eSign ASP User Interface Integration Guide **CUSTOMER**

Document Version: 1.1 – 2017-08-14

Integration between Digital Compliance Service User Interface and eSign Application Service Provider User Interface

Version 1.1.0

Typographic Conventions

Type Style	Description
Example	Words or characters quoted from the screen. These include field names, screen titles, pushbuttons labels, menu names, menu paths, and menu options. Textual cross-references to other documents.
Example	Emphasized words or expressions.
`EXAMPLE`	Technical names of system objects. These include report names, program names, transaction codes, table names, and key concepts of a programming language when they are surrounded by body text, for example, SELECT and INCLUDE.
`Example`	Output on the screen. This includes file and directory names and their paths, messages, names of variables and parameters, source text, and names of installation, upgrade and database tools.
`Example`	Exact user entry. These are words or characters that you enter in the system exactly as they appear in the documentation.
`<Example>`	Variable user entry. Angle brackets indicate that you replace these words and characters with appropriate entries to make entries in the system.
`EXAMPLE`	Keys on the keyboard, for example, `F2` or `ENTER`.

Document History

Version	Date	Change
1.0	2017-07-24	First release of the eSign User Interface Integration Guide
1.1	2017-08-14	Document enhancement with additional information

Contents

1 Introduction

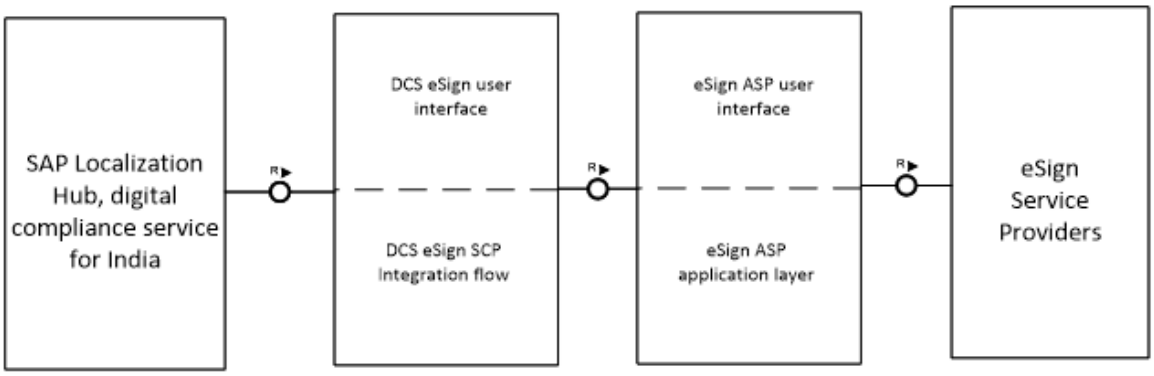
You integrate SAP Localization Hub, digital compliance service with the eSign application service provider (eSign ASP) to digitally sign filing document content before filing returns. The SAP Cloud Platform Integration Service integrates the SAP Localization Hub, digital compliance service with the eSign application service provider (eSign ASP). This document provides steps to establish communication between SAP Localization Hub, digital compliance service and eSign ASP application.

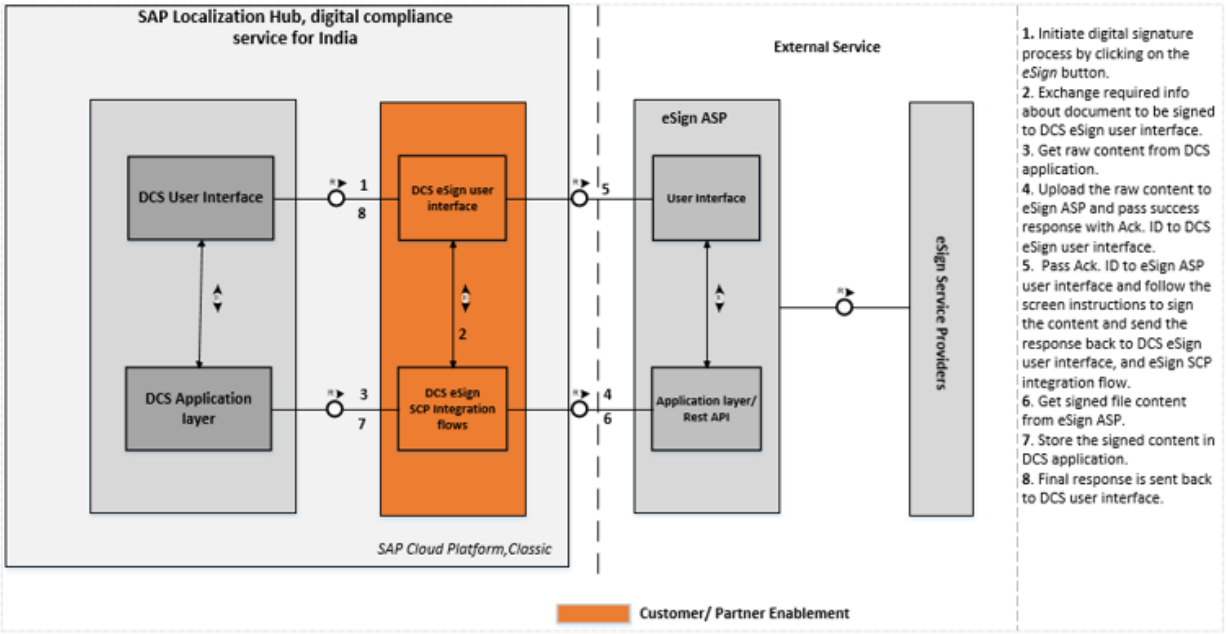
> **Note**
>
> This documentation and configuration would be required only if you want to digitally sign the document content via the online process as mentioned in the **SAP Localization Hub, digital compliance service: Digital Signature Process** documentation. Alternately, you can use the offline process using Digital Signature Certificate(DSC) USB device to digitally sign the filing content.

eSign Service Provider (ESP), also known as certificate authority or *Certifying Authorities* (CA), is an entity that issues digital signature certificates for electronic authentication of users.

eSign Application Service Provider (eSign ASP) is an entity that signs the contract with the ESP to provide electronic signature Service. eSign ASP uses eSign service as part of their application to digitally sign the content. eSign ASP provides the interface software platform for users to sign documents using Aadhaar + OTP or using Digital Signature Certificate (DSC) USB device.

The below diagram provides an overview of the integration between SAP Localization Hub, digital compliance service and eSign ASP.





2 Prerequisites

Before you start with the activities mentioned in this document, ensure that the following prerequisites are met:

1. Installed the *SAP Localization Hub, digital compliance service for India* solution in your *SAP Cloud Platform, Neo* test and/or productive landscape. For detailed information, refer SAP note 2460667.
2. Obtained the DSC USB device.
3. Have a registered PAN, as recommended by the GSTN, at the time of filing digitally signed content.
4. Provisioned live SCP Integration Service test and/or productive tenants.
5. Chosen eSign ASP or eSign gateway service providers and completed the registration process.
6. You have received the following information or documents after on boarding with *eSign ASP*:
   - eSign ASP integration or setup manual guide
   - JavaScript SDK
   - Certificate required for establishing SSL handshake with eSign ASP.
7. You have already configured and deployed eSign ASP Integration flow and have obtained endpoint URL to access Integration flow.
   For more information, see *eSign ASP Integration Guide*.

3 eSign ASP Integration Content

The content catalog package *Integration between Digital Compliance Service to GST Suvidha Provider and eSign Application Service Provider* contains the following iFlows:

iFlow Name in WebUI	Project Names/Artifact Names
GSP Integration Template	com.sap.slh.dcs.gsp.template
eSign ASP Integration Template	com.sap.slh.dcs.esp.template

eSign ASP integration flow is a generic integration template and should be modified or enhanced for integration with eSign ASP. This template contains process calls to set up communication with the DCS application. The process call for communication with eSign ASP must be added based on the details received after registering with eSign ASP.

> **Note**
>
> You can find Integration flow in content catalog *Integration between Digital Compliance Service to GST Suvidha Provider and eSign Application Service Provider* by following the below procedure.

Procedure

To find the integration flow in the content catalog, perform the following steps:

1. In your browser, go to the WebUI of the SCP integration tenant using the url:
   `<SCP Integration Tenant URL>/itspaces`.
2. To logon, enter your **P user** or **S user**.
   If you get the *HTTP Status 403* error, contact your tenant administrator.
3. After successful login, from the menu in the upper left corner, choose *Discover*.
4. In the subsequent screen, search for *Integration between Digital Compliance Service to GST Suvidha Provider and eSign Application Service Provider*, and select the package.

4 DCS eSign ASP SAPUI5 Application Setup

The DCS eSign ASP SAPUI5 application acts as an interface between the DCS application screen and eSign ASP or Web service screen by exchanging data between the screens.

4.1 Create Destination for eSign ASP SAP Cloud Platform Integration flow

To access eSign ASP Integration flow, create a destination which will be accessed by the DCS eSign ASP user interface application.

Procedure

1. Create a new destination in the SCP cockpit following the steps mentioned here.
2. Maintain the following values

- Name: <destination name>
- Type: HTTP
- Description:<Description of the destination>
- URL: <eSign ASP Integration flow endpoint URL https://<SCP Integration Tenant URL>>/http/dcs/esign>
- Proxy Type: Internet
- Authentication: Basic Authentication
- User: < enter your **P user** or **S user** with access to SAP Cloud Platform Integration tenant where eSign ASP Integration flow is deployed>
- Password: <user password>

4.2 Create SAPUI5 application

Create a custom SAPUI5 application to capture the relevant information as per the requirements of the eSign ASP.

To create an SAPUI5 application in SAP Cloud Platform Web IDE, refer here.

Procedure

1. Open SCP cockpit in browser and select *Service*.
2. Search and select *SAP Web IDE*.
3. In the subsequent screen, select *Go to Service*.
   The system redirects you to the SCP Web IDE screen.
4. To create a new SAPUI5 application, select *New -> Project from Template.*
   For more information refer here.
5. Select *SAPUI5 application* from the template selection and follow the wizard to create the project.
6. From the created project, open the **neo-app.json** file and maintain the destination details that you have already created in section 4.1.

```
{
        "path": "/destinationaccess",
        "target": {
 "type": "destination",
 "name": "< destination name>",
 "entryPath": "/"
  },
 "description": "<Description of the destination>"
 }
```

7. From the created project, open the view and make changes as per your business requirements to capture the relevant information required to be exchanged with eSign ASP APIs and DCS APIs.

   For example, the below code can be used in view (view type: XML) to create:

   - input field for capturing email ID and PAN
   - button to trigger eSign.

   If you use the following code snippet, also ensure that you add the *xmlns:f="sap.ui.layout.form"* library to the view *mvc*.

   Code

```
<content>
<f:SimpleForm editable="true" layout="ResponsiveGridLayout" maxContainerCols="2" labelSpanL="4" labelSpanM="4" labelSpanS="6">
<f:content>
<Label text="{i18n>email}" required="true"/>
<Input type="Email" id="email" width="60%" />
```

```
<Label text="{i18n>PanDetails}" required="true"/>
<Input id="panDetId" width="60%" maxLength="10" />
                                                    </f:content>
                                                </f:SimpleForm>
                                            </content>
                                            <footer>
<Toolbar>
<ToolbarSpacer/>
<Button text="{i18n>Esign}" press="onPress"/>
</Toolbar>
</footer>
```

8. Open the controller file linked to the view created above, and create a function that the system calls on choosing the *eSign* button.

   > **Note**
   >
   > The DCS application will pass below mentioned parameters/values to DCS eSign UI application during cross-application navigation. Provided information should be used to retrieve the document that is to be signed.

   The following table details the information passed from DCS application UI to DCS eSign UI:

Parameter Name	Type	Description	Example
GSTNumber	String	GST Identification number	
ReportCategoryID	String	GST Return type.	IN_GSTR1
ReportingPeriod	String	Reporting period	022017 where **02** indicates reporting month and **2017** indicates reporting year
ReportingActivityID	String	Activity for which filing is done.	GSTR1FILE

9. When you choose the *eSign* button, the system initiates the first call to eSign ASP integration flow to download the file content to be signed from DCS application API, and then upload the same to eSign ASP API. Next, navigate to the eSign ASP user interface to complete the signing process.
   Use the below code snippet in the eSign button function:

```
onPress: function(oEvent)
    {
        var oURLParsing = new sap.ushell.services.URLParsing();
        var url = oURLParsing.getHash(location.href);
        var oShellHash = oURLParsing.parseShellHash(url);
```

```
                var data = {
//request payload to be filled and passed to eSign ASP Integration flow
                        "request": {
                        "gstin": oShellHash.params.GSTNumber[0],
                        "reportingcategory": oShellHash.params.ReportCategoryID[0],
                        "period": oShellHash.params.ReportingPeriod[0],
                        "repactivityid": oShellHash.params.ReportingActivityID[0],
                        "action": "dcs_download",
                        email": this.getView().byId("email ").getValue(),
                        "docid": "",
                        "placeholder1": "",
                        "placeholder2": "",
                        "stype":"<Type of signature>",
                        "sid": this.getView().byId("panDetId").getValue()
        }
                        };

                        var destinationaccessURL = "/destinationaccess";
                        var that = this;
                        jQuery.ajax({
                                url: destinationaccessURL,
                                contentType:'application/json',
                                method:"POST",
                                data: JSON.stringify(data),
                                success: function(oResp) {
                // Integrate JavaScript SDK provided by eSign ASP
                                },
                                error: function(oError) {
        sap.m.MessageBox.error(oError.responseText.split("</h1>")[1].split("</body>")[0]);
                                }
                        });
}
```

Note

In *Success* function of AJAX call, integrate JavaScript SDK provided by eSign ASP to call eSign ASP user interface. Using JavaScript SDK, you can navigate to eSign ASP user interface.

The following table details the request payload:

Parameter name	Data type	description	example
gstin	String	GST Identification number	
reportingcategory	String	GST Return type	IN_GSTR1
period	String	Reporting period	022017 where **02** indicates reporting month and **2017** indicates reporting year
repactivityid	String	Activity for which filing is done.	GSTR1FILE
action	String	Indicator to decide action required on integration flow Possible default value: dcs_download dcs_upload	dcs_download: To download the file content to be signed from DCS application API and upload the same to eSign ASP API. dcs_upload: To download the signed content from eSign ASP API and upload the same to DCS application API
email	String	eSign requestor email	
docid	String	eSign ASP Acknowledgment Id	Acknowledgement ID received for uploading file content to be signed to eSign ASP API
placeholder1	String	Field to pass any additional value if required.	
placeholder2	String	Field to pass any additional value if required	
stype	String	sType of signature	DSC or ESIGN
sid	String	PAN of authorized representative if stype = DSC or AADHAR no. of authorized representative if stype=ESIGN	

Use the below code snippet to call eSign ASP integration flow and download the signed content, and the upload the same to the DCS API. Note that the below-mentioned code snippet should be called in *Success* function.

```
var oURLParsing = new sap.ushell.services.URLParsing();
var url = oURLParsing.getHash(location.href);
var oShellHash = oURLParsing.parseShellHash(url);
var data = {
```

```
//request payload to be filled and passed to eSign ASP Integration flow
                        "request": {
                        "gstin": oShellHash.params.GSTNumber[0],
                        "reportingcategory": oShellHash.params.ReportCategoryID[0],
                        "period": oShellHash.params.ReportingPeriod[0],
                        "repactivityid": oShellHash.params.ReportingActivityID[0],
                        "action": "dcs_upload",
                        email": this.getView().byId("email").getValue(),
                        "docid": "<acknowledgement ID received for eSign ASP>",
                        "placeholder1": "",
                        "placeholder2": "",
                        "stype":"<Type of signature>",
                        "sid": this.getView().byId("panDetId").getValue()
        }
                        };

                        var destinationaccessURL = "/destinationaccess";
                        var that = this;
                        jQuery.ajax({
                                url: destinationaccessURL,
                                contentType:'application/json',
                                method:"POST",
                                data: JSON.stringify(data),
                                success: function(oResp) {
                                window.history.back(1);
                                },
                                error: function(oError) {
sap.m.MessageBox.error(oError.responseText.split("</h1>")[1].split("</body>")[0]);
                                }});
```

4.3 Deploy SAPUI5 application to SAP Cloud Platform.

Once the SAPUI5 application development is completed and is executable, deploy SAPUI5 project from SAP Web IDE to SAP Cloud Platform as a new application or update to a previously deployed application in SAP Cloud Platform accounts.
For more details, see here.

4.4 Registering SAPUI5 application to SAP Fiori Launchpad

You can register an SAPUI5 application to SAP Fiori Launchpad directly from SAP Web IDE. Once registration is complete, a new tile is created in the launchpad which is assigned to a site, a catalog, and a group. You can navigate to the SAPUI5 application from the DCS application using a semantic object and action.

Prerequisites

- Make sure you are assigned to the **TENANT_ADMIN** role in the SAP Cloud Platform cockpit. For more information, see Accessing Services.
- You have obtained the Fiori site details after installing the *SAP Localization Hub, digital compliance service for India* solution in your *SAP Cloud Platform*.

Procedure

1. Register the SAPUI5 application created to the SAP Fiori Launchpad site created for *SAP Localization Hub, digital compliance service for India.*
   For detailed information, refer here.
2. Enter the following values:

- Semantic Object: Reporting
- Action: ESign
- Tile type: Static
- Site: Digital Compliance Services - GST India
- Catalog: DCS - Manage GSTR Returns
- Groups: Returns

> **Note**
>
> You can hide the tile by selecting tile type as *No Tile*. For more information, refer here.

www.sap.com/contactsap

Material Number: