**CLM OmniScripts Overview**

By using OmniScript, you can streamline your setup and workflow. You can use this OmniScript in CLM to generate documents from any object or any page, not just contracts.

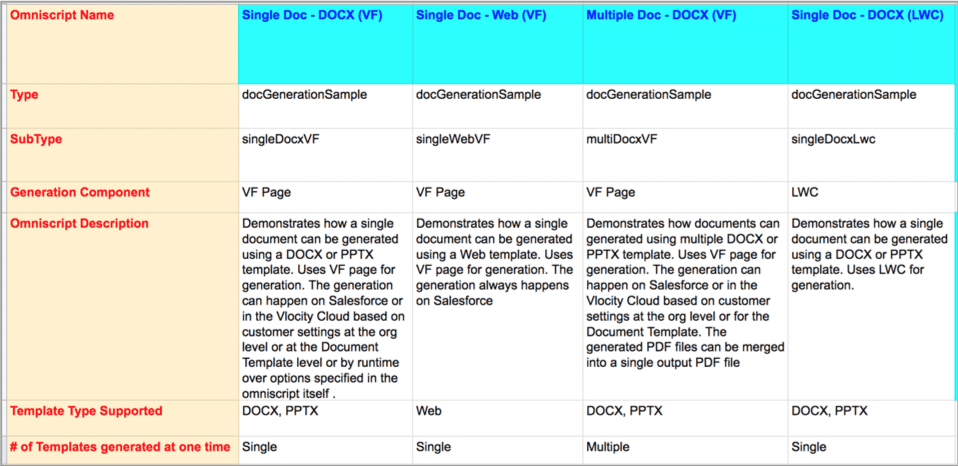
**Generic Document Generation OmniScript**

Salesforce Industries provides an OmniScript called "**Generic Document Generation"** for generating documents that have a usage type other than Contract. You can create your own usage types, and use this sample OmniScript to generate the document. This applies to JSON-based templates only.

Go to the Vlocity OmniScript Designer tab to view this OmniScript. You can clone and modify it as needed, and launch it from any page (from any record).

This OmniScript is available in our Salesforce Industries Process Library.

Sample Document Generation OmniScripts



Different types of sample OmniScripts are provided for single and multiple document generations.

These sample "purpose built' OmniScripts  can serve as starting points for an implementation.

These samples can also demonstrate document generation capabilities.

**Customizing Generic Document Generation OmniScript**

Review the below steps in this OmniScript. If you want to make changes to the structure, it's recommended that you clone a copy of this OmniScript and rename it.

|  |  |
| --- | --- |
| Generic Document/Generation OmniScript structure | * **EnterObject:**Salesforce uses ObjectId to capture the Salesforce ID of the object on which the document needs to be generated which will be assigned to contextID later in Set Values step. * **Pick a Template Type:** Salesforce uses two different types of templates: Web Templates or the Microsoft Templates (DOCX and PPTX). * **GetDocumentTemplates:**This DataRaptor Extract step uses the DataRaptor Interface GetDocumentTemplatesForType to retrieve all available active document templates depending on the document template type. * **PickTemplate:** The OmniStudio DataRaptor displays all the extracted templates in the selectable template. * **Set Values:**The type of values are set according to whether the Web templates or the DOCX and PPTX templates are set up. * **CreateObjDocument:** This is a remote action to create a document on objects. This step is required only if you're using Web Templates. |

# Generic DocuSign OmniScript

Generic DocuSign/sendEsignature OmniScript

* Sends one or more generic documents (not generated from contracts) to DocuSign for eSignatures.
* Sends documents generated from other objects, such as quote, order, or opportunity.
* Sends documents generated from custom objects.
* Documents generated using the Generic Document Generation OmniScript can only be fetched for sending to DocuSign

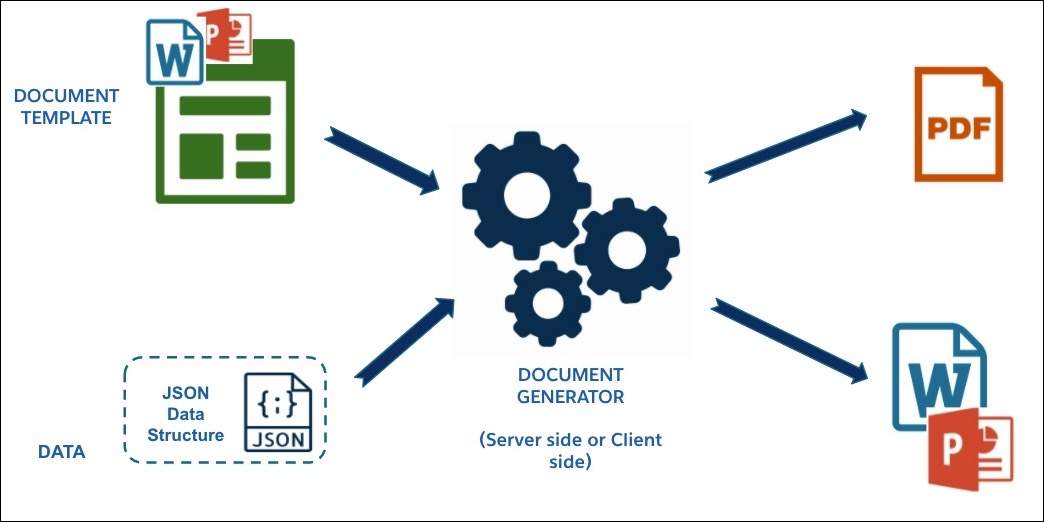
Customizing Generic DocuSign OmniScript

Review the below steps in this OmniScript. If you want to make changes to the structure, it's recommended that you clone a copy of this OmniScript and rename it.

|  |  |
| --- | --- |
| GenericDocuSign/sendEsignature OmniScript structure | * **GetContacts:**This DataRaptor extract step retrieves the recipients list, containing the names and email addresses of those you need to sign documents. * **EnterObject:**Salesforce uses ObjectId to capture the Salesforce ID of the object on which the document needs to be generated. * **Select Signers:** A custom Lightning Web Component (LWC) step for selecting the list of contacts. * **GetDocuments:**This DataRaptor Extract step retrieves the list of documents generated for the Object/ContextID provided. These documents are generated using Generic DocGen OmniScripts/docGenerationSample OmniScripts. * **SetDocumentColumns:** A required step for setting the list of columns to be shown in SelectDocuments Step. * **SelectDocuments:**This step displays the documents you can select to send for eSignature. The document list is from documents generated by other CLM OmniScripts or documents that you uploaded into the Notes & Attachment section. * **FinalizeEnvelope:** This step displays the final details, such as reviewing the document you're sending, the recipients, and the message that accompanies the documents. |

# Server-side Document Generation

Let's take a look at the dynamic document generation setup.



When you use a Microsoft Word (DOCX) or Microsoft PowerPoint (PPTX) file as a template, you insert tokens (merge fields) and display conditions directly into the DOCX or PPTX file, then upload the file into the Document Template Designer. The template is defined with a unique name and template type. The token mapping is JSON-based and the data can come from internal or external data sources.

You can set up the document generation process in one of the two ways- using either client-side or server-side document generation.

* Client-side document generation requires user interaction and generates documents using OmniScript. Salesforce provides sample OmniScripts and Integration Procedures that you can use and customize to implement these capabilities.
* Server-side document generation typically doesn’t require user interaction. Instead, a backend server processes requests to generate documents.

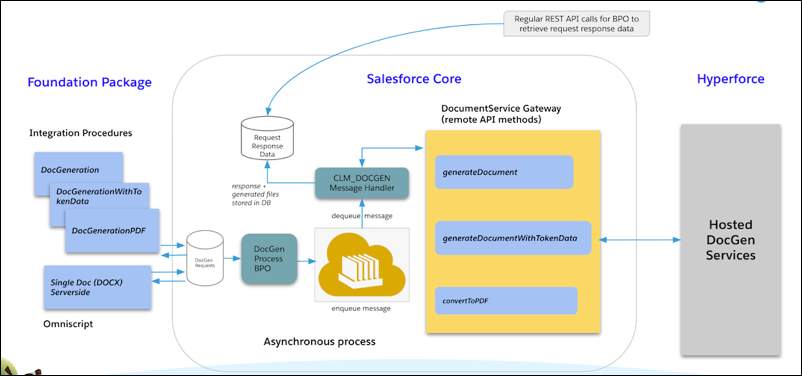
**Server-Side Document Generation**

Server-Side document generation is an asynchronous process that's best for large and rendering-heavy documents and for document generation in batches. The Server-Side document generation service is secure and scalable and is hosted on Hyperforce. The generated document is attached to the object for which it's generated.

**Benefits and Capabilities**

* **Automate document generation with APIs:** You can generate documents using APIs, without the need for user input. Configure your OmniScript to call APIs to perform server-side document generation, independent of the browser client.
* **Launch document generation from multiple sources:**You can use a backend Apex code, OmniStudio Integration Procedures, or OmniScripts.
* **Generate large documents:** You can scale up the performance significantly for large and rendering-heavy documents because it uses Salesforce compute resources instead of browser resources. It also supports batch processing when you want to generate multiple documents at the same time.
* **Implement solutions faster with sample assets:** You can use sample artifacts like OmniStudio Integration Procedures, OmniScripts, LWC components for quick implementation.
* **Secure and scalable:** Service is hosted on Hyperforce, making it secure and scalable.

Server-Side Document Generation Flow



The document generation process launches from an OmniStudio Integration Procedure, an OmniScript, or Apex code. This generates a BPO request that goes into a queue. The process can generate several Base Platform Object (BPO) requests at once, which wait in the queue until dequeuing mechanisms call remote APIs. The remote API methods run on Hyperforce, which dedicates Salesforce compute resources to the process. The methods pull requests from the queue and generate documents. Once the documents are generated, they’re stored in Salesforce as content documents

# Server Side Document Generation OmniScript

A sample docGenerationSample/singleDocxServersideLwc OmniScript is provided.

* This OmniScript generates documents either from Microsoft Word .DOCX or Microsoft PowerPoint .pptx templates by using server-side generation.
* For the OmniScript to work, IsServerSideDocGenEnabled custom setting must be set to true.
* The documents generated gets attached to the Notes & Attachments section under the Related tab of the object

**Customizing Server-Side Document Generation OmniScript**

Review the below steps in this OmniScript. If you want to make changes to the structure, it's recommended that you clone a copy of this OmniScript and rename it.

|  |  |
| --- | --- |
| docGenerationSample/singleDocxServersideLwc OmniScript structure | * **EnterObject:**Salesforce uses **ObjectId** to capture the Salesforce ID of the object on which the document needs to be generated which will be assigned to the contextID later in Set Values step. For **TemplateType**, Salesforce uses Microsoft Templates (DOCX or PPTX). * **GetDocumentTemplates:**Retrieves all available active DOCX or PPTX document templates using ExtractDocumentTemplateByTemplateType DataRaptor interface. * **PickTemplate:** Displays all the extracted templates in the selectable template. * **GenerationOptions:**Select the file format and other document generating options like ServiceFunction, the title for the generated document, DocumentContentVersionId. * **SetGenerationServiceInputParams:**Sets key-value pairs to be passed to one of the subsequent document generation steps. * **ServerSideDocGeneration:**Performs document generation if ServiceFunction is set to Server Side Document Generation. * **ServerSideDocGenerationWithTokenData:**Performs document generation if ServiceFunction is set to Server Side Document Generation w/ Token Data. * **ServerSidePdfGeneration:**Performs document generation if ServiceFunction is set to Server Side PDF Generation. * **Errorhandling:**Displays an error message if document generation was unsuccessful. * **ResponseMessage:**Displays a JobId if the document generation job is completed successfully. |

# Sample OmniStudio Integration Procedures

Sample OmniStudio Integration Procedures are provided for quick implementation. Each sample invokes different methods of Apex classes and generates a Microsoft Word, Microsoft PowerPoint, or PDF document. The Integration Procedures can also convert previously generated .docx or .pptx files to PDF. The output document format depends on the input parameters specified in the Integration Procedure.

The sample OmniScript (docGenerationSample/singleDocxServersideLwc) calls the sample Integration Procedures, but you can customize and use them as standalone Integration Procedures. You can use the sample Integration Procedures to create trigger-based document-generation workflows.

DocGeneration Integration Procedure

|  |  |
| --- | --- |
| DocumentServiceGateway_DocGeneration Integration Procedure | * The DocumentServiceGateway\_DocGeneration Integration Procedure is called if the Service Function is set as Server-Side Document Generation * Invokes generateDocument method of DocumentServiceGateway Apex class * .DOCX, .PPTX, or .PDF documents are generated depending on the specified input parameters |

DocGenerationWithTokenDate Integration Procedure

|  |  |
| --- | --- |
| DocumentServiceGateway_DocGenerationWithTokenData Integration Procedure | * DocumentServiceGateway\_DocGenerationWithTokenData Integration Procedure is called if the Service Function is set as Server Side Document Generation w/ Token Data * Invokes generateDocumentWithTokenData method of DocumentServiceGateway Apex class * .DOCX, .PPTX, or .PDF documents are generated depending on the specified input parameters and token data |

DocGenerationPDF Integration Procedure

|  |  |
| --- | --- |
| DocumentServiceGateway_DocGenerationPDF Integration Procedure | * The DocumentServiceGateway\_DocGenerationPDF Integration Procedure is called if the Service Function is set as Server Side PDF Generation * Invokes convertToPDF method of the DocumentServiceGateway Apex class * A .PDF document is generated bases on previously generated .docx or .pptx document |

**Sample Apex Code for Server-Side Document Generation**

You can use the sample code below for creating an Apex class that creates a DocumentGenerationProcess instance.

Update the parameters to use the sample code

* **requestText**: Requires 3 parameters templateContentVersionId, title, keepIntermediate.  
  templateContentversionId is the content version ID of the document template that you want to use.  
  title refers to the title for the generated document. This field is optional.  
  keepIntermediate when set to true, persists the generated intermediate Word document and the final PDF document.
* **type**: Generates a Word document and creates a PDF document.
* **tokenData**: Stores the JSON key-value pair used as an input for server-side document.
* **ReferenceObject**: The object to which the generated document is attached. eg AccountID, QuoteID etc.

|  |
| --- |
| public class TestSFDCDGP {  public static void createDocumentGenerationProcessRequest() {  String requestText = '{\"templateContentVersionId\":\"06801000001tzU5AAI\",\"title\":\"NewDocDemo\",\"keepIntermediate\":true}';  String type = 'GenerateAndConvert';  String tokenData = '{\"contract\_start\_date\":\"2020-07-07\",\"contract\_term\":\"12\",\"contract\_generation\_user\":\"Sriram\",\"contract\_generation\_date\":\"2020-07-07\"}';  DocumentGenerationProcess request = new DocumentGenerationProcess();  request.Type = type;  request.RequestText = requestText;  request.ReferenceObject = '0010100000RhXxBAAV';  request.TokenData = tokenData;  insert request;  }  } |