KSL for Salesforce project setup guide

Sommaire

Terms of use of this guide

The rights to use the software described herein are assigned under a license agreement and this guide may only be used or copied in accordance with the terms of the contract.

The information in this guide is subject to change without notice.

The reproduction or transmission of the information in this guide is limited to internal use by the customer and for the sole purpose of proper use of the software. Any other reproduction or transmission is prohibited without the express written permission of NAELAN.

This guide is provided by NAELAN for information on the software delivered. It does not in any way constitute a contractual commitment both on the features indicated and in their implementation.

Unless otherwise stated, the companies, names and data used in our examples are fictitious ; any reconciliation with real companies or entities would be the result of coincidence.

Typographic conventions

|	
Text file sample	Code example or file settings
(i)	Note, information
Chapter, Guide	Reference to another guide or chapter
Example	Example, variable, code extract
Keyword, label	Keyword, application label, important item

Contact Naelan

Headquarter - 4 rue Claude Chappe
69370 Saint-Didier au Mont d'Or France
+33 4 37 59 81 40
www.naelan.com
support@naelan.com

Office in Paris - 4 Place Louis Armand
75023 Paris France
+33 1 72 76 80 67
contact@naelan.com

1. KSL For Salesforce® for administrators

Select the application KSL Administration from the Salesforce® App Launcher menu.
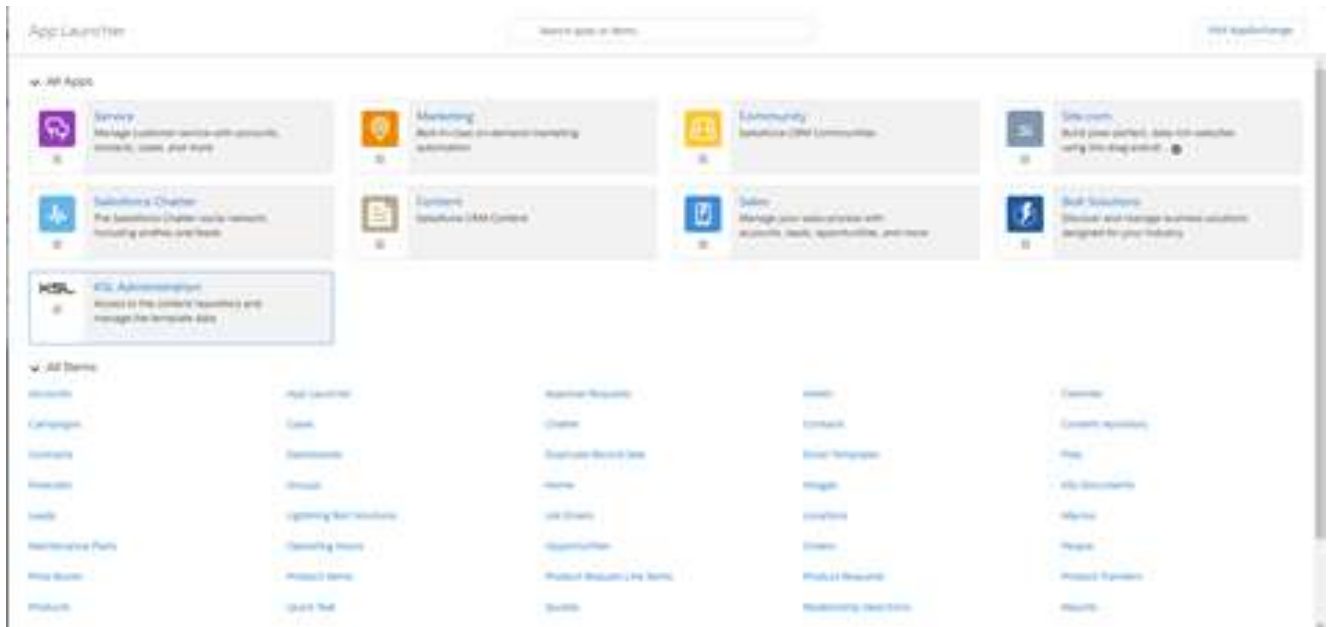
This KSL Administration application only concerns KSL for Salesforce® plug-in et does not response to the same needs as KSL Admin application, which is used to setup and supervise a KSL Server.

KSL Administration includes the following tabs:
- Setup
- Content repository
- Automatic Process Configurations
- Email Configurations
- Execution Histories

1.1. Setup tab

The Setup tab contains the following sub-tabs:
- XSD Generator
- XSD Validator
- Auth. Configuration
- Settings
- Execution Histories Settings

a. XSD Generator

This tab allows the administrator to generate an XSD schema associated with Salesforce® objects. It also allows to generate a test data file to initiate the design of a document or e-mail template.

XSD use

This XSD is used by the KSL Studio design tool to build the document templates based on the data of the selected objects. A template is associated with an XSD. The same XSD can be used by several templates.
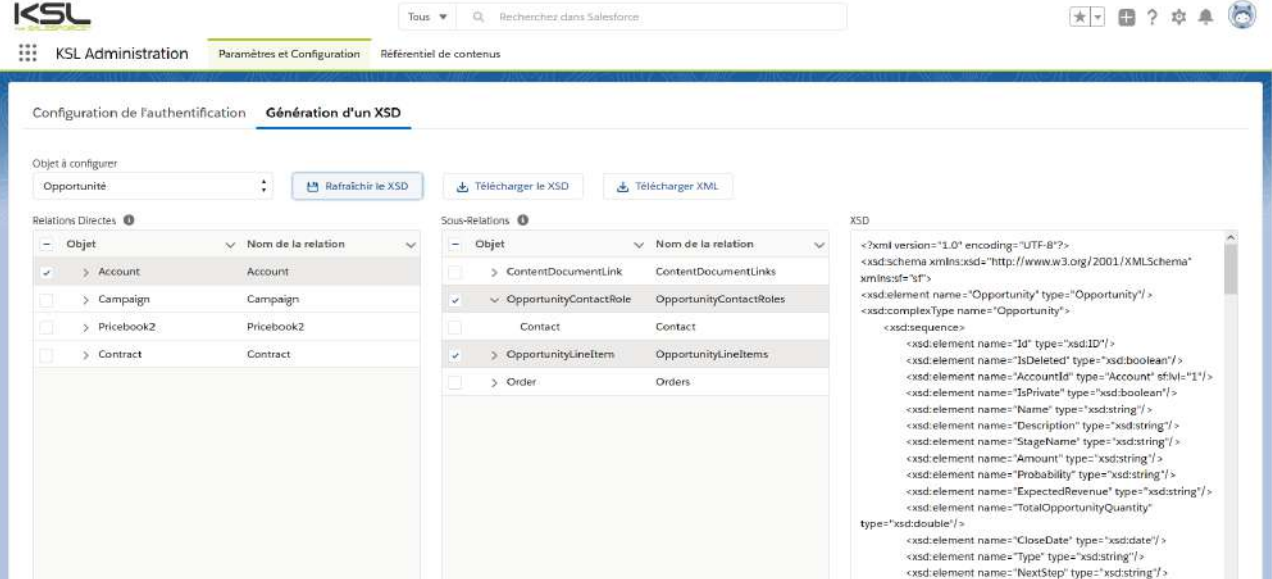
When generating a document, the KSL plug-in asks the KSL server for the XSD schema associated with the selected template; the plug-in then retrieves the necessary data in the Salesforce® objects, and instantly creates a stream in XML format with this data. This XML conforms to the XSD and allows the KSL server to provide the field values to generate the document.

(i) *This exchange makes it possible to detect as soon as possible any discrepancies between the data available in Salesforce® and the data expected by the document template; without this mechanism, deleting for example a Salesforce® field could cause a problem when generating the document, or even generate a document with erroneous contents. In the event of such an offset, an error is displayed in Salesforce® when the document is generated.*

XSD generation

The XSD generator allows to select the main object of the XSD. The builder automatically proposes the direct relationship (Parent) and indirect relationship (Child) objects that exist for this object in the Salesforce® environment. The checkboxes allow to choose the objects to include in the XSD, in relation to the fields necessary in the KSL document templates.

The objects available in the main menu are configured in Metadata Types, detailed in the part to follow.

(**i**) *It is not possible to select the same types of objects more than once in the same relationship tree. For example, if Account is selected, it will not be possible to select the parent Account because they belong to the same type of object in the same tree; KSL document templates will not allow it.*

The XSD Generator tab includes several buttons for generating XSD and XML:

- The Refresh XSD button

Click this button after selecting the objects to insert into your XSD in order for the builder to generate XSD file in the right window. This window contains all the fields of the selected objects.

- The Download XSD button

This button allows downloading the XSD file and using it in KSL Studio to create a document template. It will not work if the Refresh XSD button has not been previously clicked.

When a document template based on this XSD is created, this XSD is linked to the template. With each KSL document generation (from a template), Salesforce® asks for this XSD and uses it to retrieve the values of the Salesforce® object fields described in the schema.

(**i**) Special attention when modifying a data schema:

- If a new field is created in an object used in a KSL document template:
    - If this field has to be used in the document, generate a new XSD, download it, update the XSD component in the KSL repository template and update the KSL template using it.
    - If this field will not be used in the document, no action is necessary. Users will still have the option of generating or personalizing the KSL document in Salesforce® (the new field will be ignored).
    - Note that a new field created in a Saleforce® object must not have the prefix "ksl" which is a forbidden prefix.

- If a field of an object used in a KSL document template, is deleted:
    - Generate and download a new XSD; download it, update the XSD component in the KSL repository template and update the KSL template using it.
    - The field should also be removed from the document layout. If this update is not performed, an error will be returned to the users who will no longer be able to use the document template in Salesforce®.

- If some objects ou lookup fields present in the XSD are not finally associated with the main object (no contacts for an opportunity for example)
    - The XML generated will not be consistent with the XSD, which will cause a document generation error.
    - However, it is possible to make these objects optional by manually modifying the XSD (minOccurs = 0). Note that fields from potentially missing objects will be empty.

- ■ If the User standard object is selected, delete the KSL Role field in the XSD scheme before using it for document or e-mail templates
  - Download the XSD file from KSL Administration application
  - Edit the XSD file with a text editor
  - Delete the ksl__KSL_Role__c field from the schema by deleting the following line <xsd:element name="ksl__KSL_Role__c" type="xsd:string"/>
  - Save the XSD file le fichier XSD with UTF-8 encoding
  - More generally, no XSD field should start with the ksl string

(i) *Note that it is possible to delete some fields of the XSD that would not be necessary for the template in order to reduce the size of the data stream exchanged between Salesforce® and the KSL servers, and also to limit the number of fields accessible from the KSL Studio tool to simplify the selection of fields during template design.*

- ■ The Download XML button

This button is used to generate and download a test data file compliant with the XSD schema. It will not work if you have not previously clicked Refresh XSD.

This button allows to generate and download a test data stream compliant to the XSD schema. This button won't work if the Refresh XSD button has not been previously clicked:

- ■ Each data is surrounded by XML tags
- ■ A text field data is the field name
- ■ A date field data is a random date
- ■ An integer field data is a random integer
- ■ A decimal field data is a random decimal number
- ■ A boolean field data is set to "true" or "false"

Examples:

```
<BillingStreet>BillingStreet</BillingStreet>
<LastActivityDate>2020-09-12 00:00:00</LastActivityDa
<NumberOfWonOpportunities>83</NumberOfWonOpportunities>
<Quantity>58.04</Quan#!{Corps2_Puce}tity>
<IsDeleted>true</IsDeleted>
```

b. Custom Metadata - Allowed Objects

Generating a data schema (.xsd) refers to this interface that lists standard Salesforce® objects by default.

In Setup, look for Metadata Types.

On the "Custom MetadataTypes" page, on the allowed objects line,

Click on Manage records, then Edit.

If you want to use a custom object, add the API of the object to the Custom field. You can also add a Salesforce® standard object to this list if it is not present in the standard list.

Separate each custom object by a carriage return.
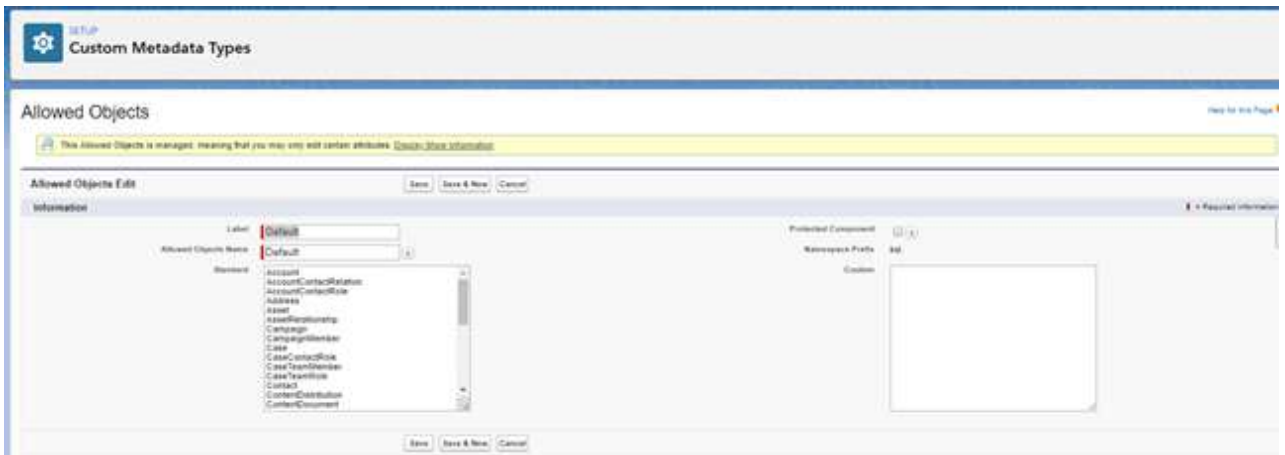
c. XSD schema validation
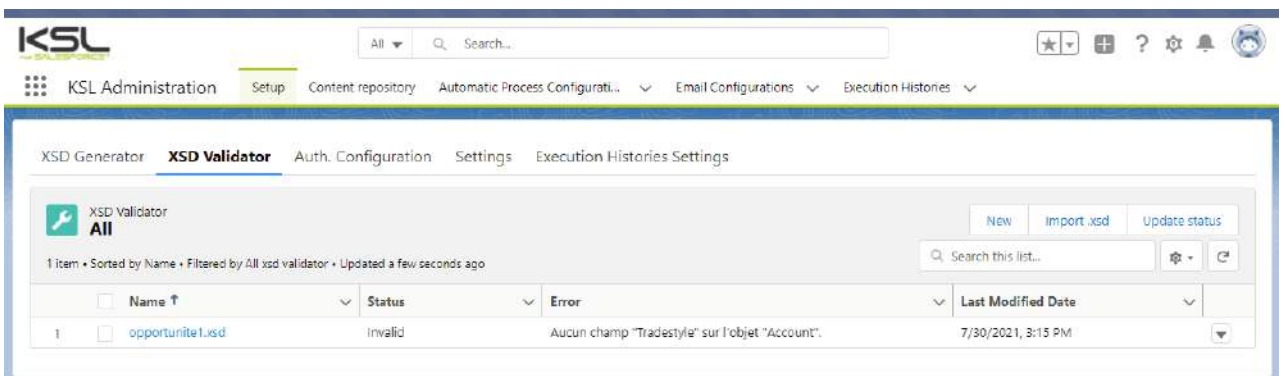
XSD Validator tab is a tool to validate schemas used by KSL documents against the Salesforce organization data. This validation secures a document process by proactively checking that data used by templates still exists in the Salesforce organization.

Some data desynchronization problems can happen in two main cases:
- when a template designer has modified manually an XSD data schema generated by the XSD Generator
- when a Salesforce administrator has removed some fields used by KSL templates

This tool allows the administrator to list the XSD schemas he wants to check by uploading them from his local workstation.

Three buttons are provided in the XSD Validator view: New, Import .xsd and Update status.

New: this button allows to add manually a new schema in the list.
Click the New button and specify the following fields in the window to define the schema:

Field	Description
Name	Name of the XSD schema in the list. This field is mandatory.
Schema	Content of the XSD schema. Copy or enter the code of the schema in this area.
Status	Status of the action. - By default the action is To validate that means that the XSD schema is validated when the schema is added to the list. - The administrator can also force the XSD schema status to Valid or Invalid. In these cases, no validation is applied after saving the schema

Error	This field allows to manually add an error message or note when an Invalid schema is added. It allows a designer to log a schema that needs correction.

Click Save to save the schema. The schema is immediately validated and a detail page is created and displayed with the result of the validation.

Related   **Details**

Name
Opprtunity2.xsd

Status
Invalid

Schema
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:sf="sf">
<xsd:element name="Opportunity"
type="Opportunity"/>
<xsd:complexType name="Opportunity">
<xsd:sequence>
<xsd:element name="Id" type="xsd:ID"/>
<xsd:element name="IsDeleted" type="xsd:boolean"/>
<xsd:element name="AccountId" type="Account"
sf:lvl="1"/>
<xsd:element name="IsPrivate" type="xsd:boolean"/>
<xsd:element name="Name" type="xsd:string"/>
<xsd:element name="Description" type="xsd:string"/>
<xsd:element name="StageName" type="xsd:string"/>
<xsd:element name="Amount" type="xsd:string"/>
<xsd:element name="Probability" type="xsd:string"/>
<xsd:element name="ExpectedRevenue"
type="xsd:string"/>
<xsd:element name="TotalOpportunityQuantity"
type="xsd:double"/>

Error
No such column "DunsNumber" on entity "Account".

(i) Note that the validation is started in background and the result can be delayed; do not hesitate to refresh the view fo get the validation result with the button ⟳ . If the file is invalid, the exact error is reported: usually a field present in an xsd but absent in Salesforce; in this case, edit the xsd, correct its content and test it again by setting the status to To Validate.

Import .xsd: this button allows to add a new XSD schema file to the list.
Click the Import .xsd button; an Upload Files button allows to select an XSD file of the local directory of the administrator's workstation and upload it in the view.

Upload .xsd files

⬆ Upload Files   Or drop files
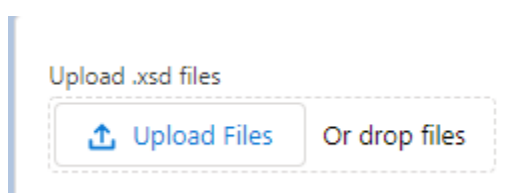
After uploading the file, the following window appears; click Done to complete the download step and return to the list.
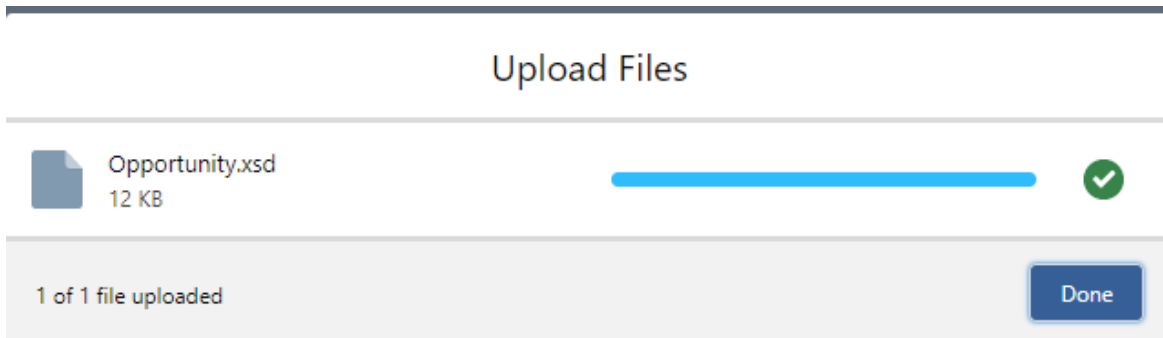
ⓘ File validation is triggered immediately in background; if you don't see the file in the list, click the Refresh button ⟳ .

Drop Files allows to drag and drop the schema from a local directory to the utility. The process remains identical to the one described above.

The schema list view provides the following possibilities:
- Delete a schema that is obsolete, duplicated or in error for example
- Edit the schema and its propertied for its correction
- Update Status of the selected files (multi-selection possible); To validate is proposed by default to launch a new validation on the selection. This status change can potentially be used to exclude obsolete files that the administrator does not want to delete and no longer test, even if they are not in error; in this case, select the status Invalid.

When the status is set to To validate, the validation of the selected schemas starts in background. Click on the Refresh button ⟳ to update the list and the actual status of the schemas.

1.2. Content repository tab

This tab opens in KSL Office on a new tab, the repository containing shared components.
The functions of this application are detailed in KSL Office user guide.

2. KSL for Salesforce® for template designers

2.1. Mandatory KSL functional parameters

When creating new KSL templates, i.e. graphical editions of documents and the associated editing services, and e-mail templates, it is necessary to create the following 3 functional parameters of string type:

- P_FILE: This parameter is used to identify the XML stream transmitted by the plug-in to the editing service. It must be initialized to 250 characters
- P_LABEL: this parameter is used to transmit the description of the document; it corresponds to the field Document description - see chapter 5.1. and 5.2; it can be initialized to 50 characters
- language: This field passes the language used by the user in their Salesforce® application; this field is for example "FR" or "en_US"; we recommend setting it to 10 characters.

2.2. Other KSL functional parameters

Other KSL functional parameters can be added for use in the Document parameters view of the interactive document or email (for example, to allow the user to select a table of contents type or to select optional annexes).

These other parameters can not be valuated by Salesforce® through parameters; to meet this need, additional data will have to be added to the transmitted XML stream.

2.3. Considerations on Salesforce® data sent to KSL

Document and email generation with KSL relies on XML data conformed to an XSD schema.

Examples of XSD schemas and XML streams can be generated using the KSL Administration application. On this issue, KSL Administration facilitates the work of the KSL template designer.

📖 See chapter KSL For Salesforce® for Administrators for more information.

2.4. Use of XML data in KSL

Before creating templates, be sure to set some of the following Advanced run parameters options in KSL Studio:

- XML file decimal character
- XML file thousands separator
- Date format in XML file

📖 See also the KSL Studio User Guide for more details.

This setting must also be reported in the project on KSL Server. Please contact the Naelan teams for this operation.

2.5. Create a document template

This chapter describes step-by-step the process of creation of a document template that will be used to generate a document from Salesforce. This description uses the example on an invoice.

This document is generated from the data of a Salesforce opportunity. Any other Salesforce object, standard or *custom*, could have been used.
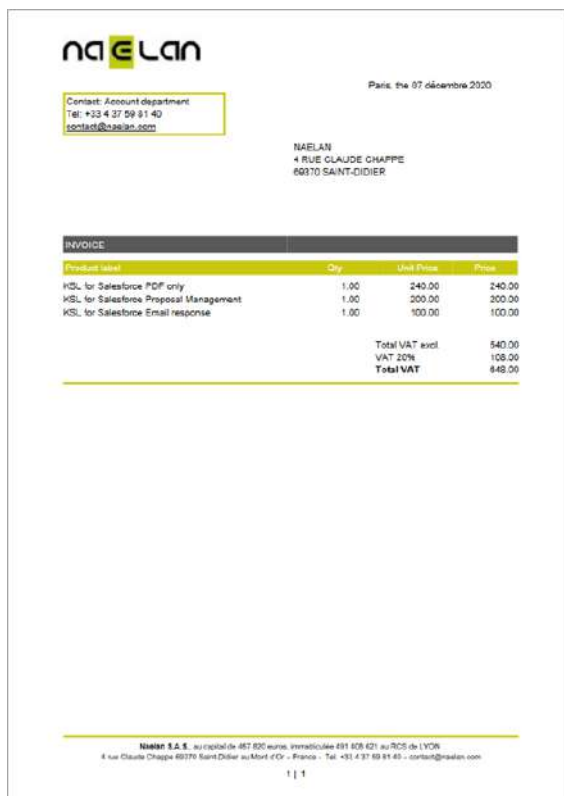
This document is composed of:
- *a document header with a logo*
- *a line displaying the invoice location and date*
- *a contact block with the sender's informations; this block is surrounded by a green frame*
- *an address block with the client's informations*
- *a dynamic table composed of:*
- *2 headlines (maroon and green) repeated on each page if the invoice contains several pages*
- *several article lines: each line corresponds to a product*
- *3 lines at the bottom of the invoice table defining the total amount of the invoice, the VAT amount and the total amount with VAT*
- *a page footer including the company details, the page number and the number of pages*

In this example, all the data come from Salesforce.

a. Step 1: Identify the personalized data of the template

The first step for modeling a document consists in:
- Checking that all the personalized data are available in Salesforce and identify precisely the objects, the fields and relationships between objects.
- Creating some data in an organization to test the document templates.

📖 Refer to Salesforce documentation to check these requirements. If any doubt, consult the Salesforce administrator or data scientist of the project.

In the example, the invoice uses data from a Sales Cloud organization. The objects used are : Opportunity, Products and Pricebook.

📖 *To implement this example, refer if necessary, to the Salesforce documentation explaining the links between products, prices, quotes, and orders.*

The key steps are given below:

The content of this document is the property of Naelan and cannot be used or reproduced without a written authorization.

16 | 49

1. Add Products and Price Books tabs in the Sales Cloud application

■ *To change the navigation bar and add the Products and Price Books objects to it, click on the pencil.*
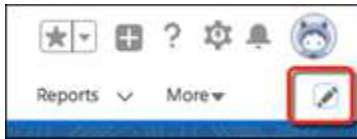
■ *Add the Products and Price Books objects by clicking Add More Items. Once selected, items can be reordered or removed before saving changes*
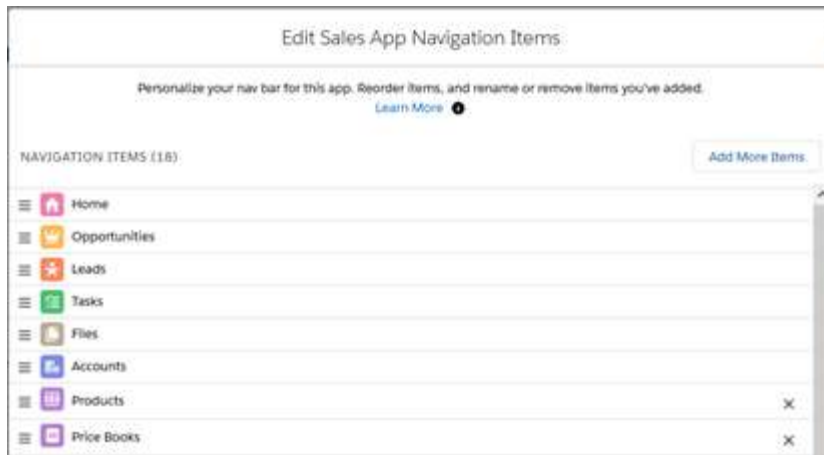
2. Create a new product family
■ *From Salesforce Setup, click Object Manager*
■ *Select the Product and click Fields and Relationships*
■ *Select Product Family*
■ *Under the Product Selection List: select the Product Family Picklist Values, click New*
■ *Add a value (the name of a family) and click Save*

3. Add new products

■ *Click* ⠿ *to open the App Launcher and select Products*
■ *Click New and enter product details*
■ *Click Save and New*
■ *Create a new product*
■ *Etc.*

4. Create standard price book
■ *From the Products object, select a product*
■ *Click the Related tab*
■ *In the Price Books section, click Add Standard Price*
■ *In the Price field, enter a value*
■ *Save*

5. Create a new price book
■ *From the Sales navigation bar, select the item Price Books*
■ *Click New and enter the details of this book*
■ *Save*

6. Add a product to the new price book
- *From the object Price Books, select a book*
- *Click the Related tab*
- *In the Price Book Entries section, click Add Products*

7. Add a product to an opportunity
- *From the Sales navigation bar, select the Opportunities object*
- *Select an opportunity*
- *From the Related tab and in the Products section click* ▼ *to add products*

An opportunity "NAELAN DEAL" is also created; this opportunity includes 3 KSL products defined by their designation, quantity and unit price.
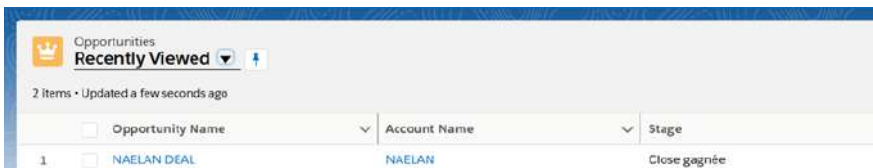
Figure above: creation of the opportunity NAELAN DEAL.
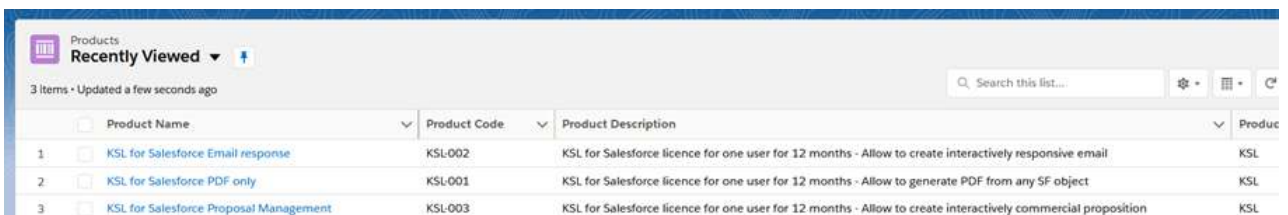
Figure above: creation of 3 products.
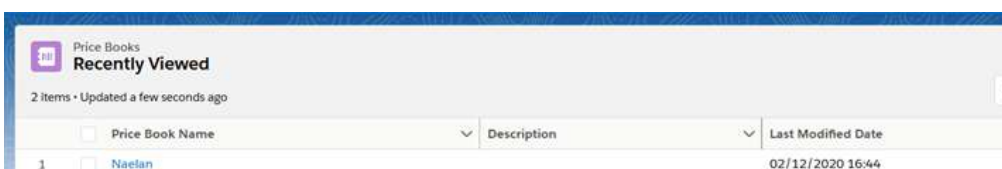
Figure above: creation of a price book.

Figure above: 3 products have been added to the opportunity.

Finally, an invoice requires a total amount, as well as amounts with VAT. These amounts could be calculated by KSL, but it is always better to rely on data from Salesforce.

For doing it, create 2 calculated fields in Salesforce on the opportunity object :

- *VAT_Amount : field equal to the value of the Amount field x 0,2 (20%)*
- *Amount VAT Included : field equal to the value of the Amount field x 1,2 (Amount +20%)*

(i) These two fields, like the amount field, are refreshed when the opportunity is changed.

b. Step 2: Define a KSL schema of the document data

Salesforce data are transmitted to a KSL document template in a format conformed to a XSD schema. This schema defines the format of Salesforce data that can be used by the template. This schema is automatically generated from the KSL Administration application.

(i) KSL document templates does not require SOQL queries. This characteristic provides a greater simplicity in the design of document templates, especially if many similar templates need to be created and share the same data fields.

To define the data schema of a template, perform the following actions:

- Launch the KSL Administration application from the Salesforce menu ⣿ , then select the main object and the necessary sub-objects.
- Click on the Refresh XSD button to view the data schema.
- Click on the Download XSD button to download the KSL data schema file. This XSD schema will be embedded in the template and will allow to query the data necessary when generating the document.
- Click on the Download sample XML button to download an XML data sample file. This file will be used for testing the template in KSL Studio.

- Connect to KSL Studio, drag and drop the two XML and XSD file on the repository tree of the KSL project and select the classification plan for these files.

(i) This step is performed only once for all templates that use the same objects. This factorization of the Salesforce data collection allows to optimize the modeling and maintenance tasks applied on a same family of templates.

In the invoice example, the opportunity object is selected, then the Account and Pricebook2 sub-objects, in relation to the OpportunityLineItem and product2 objects.

Figure above: selection of the opportunity object and sub-objects in the KSL Administration application, then download of the XSD and XML files and renaming of the files to Invoice.xsd and Invoice.xml.

(i) The XSD schema generated from KSL Administration has the following structure:

```xml
 1    <?xml version="1.0" encoding="UTF-8"?>
 2    <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:sf="sf">
 3     <xsd:element name="Opportunity" type="Opportunity"/>
 4    <xsd:complexType name="Opportunity">
 5      <xsd:sequence>
 6         <xsd:element name="Id" type="xsd:ID"/>
 7         ...
 8         <xsd:element name="Pricebook2Id" type="Pricebook2" sf:lvl="1"/>
 9         <xsd:element name="AmountVAT__c" type="xsd:string"/>
10         <xsd:element name="Amount_VAT_Included__c" type="xsd:string"/>
11         <xsd:element name="OpportunityLineItem" type="OpportunityLineItem" maxOccurs="unbounded"
12            minOccurs="0" sf:relationName="OpportunityLineItems" sf:lvl="1"/>
13       </xsd:sequence>
14    </xsd:complexType>
15    <xsd:complexType name="OpportunityLineItem">
16      <xsd:sequence>
17         <xsd:element name="Id" type="xsd:ID"/>
18         <xsd:element name="Name" type="xsd:string"/>
19         <xsd:element name="Quantity" type="xsd:double"/>
20         <xsd:element name="TotalPrice" type="xsd:string"/>
21         <xsd:element name="UnitPrice" type="xsd:string"/>
22         ...
23       </xsd:sequence>
24    </xsd:complexType>
25    <xsd:complexType name="Product2">
26      <xsd:sequence>
27         <xsd:element name="Id" type="xsd:ID"/>
28         ...
29       </xsd:sequence>
30    </xsd:complexType>
31    <xsd:complexType name="Account">
32      <xsd:sequence>
33         <xsd:element name="Id" type="xsd:ID"/>
34         <xsd:element name="BillingStreet" type="xsd:string"/>
35         <xsd:element name="BillingCity" type="xsd:string"/>
36         <xsd:element name="BillingPostalCode" type="xsd:string"/>
37         ...
38       </xsd:sequence>
39    </xsd:complexType>
40    <xsd:complexType name="Pricebook2">
41      <xsd:sequence>
42         <xsd:element name="Id" type="xsd:ID"/>
43         ...
44       </xsd:sequence>
45    </xsd:complexType>
46    </xsd:schema>
```

(i) Note that the XSD schema and the test XML file can be modified before being classified in the KSL repository. Be careful to keep a correct syntax and a consistent structure with Salesforce data.
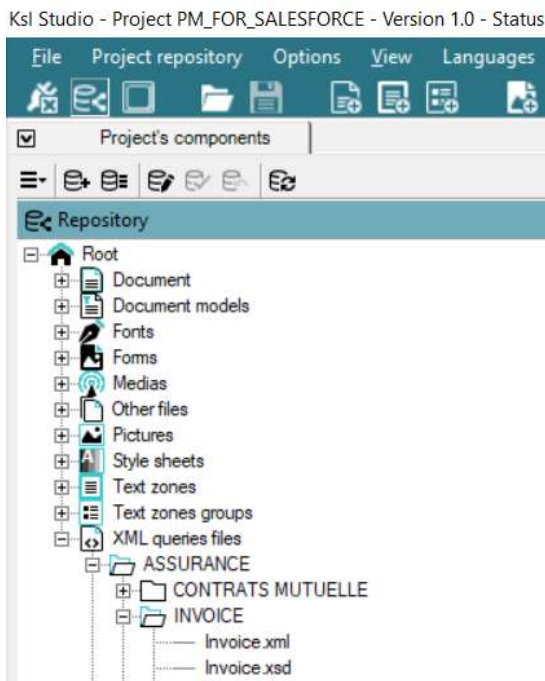
Figure above: classification plan in KSL Studio with the Invoice.xsd and Invoice.xml files.

c. Step 3: Prepare the shared components of the template

This step consists in identifying the components of the template to be designed. Some of the components already exist in the KSL repository and will be reused, others will have to be created.

KSL components include XSD data schemas (see previous chapter), style sheets (colors, fonts, bullet point styles...), headers, footers, sub-templates, images, paragraphs and chapters, external PDF files, page backgrounds, media, etc.

📖 Refer to the KSL Studio Reference Guide for more information about components of a KSL repository.

KSL document design allows to share as much as possible components between templates.

ⓘ Thanks to this particularity, KSL differs strongly from other solutions on the market for which each document template redefines its data, its graphic styles, its headers and paragraphs and its content.

This capability to share components provides many benefits:
- A rapid design of templates by reusing existing components
- An easier maintenance: changing a component updates all templates that use it
- Component management delegation: updating paragraphs, images or stylesheets can be delegated to business managers or to the communications department.

ⓘ KSL components are created and modified, either. from KSL Studio by the template designer or from

the Content Repository tab of KSL Administration application by the Salesforce administrator.

In the example below, invoice components are created using the Content Repository tab of the KSL Administration application.

Figure above: the component design tool of KSL Administration.

Style sheet component

Click on the Style sheet tab, check that a style sheet exists and matches the needs, otherwise create a style sheet for the invoice.

Defines colors of the invoice (white, black, gray and green), the two text styles Arial 11 and Arial 9 (Arial 9 for invoice footer text), the two line styles (black with 1 point thickness and green with 2 points thickness).

📖 *Refer to the KSL Office User Guide for more details the style sheet component settings.*
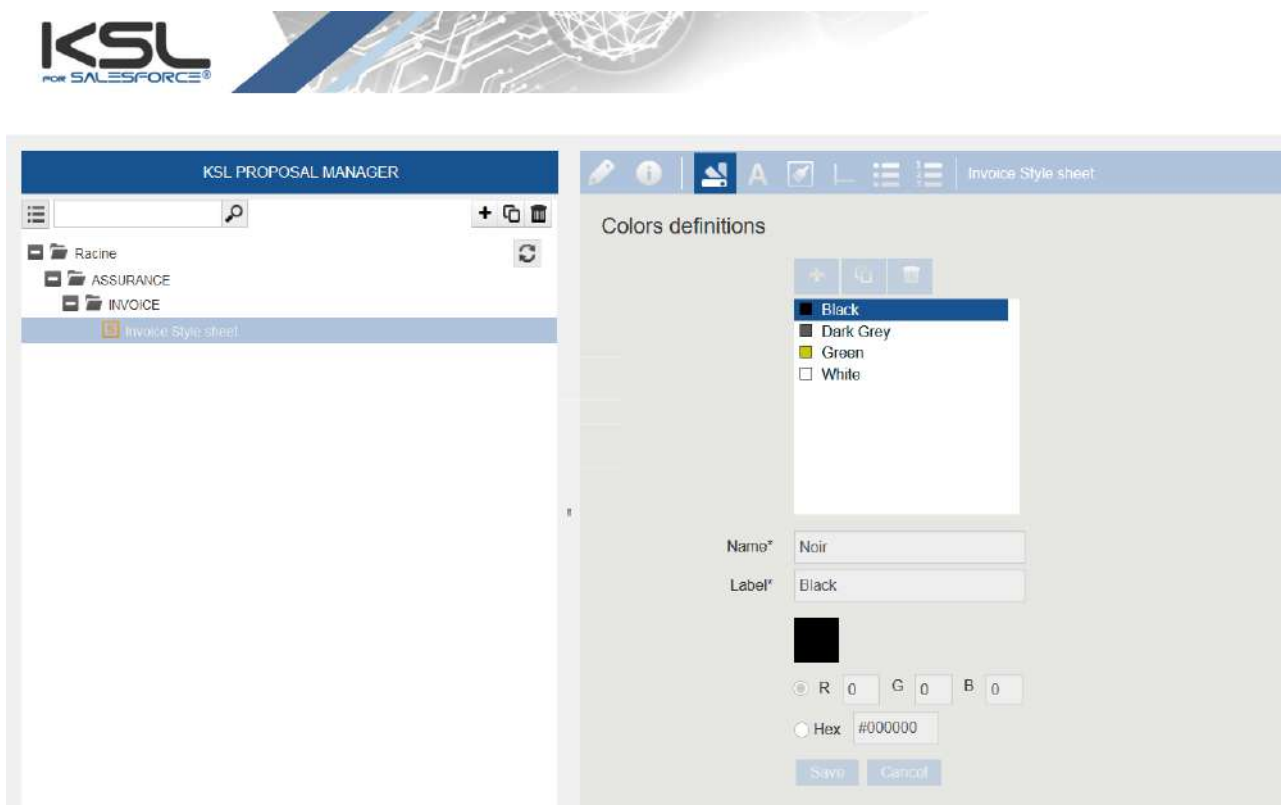
Figure above: creation of the style sheet "Invoice Style Sheet".

Images component

From the Repository tab, add the invoice logo file with the button ⊞ *. In the example below a file .png is classified in the repository under the INVOICE node with the name "Invoice logo".*

Text zone component

From the Repository tab, add the text zones of the invoice with the button ⊞ *. These text zones will be able to be shared by other document templates.*
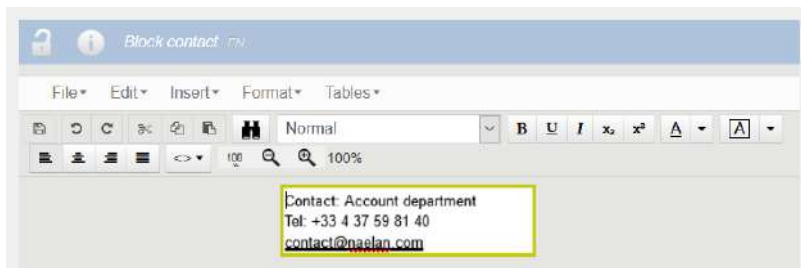
Figure above: creation of the "Block Contact" text zone, which uses the previously created style sheet: the green line style is applied to the borders of the text zone; apply the Arial 11 style to the paragraph text.
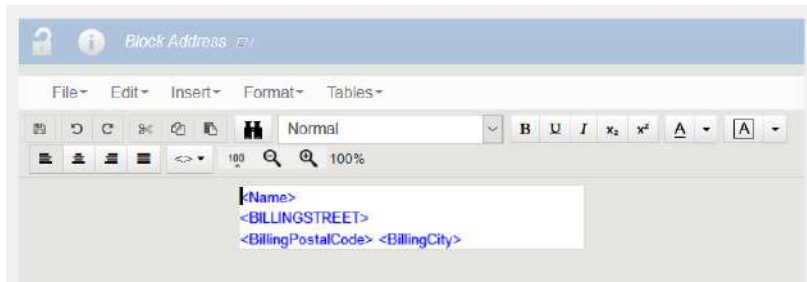
Figure above: creation of the "block Address" text zone, which uses variables; these variables are created by the menu Insert > Add a variable. In this example, they use the same name as the Salesforce fields. These variables correspond to the 4 fields of the Salesforce account object: Name, BillingStreet, BillingPostalCode, and BillingCity.

ⓘ *It is also possible to define a dictionary of variables and link the variables of this dictionary with Salesforce fields. In this case, the variable name that appears in KSL may be different from the name of the Salesforce fields.*

Figure above: creation of the "Table Header" text zone, which includes the header lines of the table of the invoice. This table is created by the Tables menu; this text zone used the style sheet previously created.

ⓘ *A date type variable has been created <CloseDate> from the editor; the value of this variable will be given when creating the document and will have the value of the current date.*
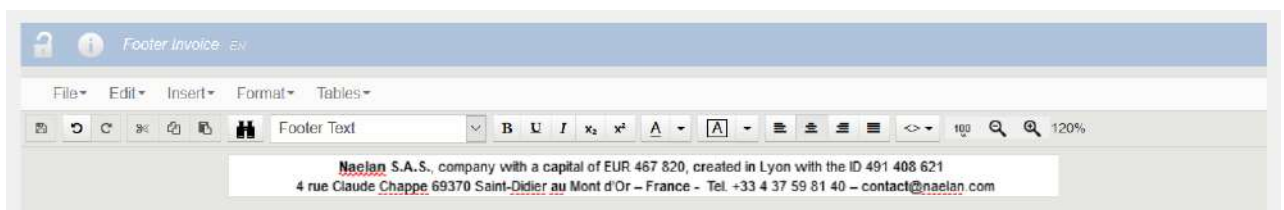
Figure above: creation of the "Footer Invoice" text zone used in the footer of the invoice.

d. Step 4 (optional): Create a document model

In most cases, it is necessary to create several document templates, which share common characteristics: same style sheet, same margins, same headers/footers, same input parameters, same page background, etc.

A document model allows these common characteristics to be grouped together to facilitate the creation and maintenance of a family of document templates. These document models are created with KSL Studio, the KSL document design tool.

A document model can be seen as a metamodel (model on which several document templates are based on). The use of a document model is optional.

In the example, the invoice is based on the "SalesforceModel" document model created and classified in the KSL repository. This document model predefines the following items:

- *A global header including the "Logo Invoice" image previously created*
- *A global footer including the text zone "Footer Invoice" previously created*
- *A page setup with margins: 2 cm at the left and at the right of the page*
- *The style sheet "Invoice Style Sheet" previously created*
- *The functional parameters necessary to the KSL for Salesforce connector i.e. P_FILE, P_LABEL and language: these parameters are common to all document and email templates in a Salesforce organization and allow respectively to transmit to the document template : the XML data stream, the name of the document created by Salesforce, and the language of the Salesforce user.*
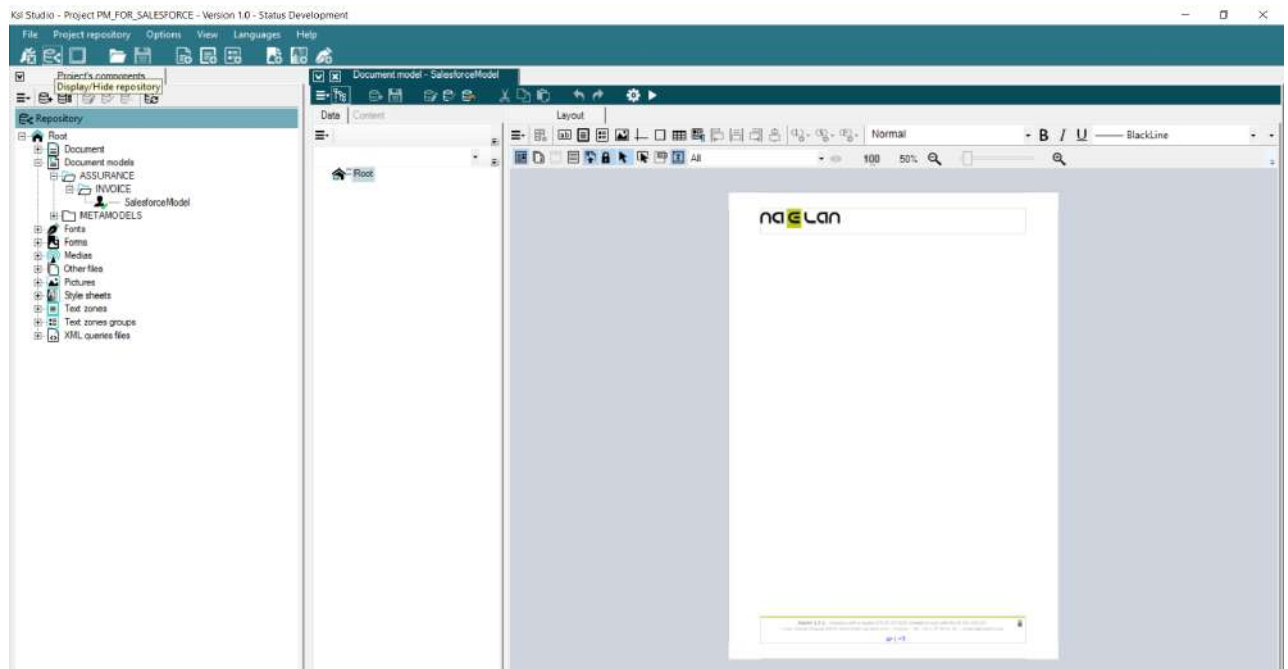
e. Step 5: Create a document template

A document template is called Document in KSL. A KSL Document can be based on a Document model

(see previous chapter). If no model exists in the repository, create it by following the Step 3 above.

In KSL Studio :

■ Connect to the KSL project.

■ Create a document with the button ⊞ and specify if necessary a document model (metamodel) - In the invoice example, *the invoice document is called MyInvoice and is based on the SalesforceModel document model; the document inherits all the characteristics of the document model.*

■ Classify this document in the repository tree with the button ⊟.
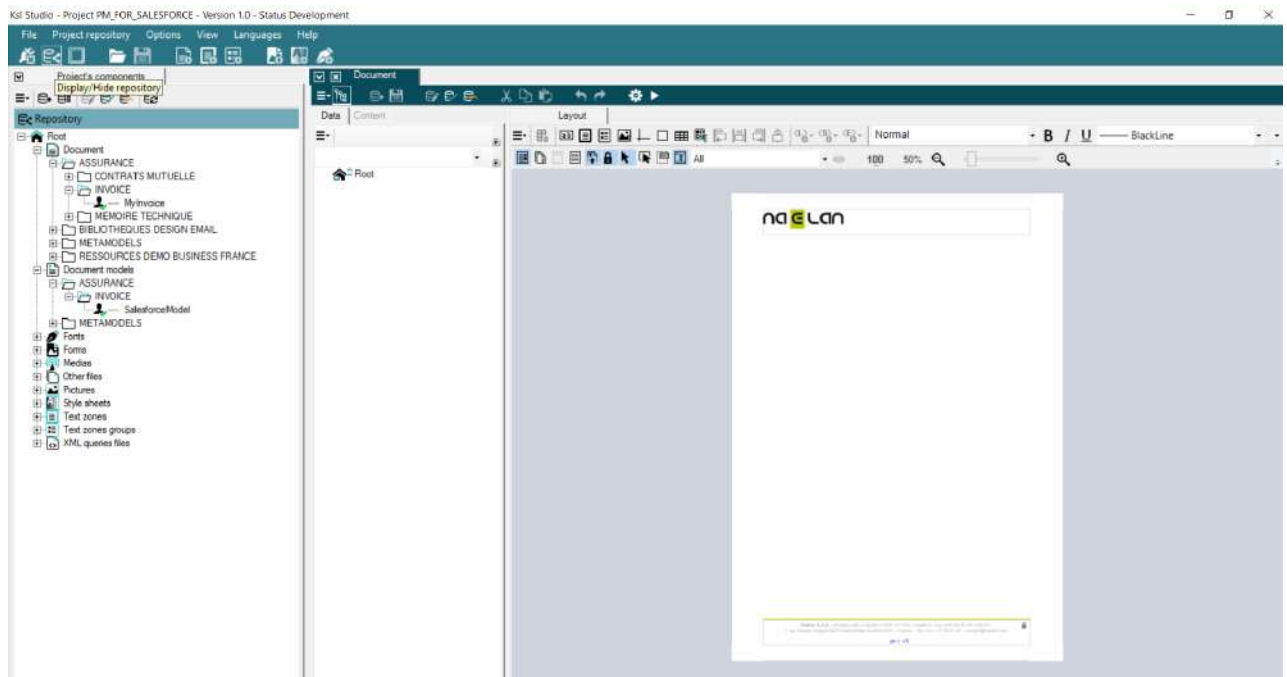
Figure above: creation of the invoice document, based on SalesforceModel document model.

The layout design of the document can start.

Define the Salesforce data schema used by the document

In the document Data view tab,

■ Select the root node 🏠, then right-click and select Add Query.

■ Give a name to this query (in the example *MyDataRequest*).

■ Select the XML query type.
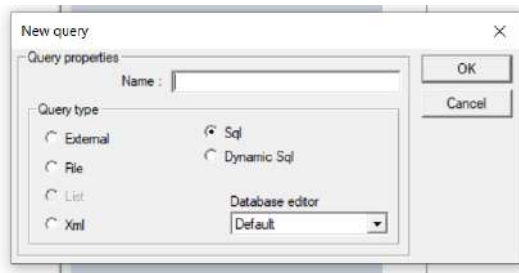
- Select the DTD or schema, by clicking on the button Search repository... and selecting the XSD schema XSD previously created (in the invoice example the file is *Invoice.xsd*).
- Select the Dynamic XML file name (for execution), by clicking on the button Ksl Assistant... and selecting the P_FILE parameter.
- Specify that root element of the query (the *Opportunity* element in the invoice example).
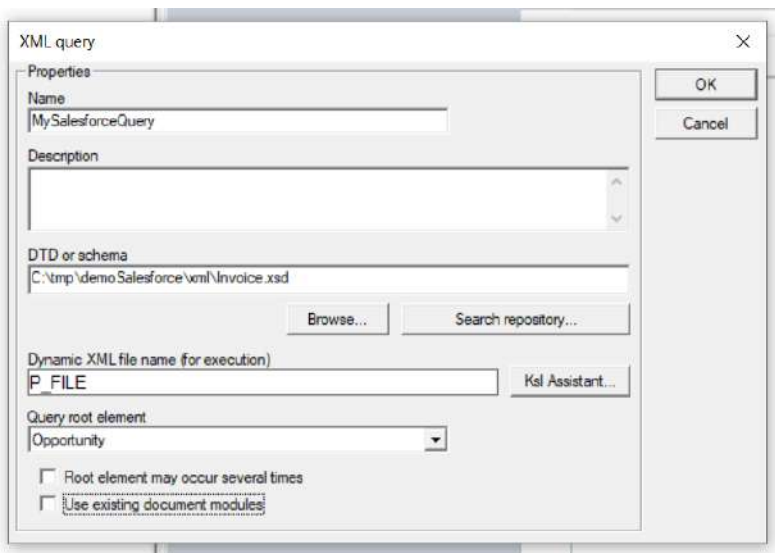
Figure above: creation of a query based on the XSD schema Invoice.xsd.

- Décocher les options L'élément racine peut être présent plusieurs fois et Utiliser les modules d'édition existants.
- Uncheck the Root element may occur several times and Use existing document modules options.
- Validate the query settings with the OK button; the XSD schema structure is displayed under the root node.
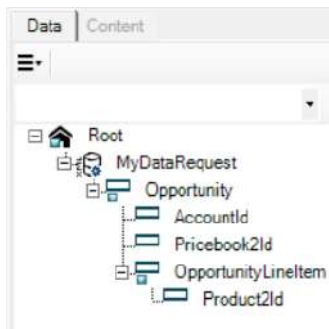
Figure above: query example on Salesforce data. In this example, the fields of the Opportunity object could be used, as well as the fields of the Account, Products, and PriceBook sub-objects.

ⓘ *Note that the Opportunity and OpportunityLineItem elements have by default a display area on the page layout (indicator 🖵). The display area of the Opportunity element corresponds to the body of the invoice; the display are of the OpportunityLineItem element corresponds to the row of the dynamic multi-row table of the invoice.*

The query definition on Salesforce data is complete and the fields of the Salesforce objects are now available for using them in the document layout.

Design the document layout

Designing the document consists in inserting and position graphically the different components of the document template to create.

- Insertion of text zones containing text, images and variables (fields)
- Direct insertion of a text label, an image, a variable, a QR-Code set by a variable...
- Insertion of a sub-document template
- Insertion of a page break, a condition allowing to display different contents,
- Etc.

In the example of the invoice, insert in the body of the invoice, the components "Block Address", "Block Contact" and "table Header", just above the OpportunityLineItem display area.
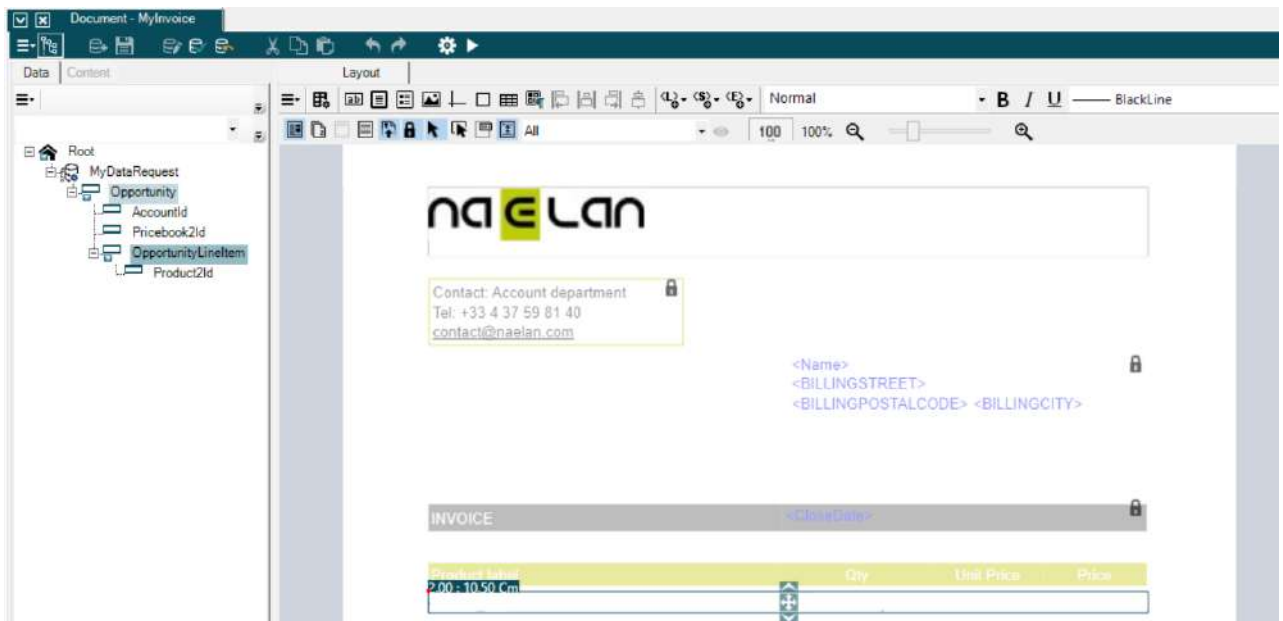
Figure above: the "Block Address", "Block Contact" and "table Header" components appear in the body of the document, below the document header and above the line display area of the invoice table.

Insert a label by clicking the button ⬛ab *with the text "Paris the ", then the system variable Run-time System date and hour with the format "%dd %MM %Y".*

Insert the following variables in the display area of the OpportunityLineItem element:

- *<Name_4>: element "Name" of Product2 (several field with the name "Name" exist in the XSD schema; KSL numbered each element to distinguish them)*
- *<Quantity>: element of Product2*
- *<UnitPrice>: element of Product2*
- *<TotalPrice>: element of Product2*

Figure above: line of a product; this line is repeated for each product of the opportunity. In this context, the repetition of the line is automatic and data-driven.

Insert the 3 variables for the total amounts of the invoice after product lines: variables <Amount>, <AmountVAT__c> and <Amount_VAT_Included__c>, as well as their labels.

Insert a green line at the bottom of the table.

Figure above: table footer with total amounts.

Update and check in the document with the button *to update the KSL repository. The design of the invoice is now finished.*

f. Step 6: Deliver the document template to Salesforce users

After creating a Document (document template), the repository administrator must perform one last operation for users to access the template and generate a document.

This operation consists of creating a Document service from the Document: only document that will be used by Salesforce users needs this operation.

- Launch KSL Admin (the technical administration interface: not to be confused with KSL Administration).
- Log in with a user with project administration permissions.
- Click on the menu Services > Create document services.
- Select the document.
- Define the repository node where the document template will be available to Salesforce users.

■ Click on the Create service button.
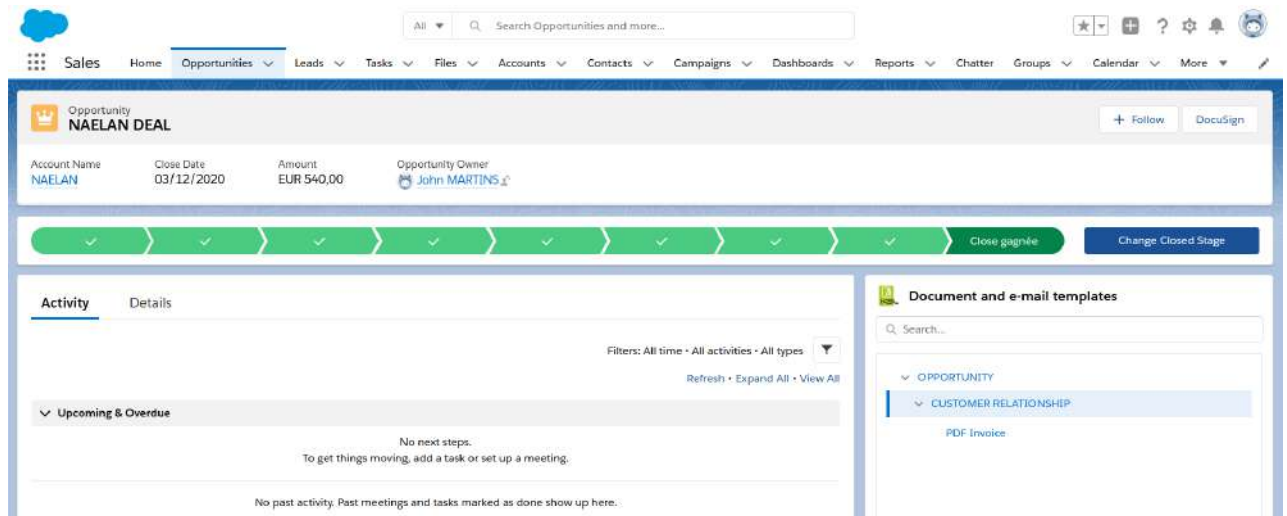
Figure above: in the invoice example, users can now select the invoice template from the Document and email template view of the opportunity layout.

g. Time necessary to design a document template in KSL

Designing a simple document template (for example : the invoice used in this chapter) and delivering it to Salesforce users takes less than 30 minutes.

The time to design additional document templates, close to an existing template, is usually halved (about fifteen minutes to create another template).

The modification of a model is also very fast and in most of the cases, applied without modifying any document template, thanks to the Content Repository tab of KSL Administration.

Please call NAELAN if you need an estimation of the development times necessary for creating templates or get a standard estimation table.

2.6. Create on-demand email template

To create on-demand e-mail templates, proceed as follows. This process is explained using the object Opportunity but any other object can be used.

(i) The following chapter is dedicated to the particular case of e-mailings.

a. Step 1: Generate the XSD and XML streams corresponding to an e-mail

In Ksl Administration, generate and download the XSD schema corresponding to the data personalizing the e-mail (and if necessary an associated XML test data file). This XSD schema will be used by the KSL service(s) in charge of creating the HTML bodies of e-mails.

Example: suppose that the body of the emails to be generated has to include the data corresponding to the Opportunity name and amount, but also lastname, the firstname of the first Contact Roles of the opportunity.

In this context, it is necessary to generate the XSD and XML files from the Opportunity object which contains the Name and Amount elements, but also the elements LastName, FirstName and Salutation of the OpportunityContactRole node of the opportunity.

Example of XML data corresponding to this XSD schema:

```
<?xml version="1.0" encoding="UTF-8"?>
<Opportunity>
  <Name>My opportunity</Name>
  <Amount>10000,00</Amount>
  ...
  <OpportunityContactRole>
    <ContactId>
        <LastName>MARTIN</LastName>
        <FirstName>John</FirstName>
        <Salutation>Mr</Salutation>
    ...
```

b. Step 2: create a mapping file

The mapping file is used to create variables inserted in the body of an e-mail:

- These variables can be selected and inserted by the designer or the content administrator in the body of each e-mail template within text zones.
- These variables can be selected and inserted by the Salesforce user when he edits the body of his e-mail
- These variables are automatically substituted during the e-mail body generation with data from the XML stream coming from Salesforce

- Create a mapping file with a .map extension:
    - The mapping file is a text file
    - It defines each field used by an e-mail (field used in the subject, the body of the e-mail, or the recipient fields if it is an e-mailing)
    - Each field is defined by a line of the file

Each line is composed of 3 strings separated by the character ";"

■ The 1st string is the name of the field that will be used in the e-mail: if used in the template, this field will be the one displayed in the interactive personalization interface, otherwise the end user will be able to insert it in a text zone of the e-mail in this interface

■ The 2nd string is the element of the XML stream that contains the value to display (single-child node)

■ The 3rd string is the element of the XML, which is the of the element defined just above, that contains the value to display

Example with the <SF_Email_Mapping.map> mapping file:

```
OPPORTUNITY_NAME;Name;Opportunity;
OPPORTUNITY_AMOUNT;Amount;Opportunity;
CONTACT_LASTNAME;LastName;ContactId;
CONTACT_FIRSTNAME;FirstName;ContactId;
CONTACT_SALUTATION;Salutation;ContactId;
```

Example:

The XML stream and the mapping file given above will allow to automatically substitute the e-mail variables with the Salesforce data with the following result:

- *OPPORTUNITY_NAME = My opportunity*
- *OPPORTUNITY_AMOUNT = 10000,00*
- *CONTACT_LASTNAME = MARTIN*
- *CONTACT_FIRSTNAME = John*
- *CONTACT_SALUTATION = Mr*

Important notes:

- *Note that only the parent node of each element is to be defined*
- *It is possible to define several fields that point to the same element*
- *If one of the elements is iterative, the field takes the value of the element of the first iteration*
- *Each line ends with a line break (including the last line)*
- *In the case where the parent element exists several times in the flow, it is possible to specify the parent node of the parent node (example: SALUTATION_CONTACT; Salutation; OpportunityContactRole.ContactId;)*

■ From the KSL Administration application, classify this mapping file in the KSL repository:

■ Select the Content repository tab

- Add the mapping file as an external file and classify it in the tree structure

c. Step 3: Create a style sheet for the new e-mail template

If a style sheet already exists, go directly to the next step.

Otherwise, from KSL Studio (or KSL Office from KSL Server version 8.0.2.0), the FS_SALESFORCE_EMAIL

style sheet can be duplicated and modify to match the graphic rules.

d. Step 4: Create the body of the e-mail

From the Content repository tab of the KSL Administration application, create a group of text zones. This group will be the body of the e-mail template.

This group of text zones contains one or more text zones that are also created from the Content Repository application. These text zones must use the style sheet created in the previous step 3.

The variables correspond to the Salesforce data are created in the text zone that require it.
- The value of each variable comes from the XML stream sent by Salesforce
- Each variable is in string format
- The name of each variable corresponds to the 1st element of the line of the mapping file that defines it

For the example described above, the following variables will be created:
- *OPPORTUNITY_NAME*
- *OPPORTUNITY_AMOUNT*
- *CONTACT_LASTNAME*
- *CONTACT_FIRSTNAME*
- *CONTACT_SALUTATION*

e. Step 5: Create a new KSL document template for the e-mail template

In this context of creating a new e-mail template, it is necessary to create a new KSL document template from which a new KSL Server service will be created in step 6.

The procedure from KSL Studio is as follows:
- Add the XSD file generated in step 1 to the KSL project classification plan
- Duplicate the generic edition SF_Email_500px and add it to the KSL project classification tree
- In this new document template, select the style sheet by selecting the style sheet component created in step 3 above
- Modify the XML request for this document template by specifying the new XSD
- Delete the group in the document template
- Insert the new text zone group created in step 4
- Save and Update and release the document template

f. Step 6: Create the document service corresponding to the email template

- From KSL Admin, create the document service associated with the document template created in step 5

- Set a label in the description field: this description corresponds to the label displayed for this e-mail template in the list of templates presented to the Salesforce user in its Document Templates component.
- Classify this service in the project tree: the document service should be classified in a branch of the tree accessible to the Salesforce user in its Document Templates component accessible from the Opportunity object.
- Specify the preview action group: EmailActions
- Fill in the parameters of this document service using the information below

Parameter	Default value or example	Description
feuillestyles (stylesheet)	FS_SALESFORCE_EMAIL	Select the style sheet used for editing (without extension) Deselect the option "Show this parameter in Ksl Office"
MailAccount		*Parameter not used in the case of a single on demand e-mail sent by Salesforce®* Deselect the option "Show this parameter in Ksl Office"
MailBat	False	Set this parameter to False for this single on-demand e-mail sending case Deselect the option "Show this parameter in Ksl Office"
MailCharset	UTF-8	Set this parameter to UTF-8 *Encoding of the CSV file generated from the XML stream sent by the KSL plug-in* *Set this parameter by default to the universal UTF-8 code page; it is possible to change this encoding (example: UTF-8, Windows-1252, ISO8859-15 (Linux)* Deselect the option "Show this parameter in Ksl Office"
MailContentGroup	Example: \<SF_Email_ContractConfirm.ksg\>	*Optional parameter for transmitting the group of text fields used as e-mail body* *If not specified, the group inserted by default in the document template is used* *This parameter requires that the document template dynamically calls the group (default group dynamically called with the expression MailContentGroup)* *The group is defined between the characters < and > with the extension .ksg* Deselect the option "Show this parameter in Ksl Office"
MailCsvFile	\<default\>	This file is used only if the P_MAP_FILE file is not defined. The default value can be used for debugging tests without XML stream. A value between the characters < and > indicates that the file is in the KSL repository.

		The CSV file is a text file with one contact / recipient per line, possibly with a header line that names the columns and column separators ";"
		Set a default value (for example "<default>") even if the P_MAP_FILE is defined
		Deselect the option "Show this parameter in Ksl Office"
MailCsvHeader	Correspond to the header line of the CSV file without #	*Parameter not used in the case of a single on demand e-mail sent by Salesforce®*
		This CSV is only used if P_MAP_FILE is not defined.
		Deselect the option "Show this parameter in Ksl Office"
MailLinkFile	<emailing_links.csv>	Set if necessary the file giving the definition of hypertext links in the e-mail (see documentation on the KSL Email Designer function)
		Deselect the option "Show this parameter in Ksl Office"
MailReturnMode	1	Set this parameter to the value 1.
		Deselect the option "Show this parameter in Ksl Office"
		For information:
		- *1: for the creation of an e-mail sent by Salesforce*
		- *2: for the creation of an e-mailing sent by KSL Server or asynchronously sent by an external e-mail send & track provider*
MailTitle	default	*Parameter not used in the case of a single on demand e-mail sent by Salesforce®*
		This parameter must have a value (for example "default")
		Deselect the option "Show this parameter in Ksl Office"
MailUrl	Example: https://www.naelan.fr/ksl_images/	Set the web directory Url used for storing images (see the KSL Email Designer documentation)
		Deselect the option "Show this parameter in Ksl Office"
P_FILE	Example: <SF_email_Opportunity.xml>	*XML data stream passed by the KSL for Salesforce plug-in*
		This default value is used only during debugging tests
		Deselect the option "Show this parameter in Ksl Office"
P_LABEL		*Parameter not used in the case of a single on demand e-mail sent by Salesforce®*
		Deselect the option "Show this parameter in Ksl Office"
language	Example: FR	Salesforce interface language passed through the KSL plug-in
		Deselect the option "Show this parameter in Ksl Office"
P_MAP_FILE	Example:	Indicate the mapping file created for the template and corresponding to its XSD

| <SF_email_mapping.map> | Deselect the option "Show this parameter in Ksl Office" |

g. Step 7 (optional): create other e-mail templates based on the same data

In many business contexts, it is necessary to create a family of e-mail templates that share the same graphic design and based on the same Salesforce objects and data.

In this case, it may be more efficient to create KSL e-mail services based on the same KSL e-mail template. The procedure is then the following.

- In KSL Studio:
    - Edit the document template created in step 5
    - Reserve the template and edit the properties of the text zone group by right-clicking / Properties
    - In the window that appears, specify the expression MailContentGroup the Expression field of the dynamic group call
    - Validate, save and release the edition

- From the Content repository tab of the KSL Administration application,
    - Create as many groups of text boxes as desired body of e-mail (1 group per model of e-mail)

- In KSL Admin:
    - Recompile the document template above
    - Copy the document service as many times as needed (as much service as the family's email template)
    - Give a description to each copied service with the name of the template in Salesforce
    - Edit the functional parameters of each copied service and set the MailContentGroup parameter specifying the name of the group to call (Service 1> Group 1, Service 2> Group 2, etc.)

Benefits:

Thanks to this setting:
- All e-mail templates share the same document template, the same document service, and the same style sheet
- Creating and editing an e-mail body is done directly from Salesforce
- Adding a template is done in seconds by duplicating a service

Example:

SF_Email_500px service is an example of service shared by the 3 e-mail templates implemented by the following 3 services:
- SF_Email_ContractConfirm_500px
- SF_Email_ContractTermin_500px

■ SF_Email_TrainingConfirm_500px

2.7. Create emailing templates

To create e-mailing templates, proceed as follows. This process is explained using the object Campaign but any other object can be used.

Note: The previous chapter is dedicated to the particular case of on-demand single e-mail.

a. Step 1b: Generate the XSD and XML streams corresponding to the e-mailing

In Ksl Administration, generate and download the XSD schema corresponding to the data personalizing the e-mailing (and if necessary an associated XML test data file). This XSD schema will be used by the KSL service(s) in charge of creating the HTML bodies of the e-mails.

Example:
- *Suppose that the body of the emails to be generated has to include the data corresponding to the fields Salutation, LastName and FirstName of the targeted contacts and the Campaign name.*
- *Moreover, the E-mail address of each targeted contact must also be used by the e- mailing, since the KSL server is responsible for sending, or transmitting e-mails generated to an external send & track provider.*

In this context, it is necessary to generate the XSD and XML files from the Campaign object which contains the Description element, but also from the CampaignMember sub-object which contains the elements LastName, FirstName, Salutation and Email.

Example of XML data corresponding to this XSD schema:

```
<?xml version="1.0" encoding="UTF-8"?>
<Campaign>
  <Description>My emailing</Description>
  ...
  <CampaignMember>
        <LastName>MARTIN</LastName>
        <FirstName>John</FirstName>
        <Salutation>Mr</Salutation>
        <Email>martin@naelan.com</Email>
        ...
  </CampaignMember>
  <CampaignMember>
        <LastName>SIMONS</LastName>
        <FirstName>Marie</FirstName>
        <Salutation>Mrs</Salutation>
        <Email>simons@naelan.com</Email>
        ...
  </CampaignMember>
  ...
```

b. Step 2b: create a mapping file

The mapping file is used to create variables inserted in the body of an e-mailing:

- These variables can be selected and inserted by the designer or the content administrator in the body of each e-mailing template within text zones.
- These variables can be selected and inserted by the Salesforce user when he edits the body of the e-mailing
- These variables are automatically substituted during the e-mail bodies generation with data from the XML stream coming from Salesforce

- In KSL Admin, create a mapping file with a .map extension:
    - The mapping file is a text file
    - It defines each field used by an e-mail (field used in the subject, the body of the e-mail, or the recipient fields if it is an e-mailing)
    - Each field is defined by a line of the file

Each line is composed of 3 strings separated by the character ";"
- *The 1st string is the name of the field that will be used in the e-mail: if used in the template, this field will be the one displayed in the interactive personalization interface, otherwise the end user will be able to insert it in a text zone of the e-mail in this interface*
- *The 2nd string is the element of the XML stream that contains the value to display (single-child node)*
- *The 3rd string is the element of the XML, which is the of the element defined just above, that contains*

the value to display

Example with the <SF_Email_Mapping.map> mapping file:

```
CAMPAGNE_NAME;Description;Campaign;
CONTACT_LASTNAME;LastName;CampaignMember;
CONTACT_FIRSTNAME;FirstName;CampaignMember;
CONTACT_SALUTATION;Salutation;CampaignMember;
Email;Email;CampaignMember;
```

Example:

The XML stream and the mapping file given above will allow you to automatically substitute the e-mail variables with the Salesforce data with the following result:

For the 1st e-mail:
- *CAMPAGNE_NAME = My emailing*
- *CONTACT_LASTNAME = MARTIN*
- *CONTACT_FIRSTNAME = John*
- *CONTACT_SALUTATION = Mr*
- *Email = martin@naelan.com*

Pour le 2nd e-mail:
- *CAMPAGNE_NAME = My emailing*
- *CONTACT_LASTNAME = SIMONS*
- *CONTACT_FIRSTNAME = Marie*
- *CONTACT_SALUTATION = Mrs*
- *Email = simons@naelan.com*

Important notes:
- *Note that only the parent node of each element is to be defined*
- *It is possible to define several fields that point to the same element*
- *If one of the elements is iterative, the field takes the value of the element of the first iteration*
- *Each line ends with a line break (including the last line)*
- *In the case where the parent element exists several times in the flow, it is possible to specify the parent node of the parent node (example: SALUTATION_CONTACT; Salutation; OpportunityContactRole.ContactId;)*

- From the KSL Administration application, classify this mapping file in the KSL repository:
  - Select the Content repository tab
  - Add the mapping file as an external file and classify it in the tree structure

c. Step 3b: Create a style sheet for your new e-mailing template

If you already have a style sheet, go directly to the next step.

Otherwise, from KSL Studio (or KSL Office from KSL Server version 8.0.2.0), you can duplicate the FS_SALESFORCE_EMAIL style sheet and modify it to match your graphic charter.

d. Step 4b: Create the body of your e-mailing

From the Content repository tab of the KSL Administration application, create a group of text zones. This group will be the body of your e-mailing template.

This group of text zones contains one or more text zones that are also created from the Content Repository application. These text zones must use the style sheet created in the previous step 3.

The variables correspond to the Salesforce data are created in the text zone that require it.
- The value of each variable comes from the XML stream sent by Salesforce
- Each variable is in string format
- The name of each variable corresponds to the 1st element of the line of the mapping file that defines it

For the example described above, the following variables will be created:
- *CAMPAGNE_NAME*
- *CONTACT_LASTNAME*
- *CONTACT_FIRSTNAME*
- *CONTACT_SALUTATION*
- *Email*

e. Step 5b: Create a new KSL document template for the e-mailing template

In this context of creating a new e-mailing template, it is necessary to create a new KSL document template from which a new KSL Server service will be created in step 6.

The procedure from KSL Studio is as follows:
- Add the XSD file generated in step 1 to the KSL project classification plan
- Duplicate the generic edition SF_Emailing_500px and add it to the KSL project classification tree
- In this new document template, select the style sheet by selecting the style sheet component created in step 3 above
- Modify the XML request for this document template by specifying the new XSD
- Delete the group in the document template
- Insert the new text zone group created in step 4
- Save and Update and release the document template

f. Step 6b: Create the document service corresponding to the emailing template

- From KSL Admin, create the document service associated with the document template created in step 5
- Set a label in the description field: this description corresponds to the label displayed for this e-mail template in the list of templates presented to the Salesforce user in its Document Templates component.
- Classify this service in the project tree: the document service should be classified in a branch of the tree accessible to the Salesforce user in its Document Templates component accessible from the Campaign object.
- Specify the preview action group: EmailActions
- Fill in the parameters of this document service using the information below

Parameter	Default value or example	Description
feuillestyles (stylesheet)	FS_SALESFORCE_EMAIL	Select the style sheet used for editing (without extension) Deselect the option "Show this parameter in Ksl Office"
MailAccount	MARKETING	Specify the e-mail distribution account if KSL Server is the SMTP server, otherwise left this field blank This distribution account is used for mass e-mailing sending; it must match a distribution account created in KSL Admin Deselect the option "Show this parameter in Ksl Office"
MailBat	True	Set this parameter to True for this single on-demand e-mail sending case Deselect the option "Show this parameter in Ksl Office"
MailCharset	UTF-8	Set this parameter to UTF-8 *Encoding of the CSV file generated from the XML stream sent by the KSL plug-in* *Set this parameter by default to the universal UTF-8 code page; it is possible to change this encoding (example: UTF-8, Windows-1252, ISO8859-15 (Linux)* Deselect the option "Show this parameter in Ksl Office"
MailContentGroup	Example: <InvitationEvenement.ksg>	*Optional parameter for transmitting the group of text fields used as e-mail body* *If not specified, the group inserted by default in the document template is used* *This parameter requires that the document template dynamically calls the group (default group dynamically called with the expression MailContentGroup)* *The group is defined between the characters < and > with the extension .ksg*

		Deselect the option "Show this parameter in Ksl Office"
MailCsvFile	<default>	This file is used only if the P_MAP_FILE file is not defined.
		The default value can be used for debugging tests without XML stream. A value between the characters < and > indicates that the file is in the KSL repository.
		The CSV file is a text file with one contact / recipient per line, possibly with a header line that names the columns and column separators ";"
		You need to set a default value (for example "<default>") even if the P_MAP_FILE is defined
		Deselect the option "Show this parameter in Ksl Office"
MailCsvHeader		*Parameter not used in the case of an e-mailing*
		This CSV is only used if P_MAP_FILE is not defined.
		Deselect the option "Show this parameter in Ksl Office"
MailLinkFile	<emailing_links.csv>	Set if necessary the file giving the definition of hypertext links in the e-mail (see documentation on the KSL Email Designer function)
		Deselect the option "Show this parameter in Ksl Office"
MailReturnMode	2	Set this parameter to the value 2.
		Deselect the option "Show this parameter in Ksl Office"
		For information:
		- *1: for the creation of an e-mail sent by Salesforce*
		- *2: for the creation of an e-mailing sent by KSL Server or asynchronously sent by an external e-mail send & track provider*
MailTitle	Example: <CONTACT_SALUTATION> <CONTACT_LASTNAME> - Visit our trade show	Select a text only if you want the subject of the e-mails to be predefined
		You can enter a free text in and insert some variables defined in the mapping file
		Select or deselect the option "Show this parameter in Ksl Office", depending on whether you want the user to be able to change or not, the subject used when creating e-mailing
MailUrl	Example: https://www.naelan.fr/ksl_images/	Set the web directory Url used for storing images (see the KSL Email Designer documentation)
		Deselect the option "Show this parameter in Ksl Office"
P_FILE	Example: <SF_Emailing_Campaign.xml>	*XML data stream passed by the KSL for Salesforce plug-in*
		This default value is used only during debugging tests
		Deselect the option "Show this parameter in Ksl Office"

P_LABEL		*Parameter not used in the case of a single on demand e-mail sent by Salesforce®*
		Deselect the option "Show this parameter in Ksl Office"
language	Example: FR	Salesforce interface language passed through the KSL plug-in
		Deselect the option "Show this parameter in Ksl Office"
P_MAP_FILE	Example: <SF_emailing_mapping.map>	Indicate the mapping file created for the template and corresponding to its XSD
		Deselect the option "Show this parameter in Ksl Office"

g. Step 7b (optional): create other e-mailing templates based on the same data

In many business contexts, it is necessary to create a family of e-mailing templates that share the same graphic design and based on the same Salesforce objects and data.

In this case, it may be more efficient to create KSL e-mail services based on the same KSL e-mailing template. The procedure is then the following.

- In KSL Studio:
    - Edit the document template created in step 5
    - Reserve the template and edit the properties of the text zone group by right-clicking / Properties
    - In the window that appears, specify the expression MailContentGroup the Expression field of the dynamic group call
    - Validate, save and release the edition

- From the Content repository tab of the KSL Administration application,
    - Create as many groups of text boxes as desired body of e-mailing (1 group per model of e-mailing)

- In KSL Admin:
    - Recompile the document template above
    - Copy the document service as many times as needed (as much service as the family's email template)
    - Give a description to each copied service with the name of the template you want in Salesforce
    - Edit the functional parameters of each copied service and set the MailContentGroup parameter specifying the name of the group to call (Service 1> Group 1, Service 2> Group 2, etc.)

Benefits:

Thanks to this setting:
- All e-mailing templates share the same document template, the same document service, and the same style sheet

- Creating and editing an e-mailing body is done directly from Salesforce
- Adding a template is done in seconds by duplicating a service

2.8. Proposing an e-mail or e-mailing with alternative contents

KSL provides a powerful feature that allows to offer to users e-mail or e-mail templates with alternative content.

Thanks to this function, the user edits his e-mail or his e-mailing and can choose between several alternative text boxes in the KSL editor.

The SF_Emailing_500px service given as an example, implements this function and proposes in a single 4 e-mailings in a single template. The user selects the template "Family of e-mailings" and in a second time can select one of the 4 proposed contents corresponding to the groups:

- SF_InvitationEvenement.ksg
- SF_KSLEmailDesigner.ksg
- SF_NewsletterNaelan.ksg
- SF_Teasing_Solutions.ksg

Refer to the KSL Office User Guide for creating these alternative groups or text boxes.

2.9. Create e-mails in non Latin languages

The KSL solution allows the creation of content and emails based on non-Western character fonts, i.e. based on non-Latin character tables.

Important note: in this context, it is necessary to configure the email encoding of Salesforce users. Otherwise, the characters of the email created by KSL will not be readable by recipients.

To configure the encoding:

- Edit the user settings from the Salesforce configuration interface   Setup
- Search Users (via the Quick search field)
- select the menu Users > Users
- select the Unicode (UTF-8) value of the Email encoding user field
- and save the user

Multi-language content examples:

Последние новости, объявления о специальных акциях и информация о том, как заработать и потратить Мили, ждут Вас на www.naelan.com.

最新情報やプロモーション、マイルの獲得とご利用の方法に関する詳細につきましては、www.naelan.com のサイトをご覧ください。
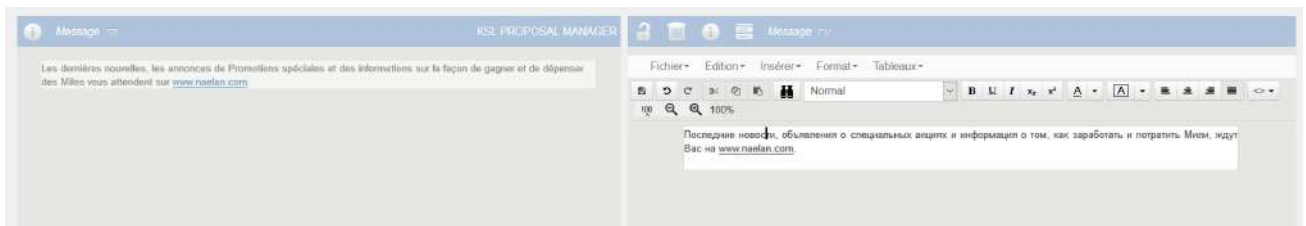
آخر الأخبار والإعلانات من العروض الترويجية الخاصة ومعلومات عن كيفية كسب وقضاء ميل في انتظاركم على www.naelan.com.

Les dernières nouvelles, les annonces de promotions spéciales et les informations sur la façon de gagner et de dépenser des bonus vous attendent sur www.naelan.com.

(i) Note:

- Use the KSL Administration application and its Content repository tab, to create the content in the desired languages.
- Set some variant rules if the content exists in several languages. This setting allows to use the Translations tab of the Content repository tab (see example below). 📖 Refer to KSL email Designer user guide for more information.

The e-mail generated by KSL is sent by Salesforce and is saved in the user's Salesforce activities.

The content of this document is the property of Naelan and cannot be used or reproduced without a written authorization.

48 | 49

NAELAN

Headquarter - 4 rue Claude Chappe
69370 Saint-Didier au Mont d'Or
France - Tél. +33 4 37 59 81 40

Paris Office - 4 Place Louis Armand
75023 Paris - France
Tél. +33 (0)1 72 76 80 86

www.naelan.com
contact@naelan.com
support@naelan.com