detailed configuration for the 'ZTEST_CARR' arrangement. It includes 'Common Data' (Arrangement Name: ZTEST_CARR, Communication System: ZTEST), 'Outbound Communication' (OAuth 2.0 Client ID, Authentication Method: OAuth 2.0), and 'Outbound Services' (Adobe Test Outbound Service, Service Status: Active, Port: 443). A message states: 'There are 2 parameters after the method in path. The purpose of the fixed values can be checked in the document.' A 'Check Connection' button is visible.

4. Call Forms Service by Adobe from ABAP Code

Objects should be created before writing code.

```

ZTEST_S_BINDING x
1 @EndUserText.label : 'Binding Structure'
2 @AbapCatalog.enhancement.category : #NOT_EXTENSIBLE
3 define structure ztest_s_binding {
4
5     datefield1      : abap.dats;
6     decimalfield1  : abap.dec(8,5);
7     textfield1     : abap.char(50);
8
9 }

```

Structure of data binding is created. That can be changed in every form. It is used in transformation.

```

1 @EndUserText.label : 'Request Body'
2 @AbapCatalog.enhancement.category : #NOT_EXTENSIBLE
3 define structure ztest_r_binding {
4
5     xdp_template    : abap.string(0);
6     xml_data        : abap.string(0);
7     form_type       : abap.string(0);
8     form_locale     : abap.string(0);
9     tagged_pdf      : abap.string(0);
10    embed_font      : abap.string(0);
11    change_not_allowed : abap_boolean;
12    print_not_allowed : abap_boolean;
13
14 }

```

This structure will be used for the request body of the method to be used.

```

1 @EndUserText.label : 'Response Body'
2 @AbapCatalog.enhancement.category : #NOT_EXTENSIBLE
3 define structure ztest_r_binding {
4
5     filecontent     : abap.string(0);
6
7 }

```

Filecontent field is in response body which is returned in response code 200. That structure is used to convert response json body in deserialize method.

```

(CBN) ZTEST_TR_BINDING x
1 <?xml version="1.0"?>
2 <!-- Transformation -->
3 <!-- xmlns:tt="http://www.sap.com/transformation-templates" xmlns:dic="http://www.sap.com/abapxml/types/dictionary" xmlns:df="http://www.sap.com/abapxml/types/defined" -->
4 <tt:root xmlns="ROOT"/>
5 <tt:template>
6 <form>
7 <datefield tt:value-ref="Form.DateField1"/>
8 <decimalfield tt:value-ref="Form.DecimalField1"/>
9 <textfield tt:value-ref="Form.TextField1"/>
10 </form>
11 </tt:template>
12 </tt:transform>
13
14 </tt:transform>

```

This transformation object is used to convert a binding structure to XML. Content of the transformation object can be changed other forms.

ABAP codes to render PDFs are as follows.

```

TRY.
  DATA(lo_dest) = cl_http_destination_provider=>create_by_comm_arrangement(
    comm_scenario = 'ZTEST_CS'
    comm_system_id = 'ZTEST'
    service_id    = 'ZTEST_OS'
  ).
  CATCH cx_http_dest_provider_error INTO DATA(lx_error).
ENDTRY.

TRY.
  DATA(lo_client) = cl_web_http_client_manager=>create_by_http_destination( lo_dest ).
  CATCH cx_web_http_client_error INTO DATA(lx_client_error).
ENDTRY.

DATA(lo_request) = lo_client->get_http_request( ).
lo_request->set_header_fields( VALUE #(
  ( name = 'Accept' value = 'application/json, text/plain, */*' )
  ( name = 'Content-Type' value = 'application/json;charset=utf-8' )
) ).

```

HTTP objects should be created. Custom comm. scenario, custom outbound service and custom comm. system are used when creating destination object.

```

ls_data = VALUE #( datefield1 = '20240730'
                   decimalfield1 = '55.99'
                   textfield1 = 'Test Form' ).

TRY.
  CALL TRANSFORMATION ztest_tr_binding
  SOURCE form = ls_data
  RESULT XML DATA(lv_xml).

  CATCH cx_root INTO DATA(lo_root).
ENDTRY.

DATA(lv_base64_data) = cl_web_http_utility=>encode_x_base64( unencoded = lv_xml ).

ls_req_xdp_template = 'ZTEST/Test'.
ls_req_xml_data = lv_base64_data.
ls_req_form_type = 'print'.
ls_req_form_locale = 'tr_TR'.
ls_req_tagged_pdf = 1.
ls_req_embed_font = 0.
ls_req_change_not_allowed = abap_false.
ls_req_print_not_allowed = abap_false.

TRY.
  CALL METHOD /ui2/cl_json=>serialize
  EXPORTING
    data = ls_req
    pretty_name = /ui2/cl_json=>pretty_mode_camel_case
  RECEIVING
    r_json = DATA(lv_body).

  CATCH cx_root INTO DATA(lx_root).
ENDTRY.

lo_request->set_text(
  EXPORTING
    i_text = lv_body
).

```

Creating test data

Test data should first be converted to XML using transformation object. That isn't enough for transferring it to request body. It should be converted to BASE64 type after XML transformation.

Template and form name format

These are fixed values. If needed, they can be changed by examining the API document.

Serialize from structure to request json body

Set request json body to object

```

TRY.
  DATA(lo_response) = lo_client->execute(
    i_method = if_web_http_client=>post
    i_timeout = 4
  ).
  CATCH cx_web_http_client_error INTO lx_client_error.

  DATA(lv_response) = lo_response->get_text( ).
  DATA(lx_status) = lo_response->get_status( ).

TRY.
  CALL METHOD /ui2/cl_json=>deserialize
  EXPORTING
    json = lv_response
    assoc_arrays = abap_true
    name_mappings = VALUE #( ( json = 'fileContent' abap = 'FILECONTENT' ) )
  RECEIVING
    data = ls_response.

  CATCH cx_root INTO lx_root.
ENDTRY.

out->write( ls_response->filecontent ).

```

Execute the request. Response value can be seen on the right side. If API doesn't work properly, it can be checked from "get_text" and "get_status" methods in response object.

Filecontent field can be extracted from response body with using deserialize method. Then BASE64 value can be used as desired. This example was tested using a base64 to pdf converter site.

Base64*

JVBERi0x1jYhJel_jz9MNCjI2IDAgb2JqCjw8L0ZpbHR1c19GbgF0ZURlY29kZS9GaXJzdCA4NC9hZm5ndGggMjg4L04gMThVHVh1wZS9PYm9TdG0+PnN0cmVhb0k0aH58kUTrg0AUhf/KWiei3P
 PyASEQCIISt16k1LED1KYGGRSmn/f66iELnR1Yc4357uJQIEIDVlcCmRgEcEaADB3T1HUHQUBJF7MaSgoSA1YRoqcYekJ01qIguXIA8w7HT1QaIGlPnz1rLQZj0Mh/kw/za0/mAc6yvjPv
 1Q1CqR43aL7XCVHZf1vd3vXns8nN1b1w7vBTZ1/ktO/gSRe1CnB37+ngh/pmf1FyOckbwupvHOD+BjX0kkrH97nhqj7pUjxfyk1tFSRPjf1Y2PG2Pw45dMhY7v7bgtTKdt08rcK1mAs06G
 WReCFgky3puzd3tSZur1U3dFovk065ZHRbacjyRhwGQv6U+dhvQNZ1a5/8EGAD9+pmScmVuzH0cmVhb0p1bmrVymokHjcgKByvmokP0vvh1sdGvYl0ZsYXRIRGvjB2RILl0Zpcn0IDEX
 L0x1bms0cA1Hy90DI1Vh1wZS9PYm9TdG0+PnN0cmVhb0k0aH58kUTrg0AUhf/KWiei3PPyASEQCIISt16k1LED1KYGGRSmn/f66iELnR1Yc4357uJQIEIDVlcCmRgEcEaADB3T1HUHQUBJF7MaSgoSA1YRoqcYekJ01qIguXIA8w7HT1QaIGlPnz1rLQZj0Mh/kw/za0/mAc6yvjPv
 oK1jggKCBYvmokP0vvh1sdGvYl0ZsYXRIRGvjB2RILl0Zpcn0IDEXL0x1bms0cA1Hy90DI1Vh1wZS9PYm9TdG0+PnN0cmVhb0k0aH58kUTrg0AUhf/KWiei3PPyASEQCIISt16k1LED1KYGGRSmn/f66iELnR1Yc4357uJQIEIDVlcCmRgEcEaADB3T1HUHQUBJF7MaSgoSA1YRoqcYekJ01qIguXIA8w7HT1QaIGlPnz1rLQZj0Mh/kw/za0/mAc6yvjPv

Decode Base64 to PDF

Preview PDF

42J8VjN7KwB3PTwXlPiQxXpFGBi3GwhiCwWu... 1 / 1 - 95% +

Date Field: 30.Tem.2024

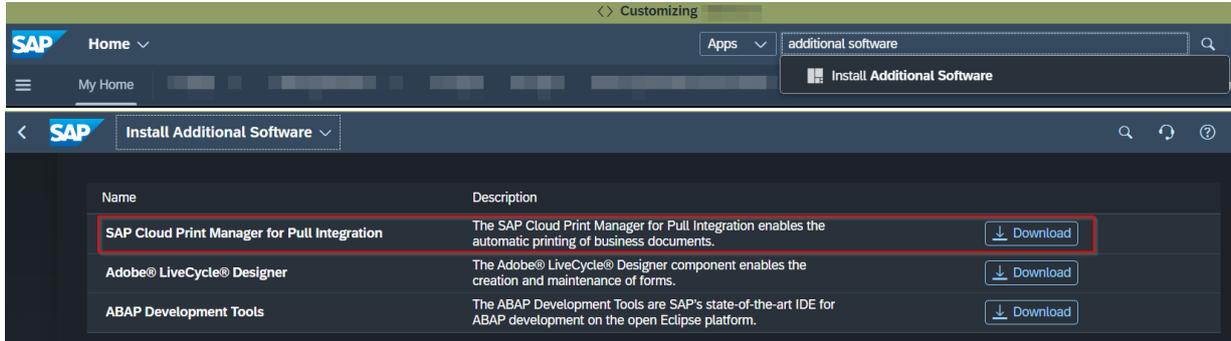
Decimal Field: 55.99

Text Field: Test Form

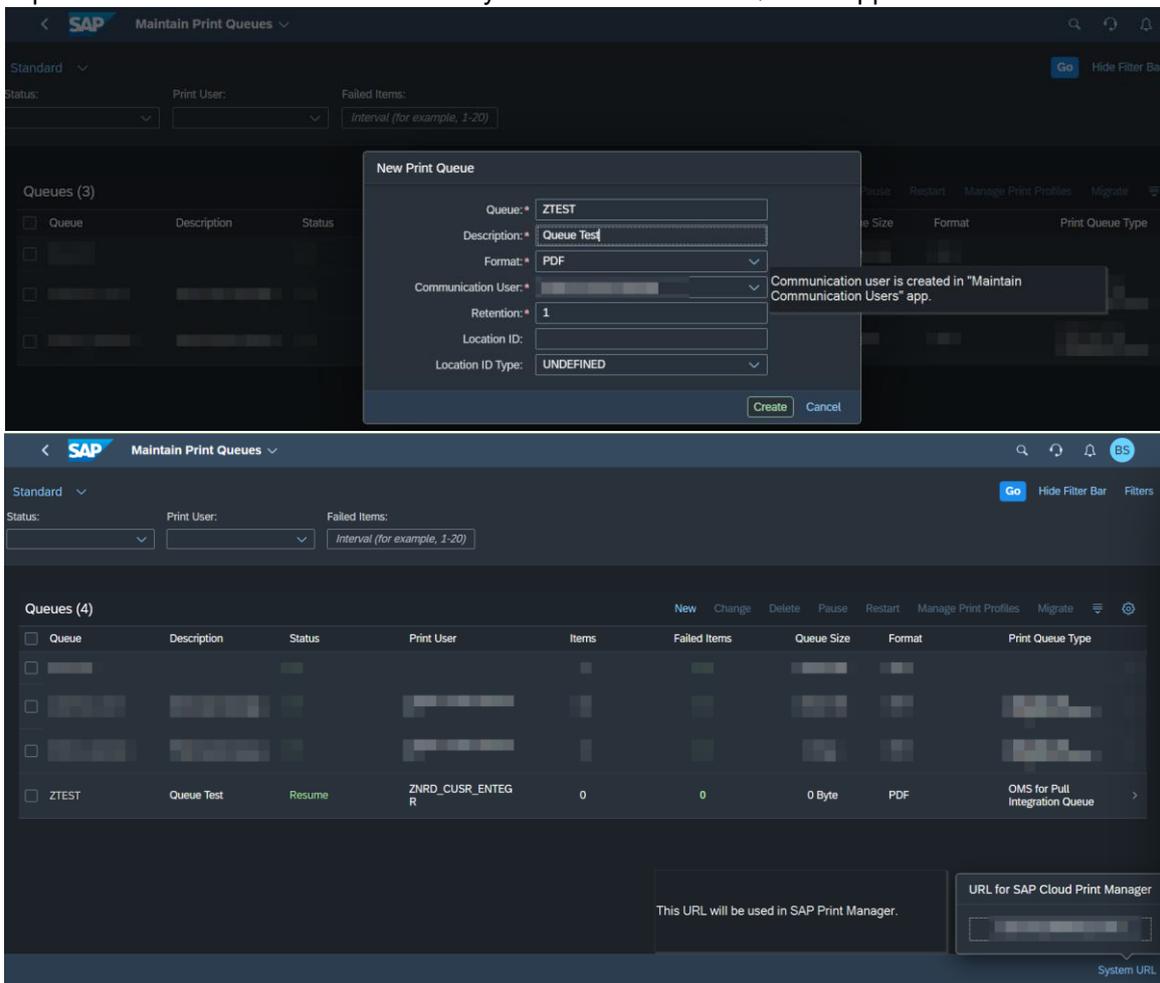
Data transferred. Bindings can be seen in the test form.

5. Send Rendered PDF to Print Queue

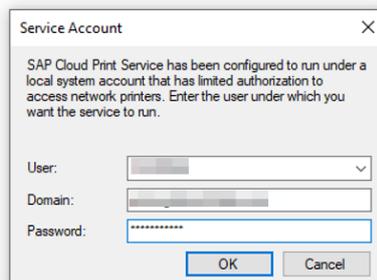
“SAP Cloud Print Manager for Pull Integration” is used to send documents to printer. If the print manager isn’t installed, it should be installed first. The installation file can be downloaded from the cloud system as follows.



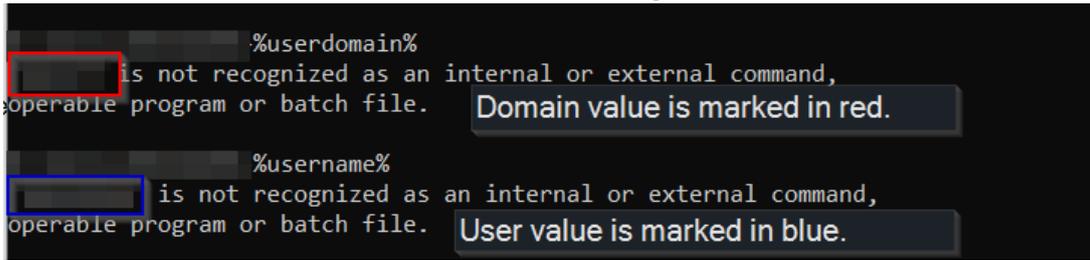
A print queue should be created in the cloud system. Maintain Print Queues app is used for that.



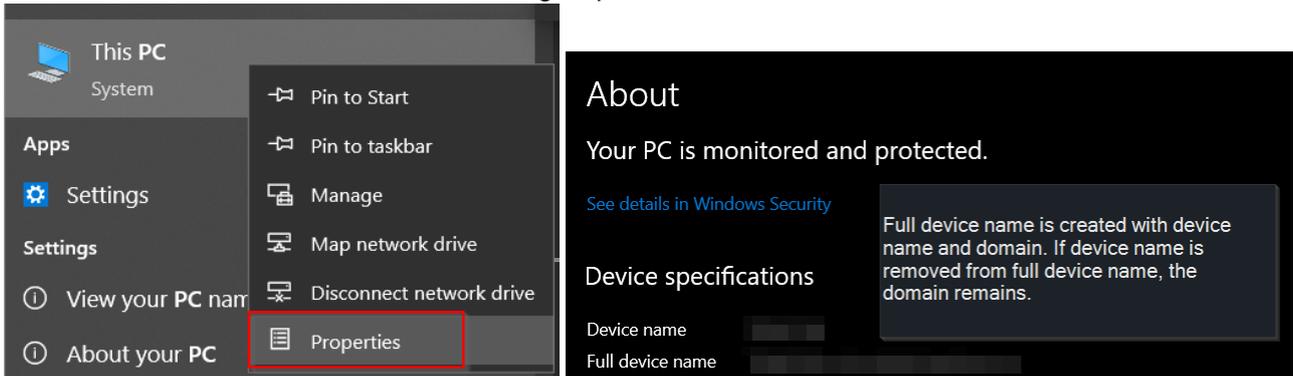
When SAP Print Manager is opened, the following screen appears.



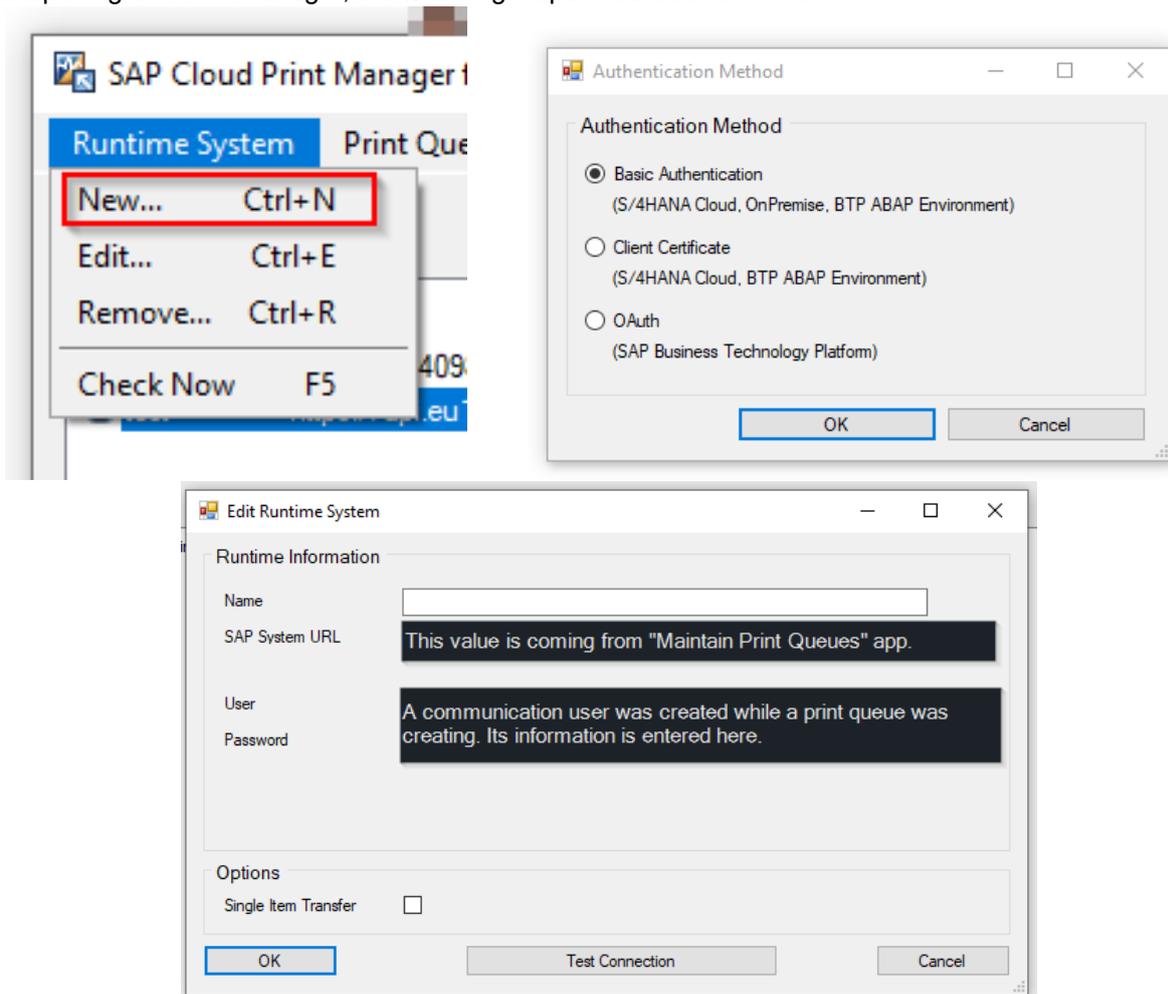
User and domain values can be found in CMD with the following commands.

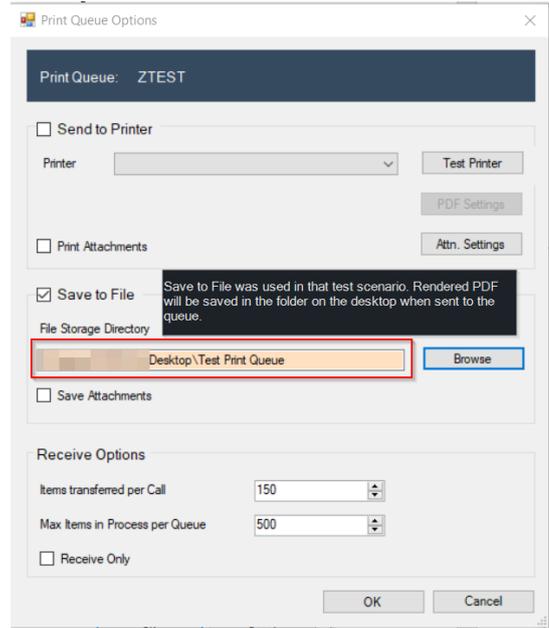
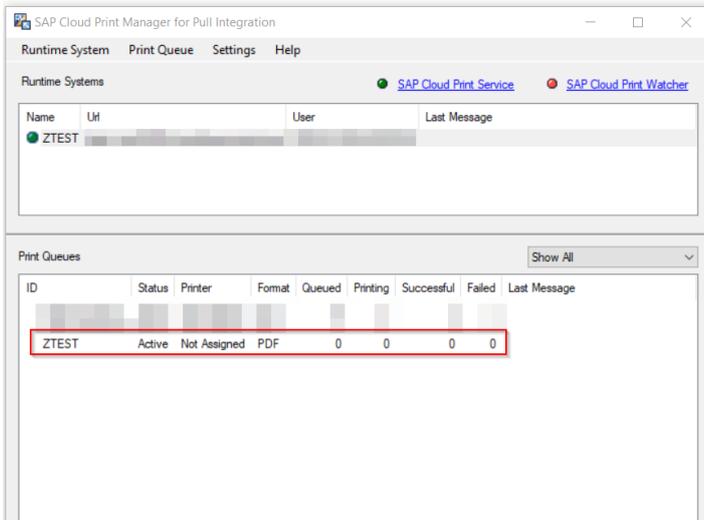


Domain value can also be found with following steps.



After opening the Print Manager, the following steps should be followed.





That was it for the Print Manager settings. To send a PDF from ABAP code to queue, the following steps are applied.

```
DATA(lv_print_data) = c1_web_http_utility=>decode_x_base64( encoded = ls_response-filecontent ), BASE64 file content should be converted XSTRING type.  
DATA(lv_qitem_id) = c1_print_queue_utils=>create_queue_itemid( ).  
  
c1_print_queue_utils=>create_queue_item_by_data(  
  EXPORTING  
    iv_qname           = 'ZTEST'  
    iv_print_data      = lv_print_data  
    iv_name_of_main_doc = 'ZTEST'  
    iv_itemid          = lv_qitem_id  
  IMPORTING  
    ev_err_msg         = DATA(lv_err_msg)  
  ).
```

"iv_qname" is used to queue name. "iv_print_data" is used to content data with XSTRING type.
"iv_name_of_main_doc" is used to assign name of the document to be saved. "iv_itemid" is used for the file's unique id. "ev_err_msg" is used for error messages.

