International Journal of Science and Research (IJSR)
ISSN: 2319-7064
ResearchGate Impact Factor (2018): 0.28 | SJIF (2019): 7.583

SAP AWS Cloud Dynamic Adobe Form Solution

Deepak Kumar[1], Karthik Matam[2]

[1]Wilmington, USA
deepak3830 [at]gmail.com
[2]Hyderabad, India
karthik452 [at]gmail.com

Abstract: ***SAP Adobe Interactive Forms are tools that seamlessly integrate with SAP systems allowing for the creation of personalized and user-friendly documents. By utilizing Adobe technology within the SAP environment these forms enhance the user experience by enabling real-time data integration interactively. Businesses, and industries, rely on SAP Adobe Interactive Forms to streamline their document processes such as generating invoices, contracts, and HR-related paperwork. However, frequent changes to the layout of SAP Interactive Forms can bring about challenges. These challenges include disruptions in data integration compatibility issues with existing scripts and the need for testing. Maintaining data integrity while adapting to evolving layouts through coordination between design modifications and system functionalities is crucial. This requires excessive planning and execution. Furthermore, frequent changes in SAP Interactive Form layouts can have implications as well. These implications may include increased costs for design and development efforts, potential disruptions in document workflows, and additional expenses for user training. To address these challenges effectively, integrating SAP with AWS Lambda and AWS S3 offers a proven solution. This collaboration ensures data flow and storage while enhancing the reliability and efficiency of form maintenance, within the combined SAP AWS environment.***

Keywords: SAP, SAP Adobe interactive forms, SAP AWS integration, AWS lambda, AWS S3

1.Introduction

SAP Interactive Forms, by Adobe, is a solution developed jointly by SAP and Adobe offering capabilities for creating and manipulating forms. This solution seamlessly integrates into both the design time and run time environments of SAP. It empowers businesses to process data through form-based interactions and can be utilized in application development settings. SAP Adobe Forms is a component of the SAP Web Application Server requiring the installation of Adobe Life Cycle Designer on your system and the configuration of Adobe Document Services (ADS) on the server to develop SAP Adobe forms. The interactive Adobe form and its corresponding driver program are implemented within the SAP system.

The integration of SAP with Amazon Web Services (AWS) combines the strengths of SAP enterprise solutions with the agility, scalability, and cost-effectiveness offered by AWS cloud infrastructure. This collaboration enables businesses to leverage cloud resources for running their SAP applications while fostering innovation.

Amazon Web Services (AWS) Lambda is a computing service that facilitates serverless execution of code without any need for server provisioning or management. With Lambda, users can execute functions in response to events, automatically scaling based on demand.

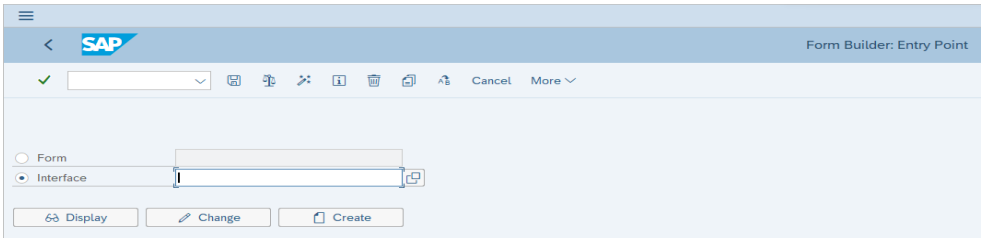
Node. js stands as a source, cross-platform JavaScript runtime built upon the V8 JavaScript engine. It empowers developers to run JavaScript code on the server side enabling the creation of network applications that are scalable and performant. Node. js is a programming framework that excels, in handling real-time applications, APIs, and microservices due to its event-driven and nonblocking nature. It has become a technology in web development providing a lightweight and versatile runtime environment for executing server-side JavaScript code.

SAP ABAP (Advanced Business Application Programming) is a programming language developed by SAP for creating enterprise-level business applications. At the heart of SAP's software stack, ABAP empowers developers to
customize SAP applications by supporting data manipulation implementing business logic and developing user interfaces. ABAP programs seamlessly integrate with SAP systems through the SAP Runtime Environment.

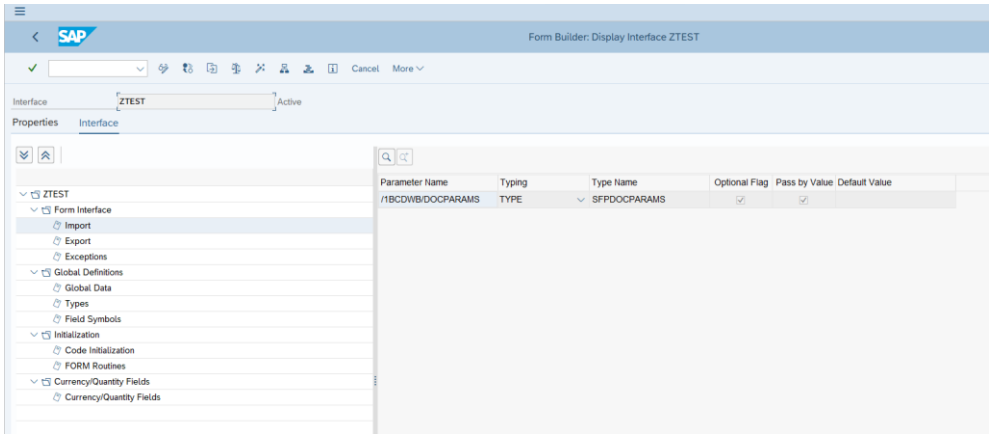
APIs (Application Programming Interfaces) act as sets of protocols and tools that facilitate communication and interaction, between software applications. By defining how various software components should interact APIs enable developers to access features or data without exposing the mechanisms underlying them.

2.How to design SAP interactive Adobe form

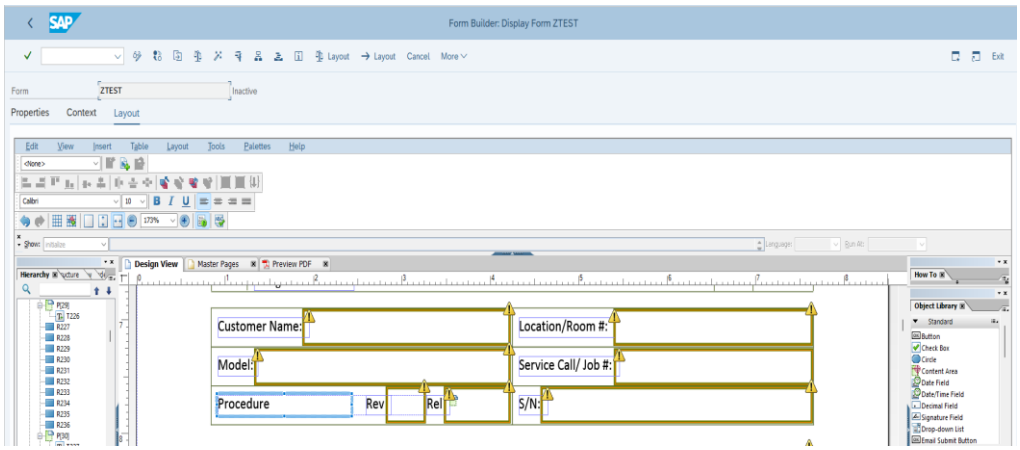
Step 1: Open transaction code SFP



Step 2: Create a form interface to establish the structure and data elements that the form will interact with. These can be fields from SAP data structures, internal tables, or context nodes.



Step 3: Design the form layout by using the LiveCycle Designer's drag-and-drop interface to design the form layout. Add form fields, text elements, and graphics as needed.



Step 4: Integrate interactive elements such as dropdowns, checkboxes, and buttons. Utilize scripting (FormCalC) to add dynamic behaviors to the form.

Step 5: Save and activate the form.

Step 6: Start Transaction SE38 and create a new driver program.

Step 7: Write ABAP code as per the business requirements to retrieve the data to be printed on the interactive Adobe form.

Step 8: Call the below important function modules to trigger the Adobe interactive form.

```
    "Get the name of the generated FM for the Adobe Form
    CALL FUNCTION 'FP_FUNCTION_MODULE_NAME'
      EXPORTING
        i_name     = lc_formname
      IMPORTING
        e_funcname = lv_fm_name.

*Pass input parameter to FP_JOB_OPEN
    ls_outputparams-getpdf    = lc_x.
    ls_outputparams-nodialog  = lc_x.
    ls_outputparams-preview   = lc_x.


    CALL FUNCTION 'FP_JOB_OPEN'
      CHANGING
        ie_outputparams = ls_outputparams
      EXCEPTIONS
        cancel          = 1
        usage_error     = 2
        system_error    = 3
        internal_error  = 4
        OTHERS          = 5.
    IF sy-subrc = 0.
      lv_docparams-langu    = sy-langu.
      lv_docparams-fillable = lc_f.
```

```
      CALL FUNCTION lv_fm_name
        EXPORTING
          /1bcdwb/docparams  = lv_docparams
          iv_guid_id         = iv_header_guid
        IMPORTING
          /1bcdwb/formoutput = ls_formoutput
        EXCEPTIONS
          usage_error        = 1
          system_error       = 2
          internal_error     = 3
          OTHERS             = 4.
      IF sy-subrc <> 0.
* Implement suitable error handling here
      ELSE.
        ev_fpcontent = ls_formoutput-pdf.
      ENDIF.

      CALL FUNCTION 'FP_JOB_CLOSE'
        IMPORTING
          e_result       = ls_jobclose
        EXCEPTIONS
          usage_error    = 1
          system_error   = 2
          internal_error = 3
          OTHERS         = 4.
      IF sy-subrc <> 0.
* Implement suitable error handling here
      ENDIF.
    ENDIF.
```

To implement any changes in the layout of an Adobe form, we need to repeat the above steps before testing and deploying the changes to the production environment. However, with the SAP cloud Adobe form solution, business teams can directly modify the interactive PDF forms using the PDF editor and replace them on the AWS

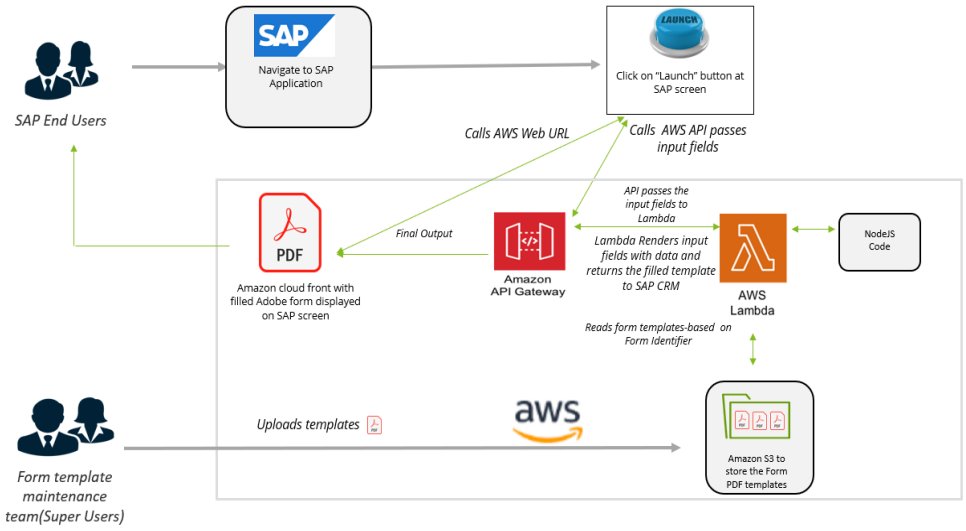
S3 repository. This eliminates the need for repetitive steps and simplifies the process of making changes to the Adobe forms.

3.Problem Statement

Designing and maintenance of Interactive Forms works well when we have a fixed layout on Adobe forms. When business requirements demand variable layouts that may be changing more frequently, the above process becomes very tedious, and it is not possible to make changes in Adobe Forms without making any technical Changes. Every time any change is required, business teams must pass on the requirements to the technical team to perform the changes and import the change to the production environment.

4.Solution Details

The approach explained here is a solution to the problem explained above. In this approach, layout design and maintenance ownership will reside with the business super users and does not require the SAP technical team's involvement. The business team can identify the sections of Adobe Forms that will require data to be rendered from SAP and sections that require fixed data or manual inputs. In this approach, data can be transferred from SAP by creating a NodeJS application and deploying it in the AWS Lambda function and keeping the layouts or the Adobe interactive forms on AWS S3 buckets. Variable data can be fetched from the SAP system and passed over to AWS Lambda functions by using the NodeJS code. In this way the dynamic and static part of the form can remain in total control of the business team and the technical team will just need the variable data to be printed on the form.

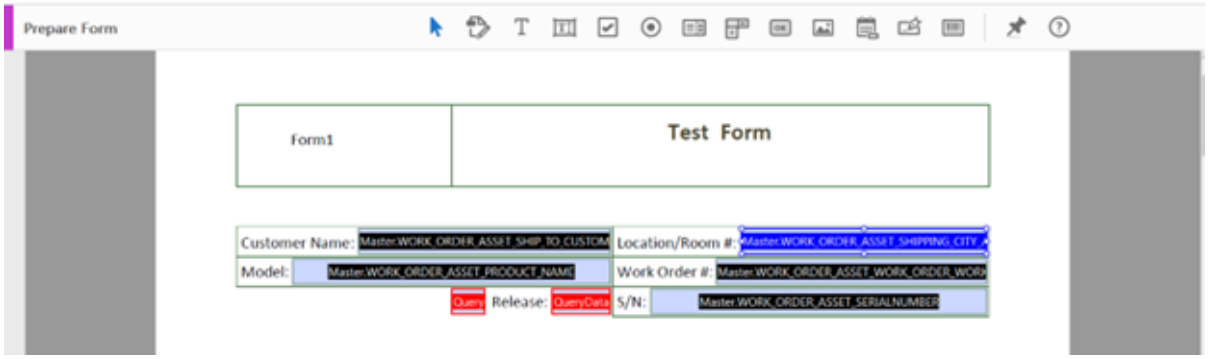


5.Business use case to explain the solution

Before showing up at the customer site, the Service Technicians will download the latest version of the service manual. These forms are dependent on the instrument and the type of service. They are typically data sheets or certificates of calibration. The correct forms will be available based on the service type and instrument in the work order. These forms will pre-populate the necessary customer information. They also will prepopulate the serial number and required calibration information for the jigs that the engineer needs to use for the service. Based on the Service Technician and Instrument, SAP will auto-populate the serialized, calibrated jigs that are required for the work. Jig details are pulled from the calibration system and stored in the SAP System.

6.Technical Solution details

1. Prepare the form using Adobe Acrobat as shown below:



2. The process will start from the SAP screen where the user will click on the button to trigger the display of the interactive Adobe form.
3. With the click of a button, the SAP code will trigger which includes the below key components.
4. Create a deep structure to be converted into JSON format.

```
TYPES: BEGIN OF ty_[redacted],
         templateid  TYPE string,
         pdffilename TYPE string,
         BEGIN OF [redacted]
           custname   TYPE string,
           location   TYPE string,
           model      TYPE string,
           workorder  TYPE string,
           rev        TYPE string,
           rel        TYPE string,
           serial     TYPE string,
           ownername  TYPE string,
           ownertitle TYPE string,
         END OF [redacted],
       END OF ty_[redacted]
```

5. After getting the details convert the corresponding structure into JSON using the method CL_FDT_JSON=>DATA_TO_JSON

```
* Convert structure to JSON
      CONCATENATE[redacted]_id '_' [redacted]_json-templatedata-serial INTO[redacted]_json-pdffilename.
      CALL METHOD cl_fdt_json=>data_to_json
        EXPORTING
          ia_data = ls_json
        RECEIVING
          rv_json = lv_json_string.
```

6. To send the JSON to AWS get the AWS Key and URL saved in SAP tables/variables.

```
*Read API key from TVARVC[redacted]_API_KEY'
       SELECT SINGLE low FROM tvarvc INTO @lv_api_key WHERE name =[redacted]API_KEY'.
*Read AWS API Url from TVARVC[redacted]_API_URL'
       SELECT SINGLE low FROM tvarvc INTO @lv_afd_url WHERE name =[redacted]API_URL'.
       IF sy-subrc <> 0.
         lv_afd_url = 'https:/[redacted]fd-update-pdf'.
       ENDIF.
```

7. The factory method CL_HTTP_CLIENT=>CREATE_BY_URL is used in this case to instantiate the ABAP HTTP object.

```
CALL METHOD cl_http_client=>create_by_url
  EXPORTING
    url                = lv_afd_url
  IMPORTING
    client             = lo_http_client
  EXCEPTIONS
    argument_not_found = 1
    plugin_not_active  = 2
    internal_error     = 3
    OTHERS             = 4.
IF sy-subrc <> 0.
  lr_message_service = me->view_manager->get_message_service( ).
  CALL METHOD lr_message_service->add_message
    EXPORTING
      iv_msg_type   = 'E'
      iv_msg_id     = [redacted]        "Messageclass
      iv_msg_number = [redacted].       "Messagenumber
  RETURN.
ENDIF.
"setting request method as POST
lo_http_client->request->set_method('POST').
lo_http_client->propertytype_accept_cookie = if_http_client=>co_enabled.
"adding headers
lo_http_client->request->set_header_field( name = 'Content-Type' value = 'application/json' ).
lo_http_client->request->set_header_field( name = 'X-API-Key' value = [redacted]api_key ).
lo_http_client->request->set_header_field( name = 'Accept' value = 'application/json' ).
LOOP AT lt_cookies ASSIGNING FIELD-SYMBOL(<cookie>).
  lo_http_client->request->set_cookie( name = <cookie>-name
  value = <cookie>-value ).
ENDLOOP.
```

8. Replace the SAP field names with the field names defined on the Adobe interactive forms.

```
REPLACE '"TEMPLATEID"' WITH '"templateId"' INTO [redacted]json_string.
REPLACE '"TEMPLATEDATA"' WITH '"templateData"' INTO [redacted]json_string.
REPLACE '"PDFFILENAME"' WITH '"pdffileName"' INTO lv_json_string.
REPLACE '"CUSTNAME"' WITH '[redacted] ASSET_SHIP_TO_CUSTOMER_NAME_FORMULA"' INTO [redacted]json_string.
REPLACE '"LOCATION"' WITH '[redacted] ASSET_SHIPPING_CITY_AND_STATE_FORMULA"' INTO [redacted]on_string.
REPLACE '"MODEL"' WITH '[redacted] ASSET_PRODUCT_NAME"' INTO [redacted]_string.
REPLACE '"WORKORDER"' WITH '[redacted] ASSET_WORK_ORDER_WORKORDERNUMBER"' INTO [redacted]n_string.
REPLACE '"REV"' WITH '[redacted] REASON_PROCEDURE_REV"' INTO [redacted]on_string.
REPLACE '"REL"' WITH '[redacted]ASON_PROCEDURE_RELEASE"' INTO [redacted]n_string.
REPLACE '"SERIAL"' WITH '[redacted] SERIALNUMBER"' INTO [redacted]on_string.
REPLACE '"OWNERNAME"' WITH '[redacted] OWNER_NAME_FORMULA"' INTO [redacted]on_string.
REPLACE '"OWNERTITLE"' WITH '[redacted] OWNER_TITLE_FORMULA"' INTO [redacted]on_string.
```

9. Trigger AWS and get the response as a URL.

```
CALL METHOD lo_http_client->request->set_cdata
  EXPORTING
    data = lv_json_string.
CALL METHOD lo_http_client->send
  EXCEPTIONS
    http_communication_failure = 1
    http_invalid_state         = 2
    http_processing_failed     = 3
    http_invalid_timeout       = 4
    OTHERS                     = 5.
IF sy-subrc = 0.
  CALL METHOD lo_http_client->receive
    EXCEPTIONS
      http_communication_failure = 1
      http_invalid_state         = 2
      http_processing_failed     = 3
      OTHERS                     = 5.
ENDIF.
IF sy-subrc <> 0.
  lr_message_service = cl_bsp_wd_message_service=>get_instance( ).
  CALL METHOD lr_message_service->add_message
    EXPORTING
      iv_msg_type   = 'E'
      iv_msg_id     = [redacted]        "Messageclass
      iv_msg_number = '003'.            "Messagenumber
  RETURN.
ENDIF.

response_str = lo_http_client->response->get_cdata( ).
```

10. This response is converted to ABAP structure using method ‘CL_FDT_JSON=>JSON_TO_DATA’.

```
      response_str = lo_http_client->response->get_cdata( ).
      DATA: BEGIN OF response_abap,
Convert response JSON to Structure
      CALL METHOD cl_fdt_json=>json_to_data
        EXPORTING
          iv_json = [redacted]str
        CHANGING
          ca_data = [redacted]_abap.
```

11. The 'WINDOW. OPEN' in will be called to open the URL in a new window in DO_PREPARE_OUTPUT.

```
* Set URL and iprint Flag which will be used in DO_PREPARE_OUTPUT to launch URL in new window
      IF response_abap-data IS NOT INITIAL.
        CONCATENATE '<script type="text/javascript"> window.open("' response_abap-data '") </script>' INTO gv_pdf_relative_url.
        gv_printpdf_flag = abap_true.
      ELSE.
        lr_message_service = cl_bsp_wd_message_service=>get_instance( ).
        CALL METHOD lr_message_service->add_message
          EXPORTING
            iv_msg_type   = 'E'
            iv_msg_id     = [redacted]      "Messageclass
            iv_msg_number = [redacted]      "Messagenumber
        RETURN.
      ENDIF.
```

12. We send values that need to be populated in the Form template in API request BODY.

```
app.post('/afd_pdf', (req, res) => {
  const reqBody = req.body;
  console.log('You provided Service Authorizer Name:' + reqBody.SERAUTHNAME + ', Service Number:' + reqBody.SERNUM + ', Service Authorizer P
    + ', Service Authorizer Email:' + reqBody.SERAUTHEMAIL + ', Service Authorizer Company Name:' + reqBody.SERAUTHCMPY + ', ModelNumber:' +
    + ', Description:' + reqBody.DESCRIPTION + ', Service Type:' + reqBody.SERTYPE + ', Customer Notes:' + reqBody.CUSTNOTES + ',Shipping Co
    + ', Shipping Attention To' + reqBody.SHIPATTO + ', Shipping Address 1' + reqBody.SHIPADDR1 + ', Shippig Address 2' + reqBody.SHIPADDR2 +
    + ', Shipping Country' + reqBody.SHIPCOUNTRY + ', Shipping Zip Code' + reqBody.SHIPZIP + ', Shipping Phone' + reqBody.SHIPPH + ', Model

  run().catch(err => console.log(err));

async function run() {
  const content = await PDFDocument.load(fs.readFileSync('./template.pdf'));
  const form = content.getForm();
  const serviceauthname = form.getTextField('Service Authorizer Name');
  const servicenumber = form.getTextField('Service Number');
  const serviceauthph = form.getTextField('Service Authorizer Phone');
```

13. The AWS Lambda NodeJS function will fetch the template from S3 based on the template ID in the API request Body, populate the other values in the template, and provide the URL which is the endpoint for the updated PDF document.

```
    //res.setHeader('Content-disposition', 'attachment; filename=' + reqBody.form + '_' + reqBody.workorder + '.pdf');
    res.setHeader('Content-disposition', 'inline; filename=' + reqBody.SERNUM + '_' + reqBody.SERAUTHNAME + '.pdf');
    res.setHeader('Content-type', 'application/pdf');
    // Write the PDF to a file
    const pdfBytes = await content.save();
    const readStream = new stream.PassThrough();
    readStream.end(pdfBytes);
    readStream.pipe(res);
  }
})
```

14. We need to use the URL returned from the POST call to launch the PDF.

7.Conclusion

1. Reduced efforts by 70% as compared to the traditional way of building layout using Adobe Forms in SAP
2. Adobe template maintenance is taken care of by the business.
3. The SAP Technical team only focuses on sending the dynamic data to AWS if any new dynamic fields are added to the template.
4. One solution for all the templates irrespective of different forms

Declarations:

- Ethics approval and consent to participate: Not Applicable
- Consent for publication: All authors have consent to submit this paper to the Journal of Cloud Computing. Also, we confirm that this paper or any part of this paper was not submitted anywhere.
- Availability of data and materials: Not Applicable
- Competing interests: Not Applicable
- Funding: Not Applicable
- Authors' contributions:

R. A. Topics Covered: SAP AWS Cloud Dynamic Adobe Form Solution
R. K. Topics Covered: Abstract, Introduction, and How to Design SAP interactive Adobe form.
D. N. M. Topics Covered: Conclusion and Declarations
All Authors have reviewed the manuscript.

Acknowledgments

Thank you co-author, Karthik Matam, for his expertise and assistance throughout all aspects of our study and for your help in covering a few topics and reviewing the manuscript.

References

[1] "SAP Help Portal," help.sap.com. https://help.sap.com/docs/SAP_NETWEAVER_740/6f3c61a7a5b94447b80e72f722b0aad7/a9b128543eaa4a508b5120b695e29391.html
[2] "SAP Adobe Form - Steps to create simple ADOBE Form and calling it from ABAP Program." https://an-sap-consultant.blogspot.com/2014/04/sap-adobe-form-steps-to-create-simple-ADOBE-Form-and-calling-it-from-ABAP-Program.html
[3] "How to create interactive PDF files: interactive PDFs | Adobe Acrobat," www.adobe.com. https://www.adobe.com/acrobat/hub/how-to-make-a-pdf-interactive.html
[4] "SAP IDoc integration with Amazon S3 by using Amazon API Gateway | AWS for SAP," aws.amazon.com, Jul. 11, 2019. https://aws.amazon.com/blogs/awsforsap/sap-idoc-integration-with-amazon-s3-by-using-amazon-api-gateway/
[5] "AWS Lambda | AWS Blog," aws.amazon.com. https://aws.amazon.com/blogs/aws/category/aws-lambda/
[6] "Best Practices for Amazon S3." https://d1.awsstatic.com/events/reinvent/2019/REPEAT_1_Best_practices_for_Amazon_S3_(including_storage_classes)_ft._Instructure_STG302-R1.pdf
[7] "ABAP to JSON Conversion | SAP Blogs," blogs.sap.com. https://blogs.sap.com/2019/10/21/abap-to-json-with-custom-transformation/
[8] S. PRESS, "How to Prepare an Interactive PDF Form for SAP," blog.sap-press.com. https://blog.sap-press.com/how-to-prepare-an-interactive-pdf-form-for-sap
[9] former_member, "Step By Step Method To Create An Adobe Form With Dynamic Variables Along With An External Layout," SAP Community, Jan. 27, 2013. https://community.sap.com/t5/technology-blogs-by-members/step-by-step-method-to-create-an-adobe-form-with-dynamic-variables-along/ba-p/13232494