What is/are eForms?

Enable Customers and Colleagues to easily provide information to the Bank via a clear, clever user interface (seamlessly integrated with other Bank systems) that efficiently delivers this information to the relevant Bank areas to then fulfil the Customer/Colleague needs.

eForms is the new (strategic, bank wide) smart web forms platform, created centrally by the eForms platform team and propagated out for consumption/leverage by franchises/domains to create/support their own individual use case forms on eForms.

Smart Web Forms provide Customers and Colleagues with electronic data capture, dynamic form sections, real-time data validation, e-submission and integration with Bank services (e.g., e-signatures, ID&V, core systems) to drive automation and reduce cost-to-serve.

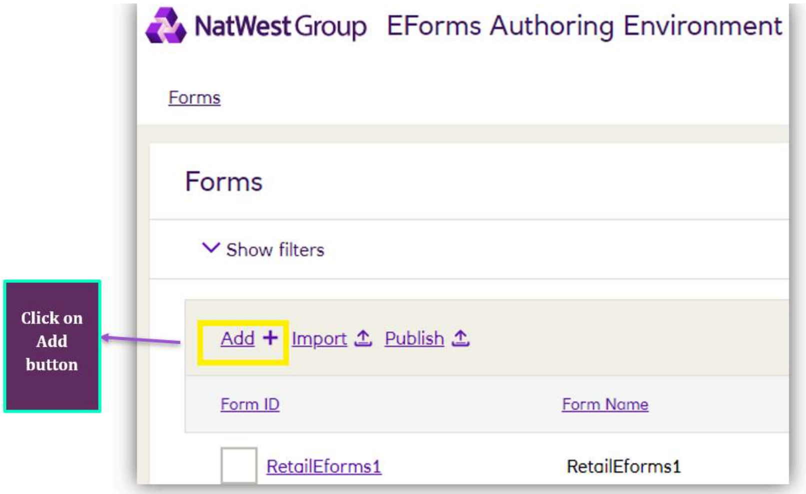
What is e-FORMS Playbook

This playbook is user manual for all the eforms stakeholders. It outlines all eform platform's tribal knowledge and makes it easy so anyone on your team can complete any task. But most importantly, it keeps your running smoothly. It can teach all the stakeholders about how eforms are run and what best practice looks like.

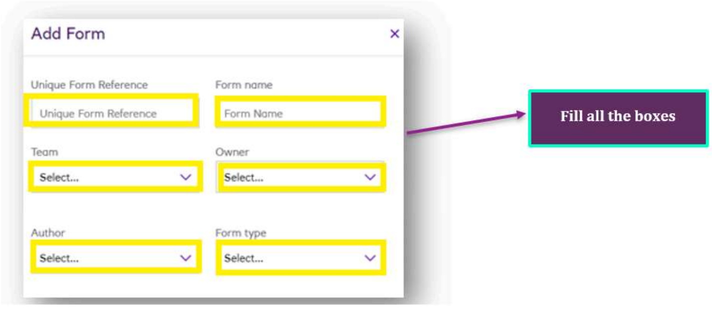
How to create or add new Forms

Addition/Creation of new e-Forms

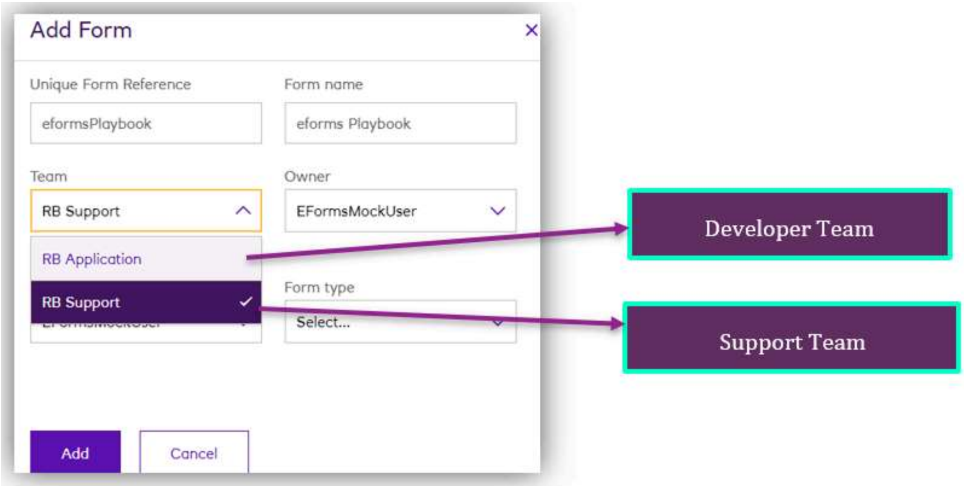
- Click on Authoring Page:- **Form definitions (page 1 of 2) - eForms Editor (banksvcs.net)**
- Below screenshot provides the details for the authoring page, where we can create the forms



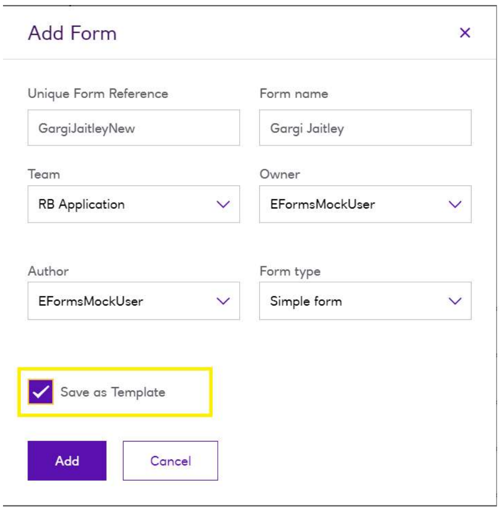
- Once we click on the add button below screen will pop up provide the details for the form.



- If you are Developer than you will select RB application and if you are from support team you will select RB Support

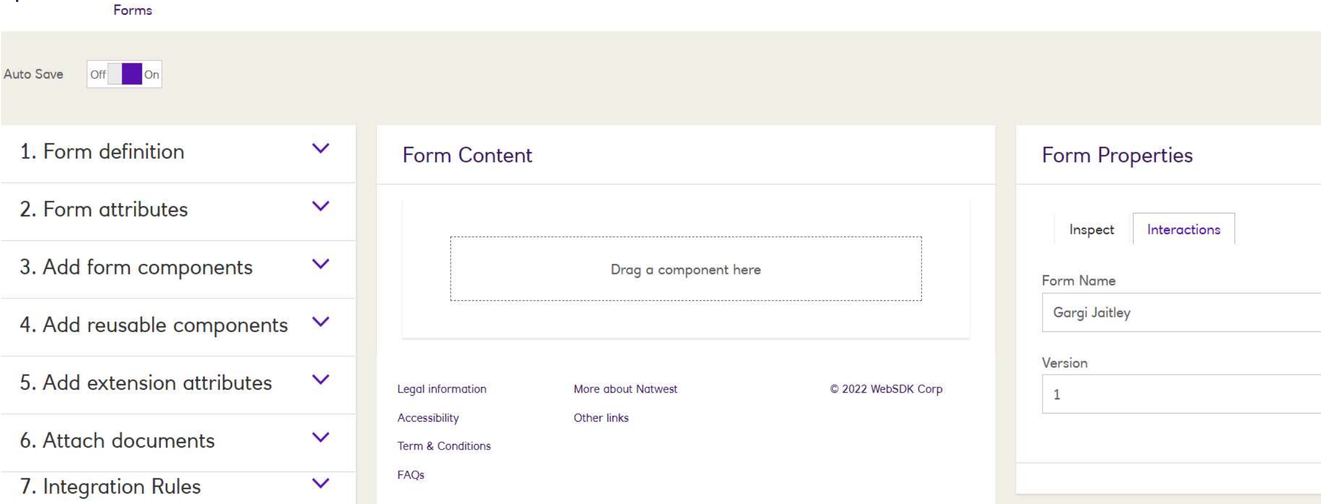


Note:- If you are creating a form you will not select "Template" as, if we select Template it will not work in Run time.
We will tick "Template" when we will be preparing Extension Template

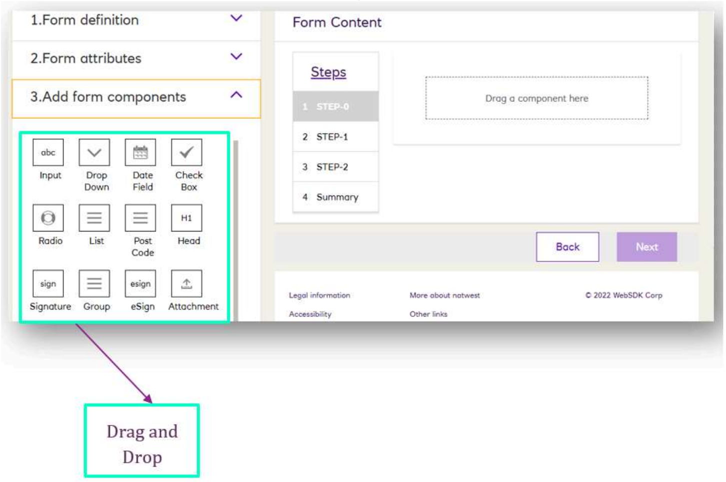


Note: While creating the form diligence should be given while entering value for Unique Form Reference, as this cannot be changed and this will be the Form ID/UUID that is going to be part of the URL that users will use.

- Below screen will appear once the details are filled in the above screen, here we will have the different option to create the forms.

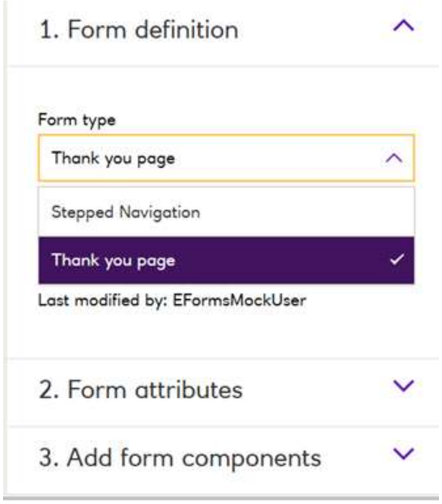


- Below screen will once the details are filled in the above screen, here we will have the different option to create the forms. We can drag and drop the components required for the forms in the forms content.



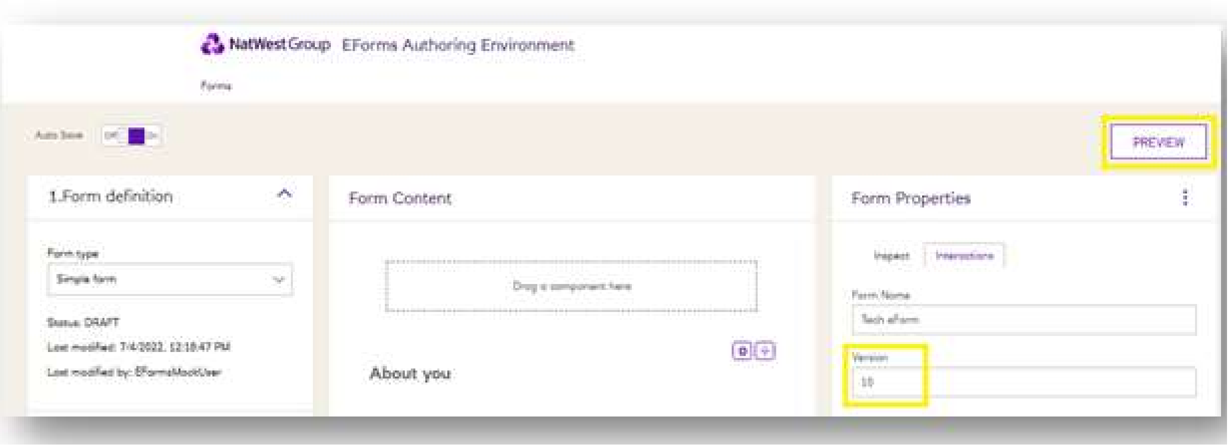
How to create a Thank you page

Go to the 1. Form Definition then go on Form type



Preview the eform and View Version control

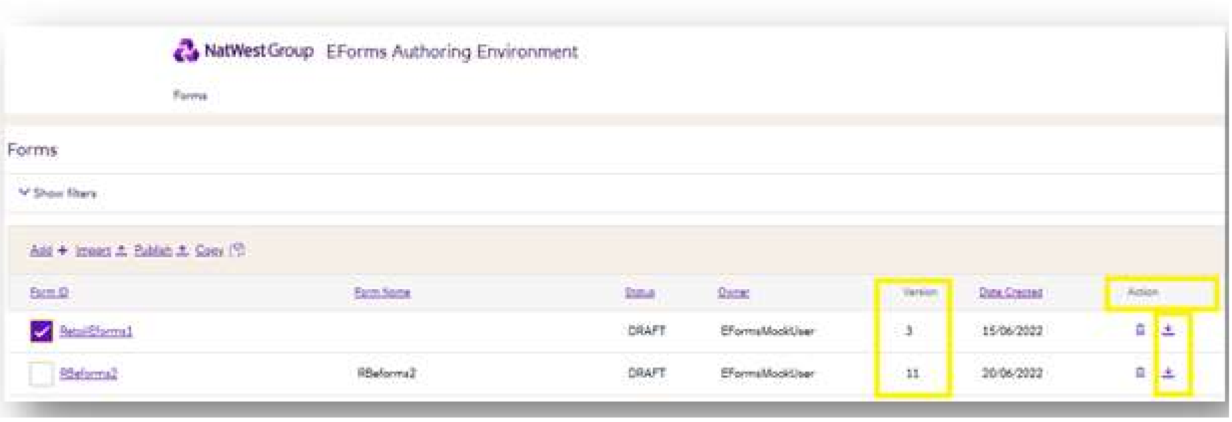
- Preview of the forms can be viewed through the **Preview** button at the right top corner.



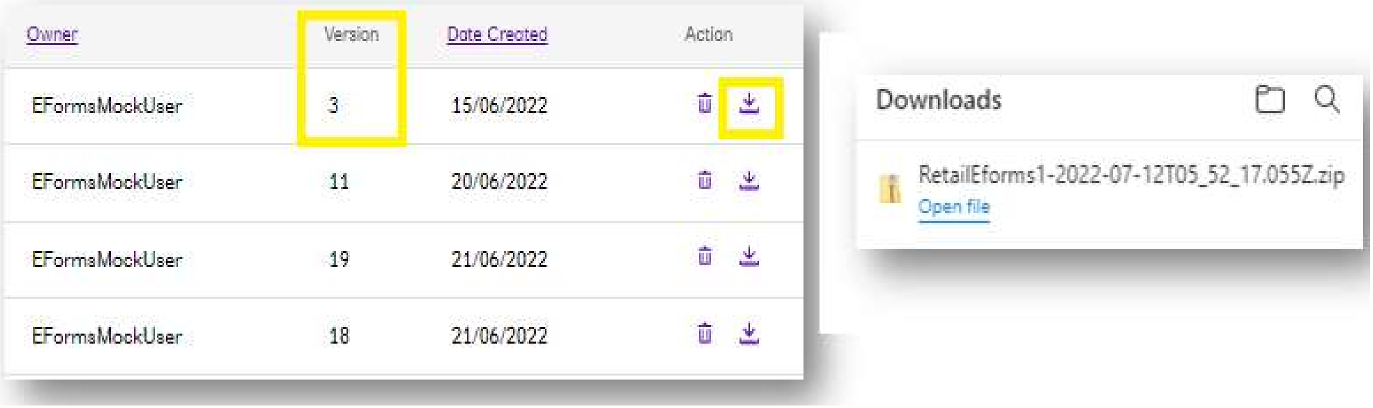
You can also view "**Version"** of your eforms, which means how many times it has been edited.

Download the eFORM

Once the form is created it can be viewed in the Authoring page. You can download the form the below highlighted icon.



- Once you will click on download option below pop will come, the downloaded file you will need to upload in next step.



Import Function

Import Do's and Don'ts,

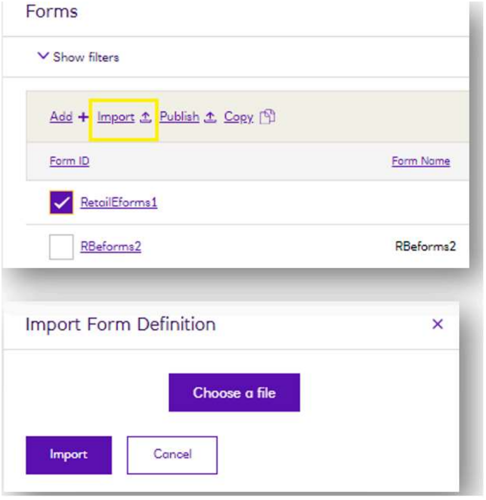
Do's

- Export a form from one environment and import it in another environment.
- If the Form has any extension attribute, before importing that form, the extension template form must be imported and published

Don'ts

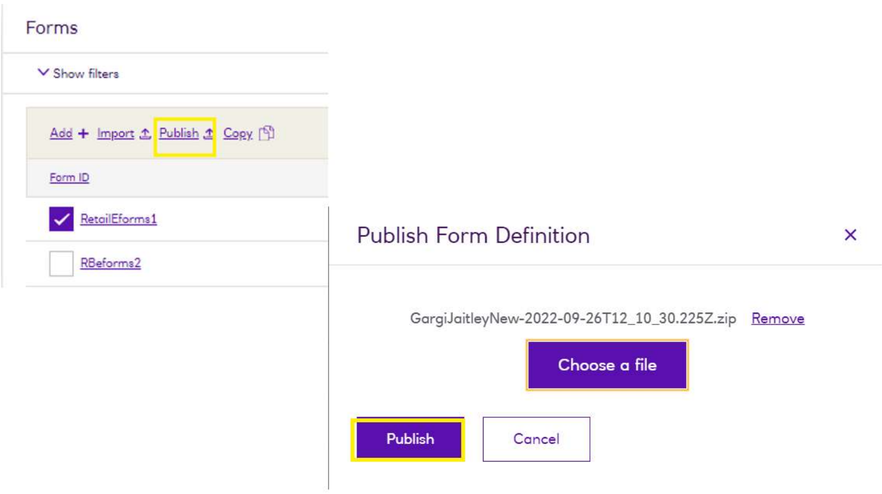
- Do not export and import in the same environment.
- Do not try to directly import a form which has an extension attribute before importing the extension template form in that particular environment.
- Below is option to publish and import the forms.
- Import: - This option can be used while we want to import the file from one environment to another environment (DEV, TEST,SIT,UAT)
- When you click on Import above screen will pop up, you need to click on "**Choose a file" and Import**
- After Import you need to publish the eforms, you can click on below **Publish icon**

- Once you click on Import icon, below screen will come and then you need to choose the Import.

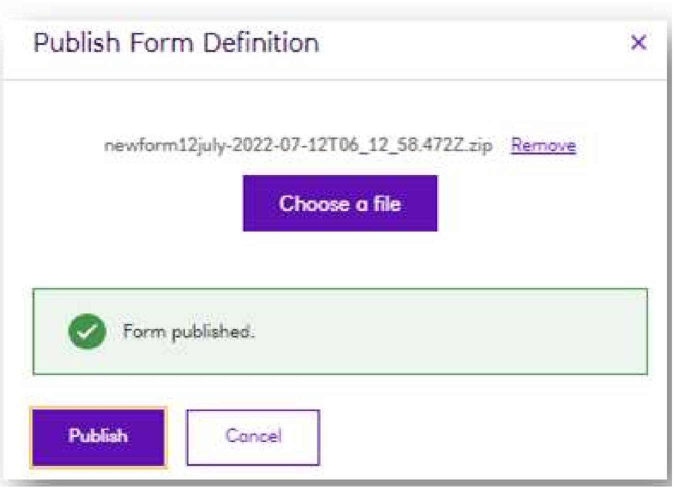


Publish

- For publishing the form , you first need to download the form
- Once you click on Publish icon, below screen will come and then you need to choose the downloaded file, which you downloaded in above steps.
- Upload the downloaded form to publish the form to runtime page



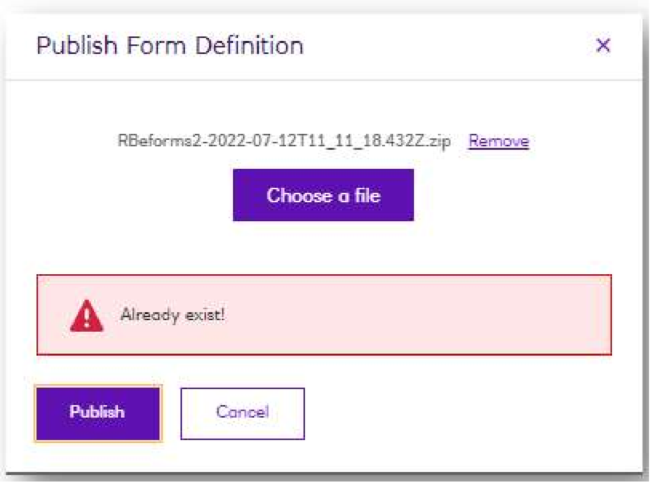
- Once your eform is published you will get below screen.



Note - Publish of the forms is required to actual view the forms on the runtime page. This will required post the import of the forms from one env to another

Error when you publish your eforms

- Since the e-form was already **Published** so we got this error, otherwise it would have successfully published.

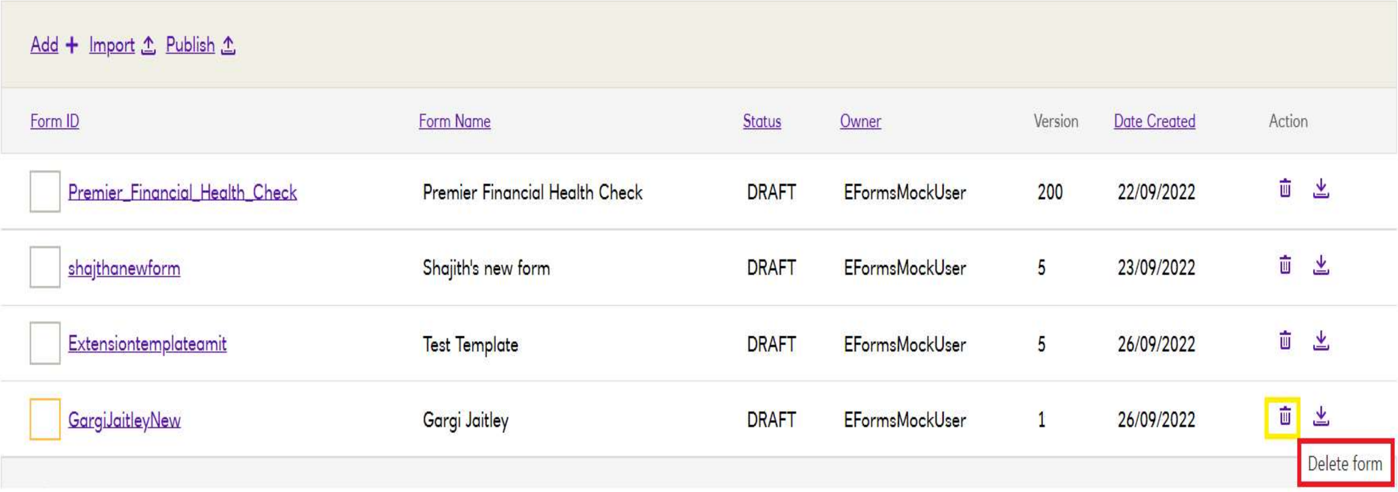


- Copy: - This option can be used to create the copy of the existing form

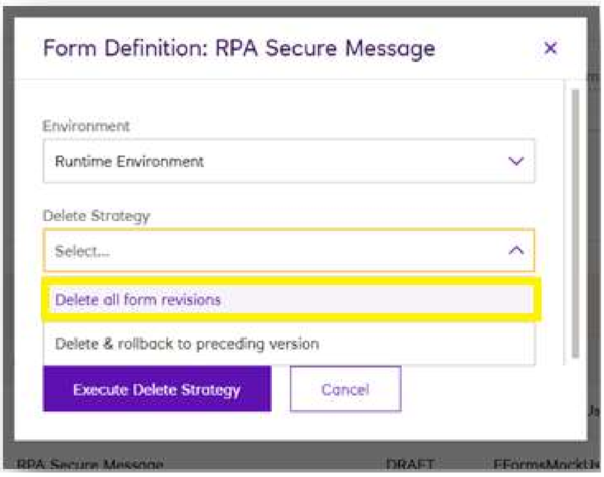
Process should be followed where a eform is published on dev and requires an edit before publishing again

You need to delete the previous published version. This can be done by the following steps :

Step 1: Select your process and click on the delete icon on action column.



Step 2 : A pop up will appear. Select Environment as “Runtime Environment” and Select “Delete Strategy as Delete & rollback to preceding versions” and click on execute delete strategy. This will led you to delete the previous published form. You can now publish the forms again.



publish your eforms in runtime

- Copy the below highlighted part from the URL of each authoring form and the append with the URL of the runtime page that will open the runtime page for that form

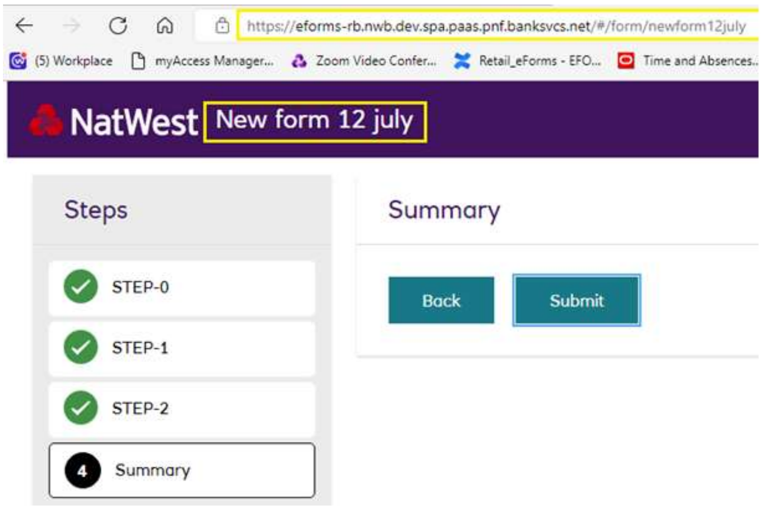
For example

Highlighted part :- form/ newform12july

Runtime page:- https://eforms-rb.nwb.dev.spa.paas.pnf.banksvcs.net/#

Runtime page for the form

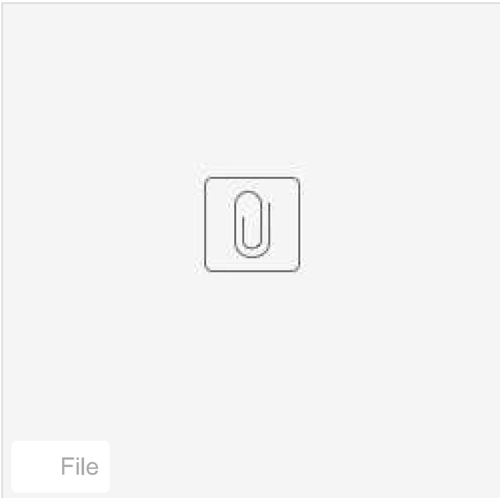
https://eforms-rb.nwb.dev.spa.paas.pnf.banksvcs.net/#/form/newform12july



How to create the FORm for email configuration

Note -so once the form is created, we will attach the Ftl file,extention template

Need 3 things- Extention file, FTL File and Email rules excel sheet (To be a send to EE team , they will raise request in BitBucket)

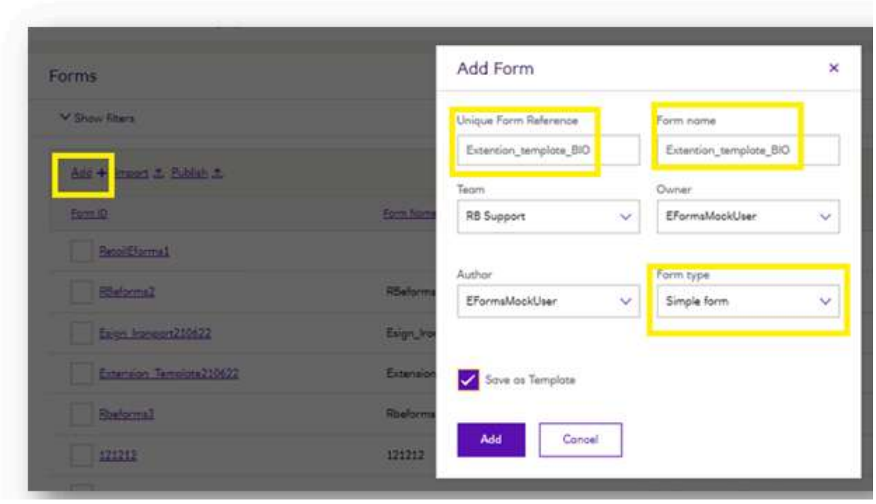


Steps are mentioned below

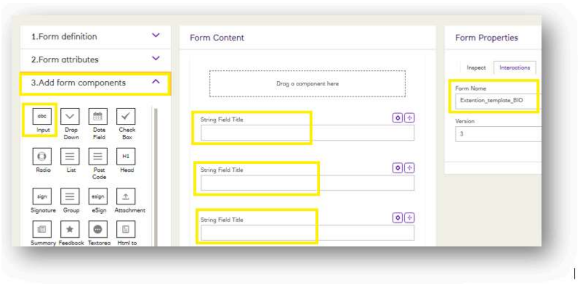
- First you need to click on this link Form definitions (page 1 of 3) - eForms Editor (banksvcs.net)
- Then you need to click on add tab, highlighted below, it will open the current page as seen below, then you need to add Unique Form Reference field **You need to mention "Extension_template" followed** by short form of your form name.

Here we have taken example of **Biometric Removal** form, so we have taken short name BIO in it.

- Likewise, we will fill the Form name field with the same name.i.e. **Extension_template_BIO**
- We will fill other respective field and in "**Form type"** we will select "**Simple form"**



- Once you submit your extension form below screen will come.
- Then you need to click on 3rd option that is “Add form components” and take “Input” components **3 times** and drag in “Form Content”
- You can now see the 3 “**String Field Title**”

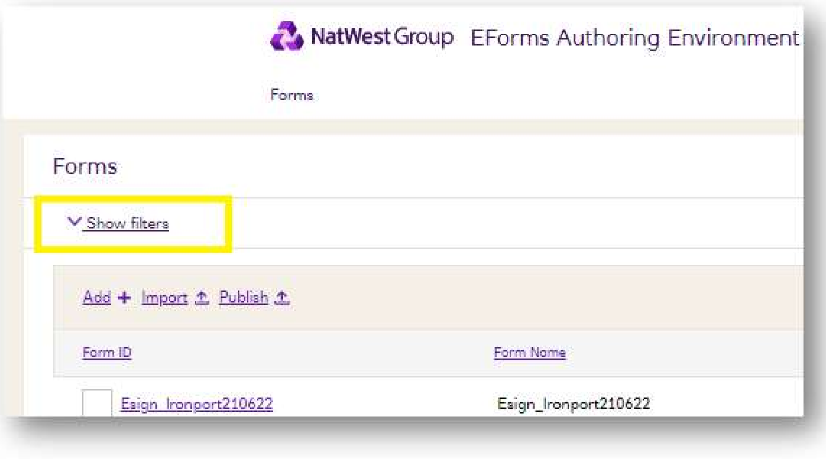


- Now we need to click on setting icon
- Then you need to fill 2 fields i.e.” **Element Name**” and “**Label**”
- The 3 components which we have added, we need to fill it with **Customer Email,**

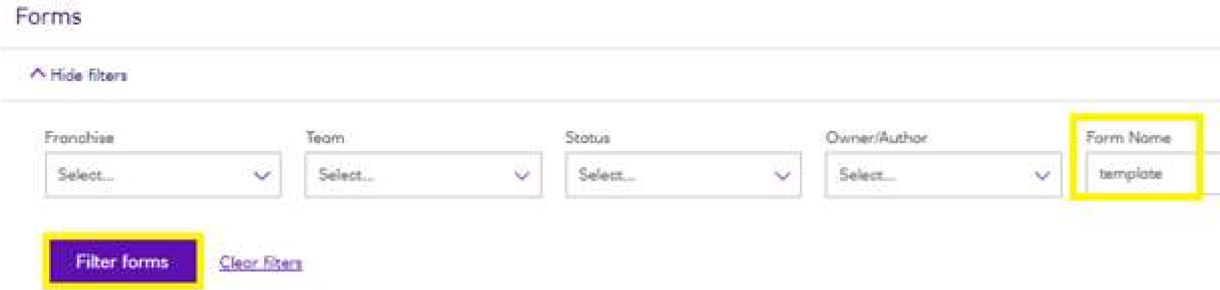
Colleague Mailbox and **Mail Template,** these 3 are mandatory field but you can also add Bcc and CC as well

How to attach the FORm for email configuration

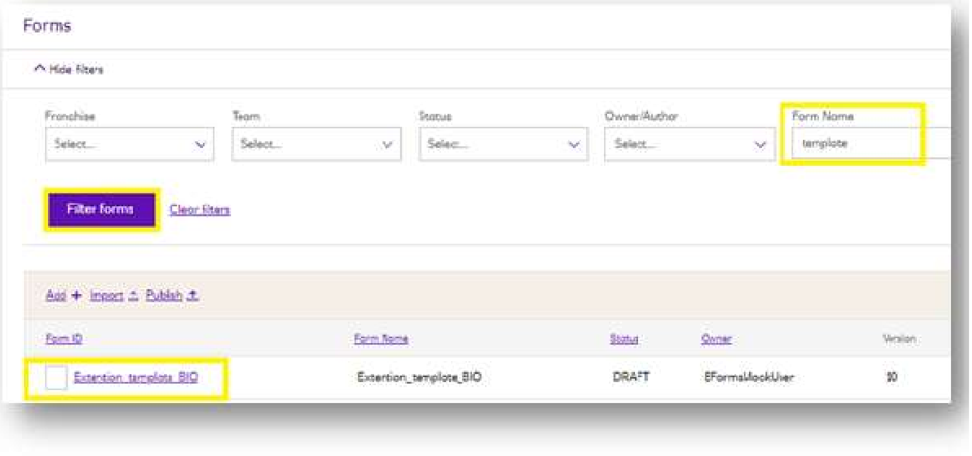
- First you need to click on this link Form definitions (page 1 of 3) - eForms Editor (banksvcs.net)
- Then you need to click on Show filters tab highlighted below.



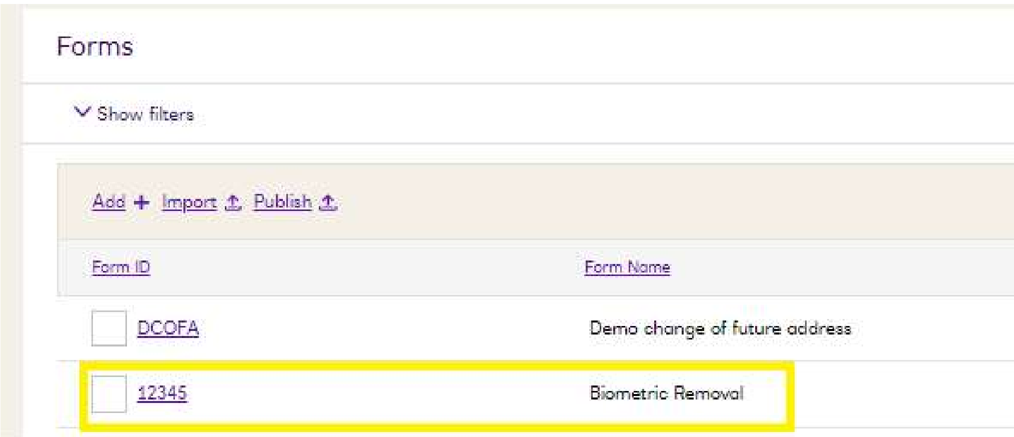
- Once you click on the filter below screen will come, you need to fill Form Name field just type "**template**" and click on "**filter forms button"**



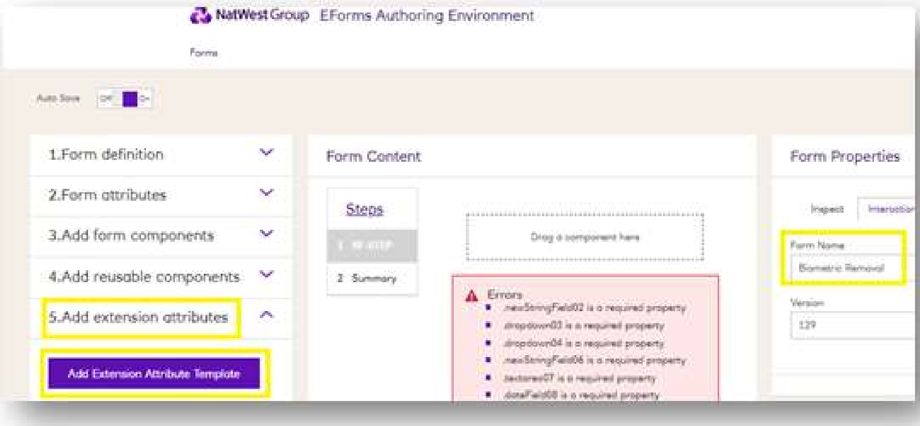
- We can view the form which we have created in above steps “Extension_template_BIO”



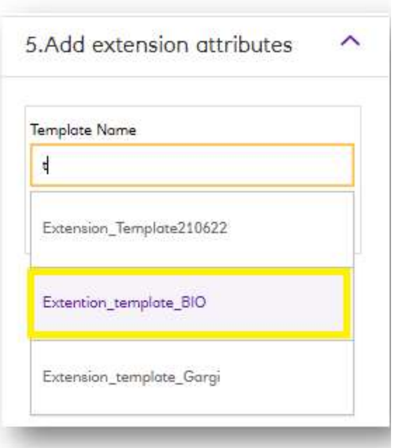
- Now we will select Biometric Removal form.



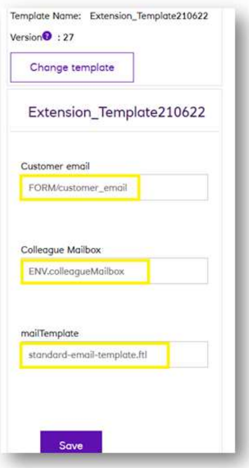
- Now we will open Biometric Removal form and select “Add extension attributes”
- Then click on “Add extension attributes Template”



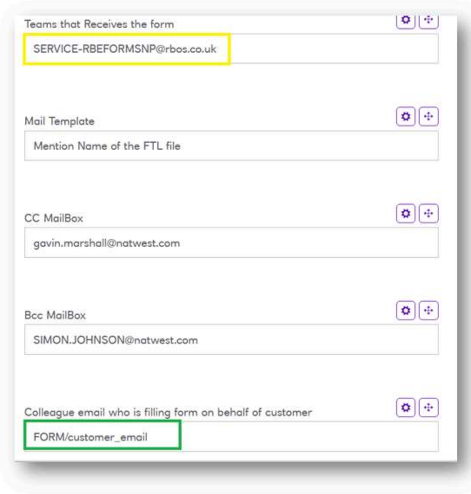
- Now we will select our Extension file i.e Extension_template_BIO



- Once we select Extension_template_BIO we will get below options
- You need to fill Customer Email- will this value "**FORM/customer_email**"
- Colleague Mailbox-currently we are using "**ENV.colleageMailbox or SERVICE-RBEFORMSNP@rbos.co.uk**"
- And mail Template, in Mail template you need to mention name of FTL file in this case its **"bio-template.ftl"**



Another Example of Extension form



Customer Email = Personal is filling the form

FORM/customer_email (Email is pulled from field value of the component Customer_email within the form itself

----

Colleague Mailbox = Team which receives the email

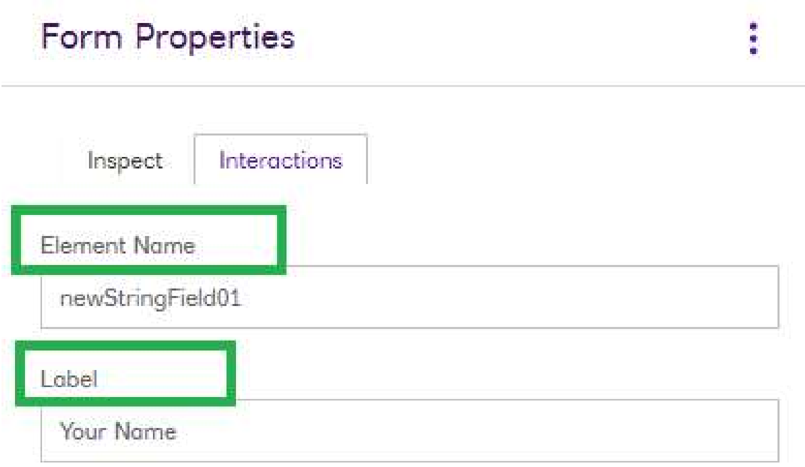
ENV.colleageMailbox (This is set in the excel document itself to send to the department/area)

What is fTL file

- FTL file is a template used by FreeMaker, a Java Template engine used to auto-generate text output. It contains source text as well as FreeMarker variable definitions and instructions that are used as placeholders for text substitutions.
- FTL files are commonly used for auto-generating HTML webpages.
- Need to edit this file according to our need and then paste it.(Open in Note++)

Steps to EDIT FTL FILE

- We will take example of Biometric removal_email_body.ftl
- In the FTL file 2 things are important" **Label and Value**"
- Label is a plain text that you can give to form fields
- Inside $(body.value) - value is the "**Element name"** defined in authoring environment for that field. which will hold the value entered by customer for that field

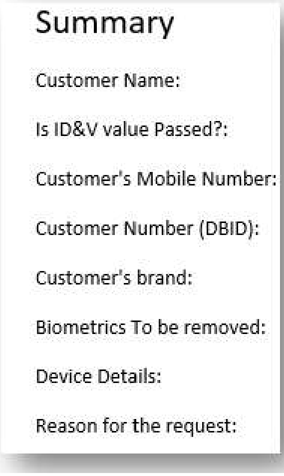


- Below the sample of the mail that we get from FTL file

- In FTL file you need to edit row no. 65 for Brand name and logo, it can be RBS or Natwest.
- Below HTML screenshot shows the Header can be edited at row no.80 and 104

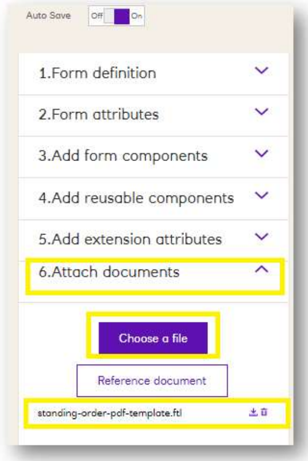
- This screenshot of mail will show how the Heading “Biometric Removal Request Confirmation” will be look like in mail.

- Below the screen shot of other attributes

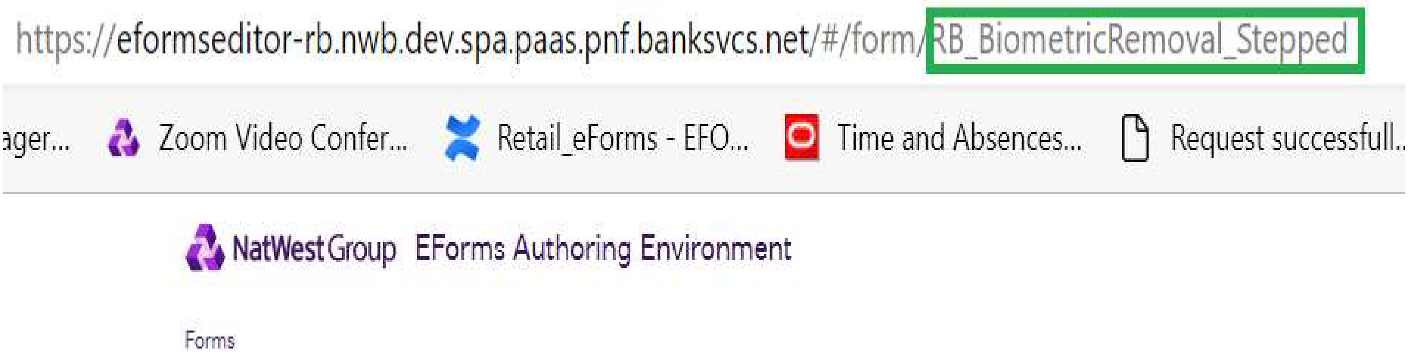


- In the FTL file 2 things are important" **Label and Value**"
- Element name is the **Label** text that you can give to form fields in authoring environment
- Inside $(body.value) - value is the "**Element Value"** defined in authoring environment for that field. which will hold the value entered by customer for that field
- You can edit in Note ++ Row no.134 for Customer name (Label)
- Row no. 137 shows Customer name (Element value)

- Above screen shot shows Element name and Label in authoring environment for that field which will hold the value entered by customer for that field.
- Once you create the FTL file you will attach that in authoring environment



- Then we need to click on "Forms" icon
- Then we will select our form, which we want to publish.
- Copy the below highlighted part from the URL of each authoring form and the append with the URL of the runtime page that will open the runtime page for that form



For example

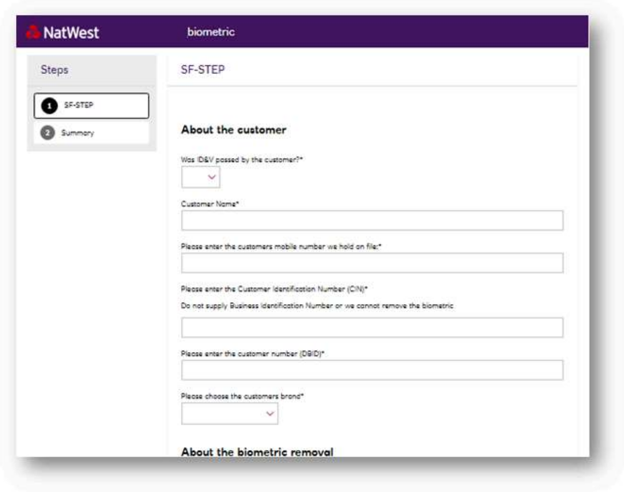
Editor page link- https://eformseditor-rb.nwb.dev.spa.paas.pnf.banksvcs.net/#/form/RB_BiometricRemoval_Stepped

Highlighted part:- form/RB_BiometricRemoval_Stepped

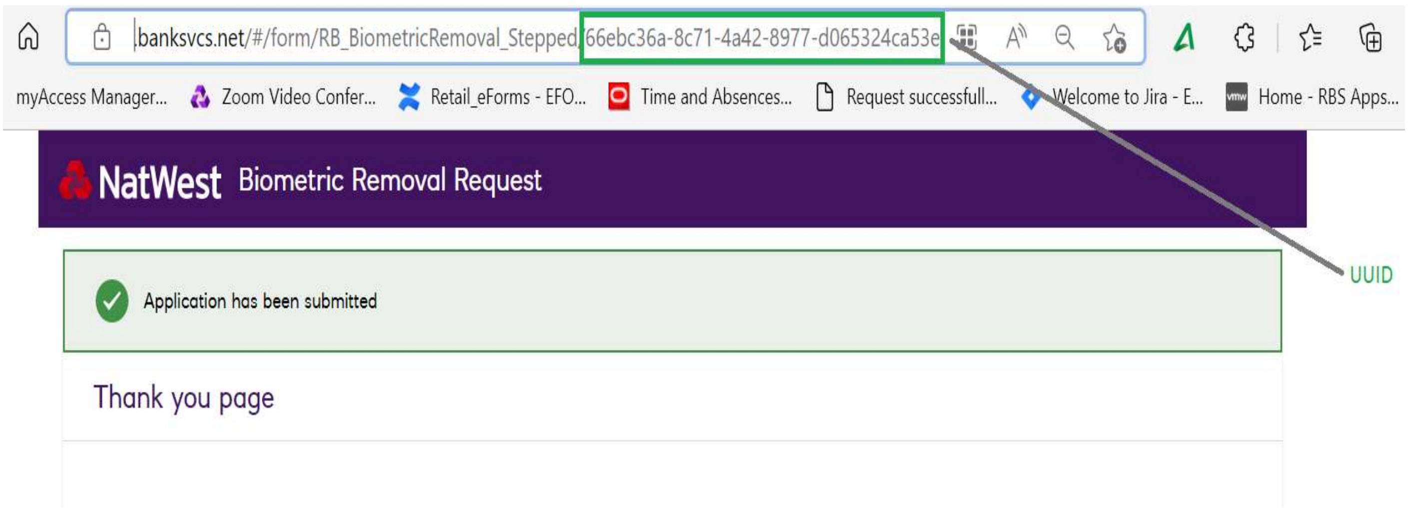
Runtime page: - https://eforms-rb.nwb.dev.spa.paas.pnf.banksvcs.net/#

Runtime page for the form- https://eforms-rb.nwb.dev.spa.paas.pnf.banksvcs.net/#/form/RB_BiometricRemoval_Stepped

- Once we copy paste the link we will get the below page



- Once you view this page on run time, you need to fill the form and submit it.



- After you submit this form UUID will be created
- And you will receive mail in your in box.

FORMAT CONVERSION SYNTAX IN FTL

Since FTL is used to create views, there are instances where we want to make a value/data more human readable. Boolean values true or false can be formatted as "1" / "0" or "Yes" / "No" text. FTL provides a simple syntax to format values.

In the following picture, we can see the syntax being used on a checkbox in some ftl file.

```
                        <p class="elementValue"> ${body.Senderemail}</p>
                    </td>
                </tr>
                <tr>
                    <td>
                        <p class="elementName">Employee ID</p>
                    </td>
                    <td>
                        <p class="elementValue"> ${body.Salaryref}</p>
                    </td>
                </tr>
                <tr>
                    <td>
                        <p class="elementName">Branch Sort Code</p>
                    </td>
                    <td>
                        <p class="elementValue"> ${body.Referring_Branch_Sort_Code}</p>
                    </td>
                </tr>
                <tr>
                    <td>
                        <p class="elementName">I confirm that the customer has provided their consent for us to pass their contact details and brief details of
                    </td>
                    <td>
                        <p class="elementValue">${body.checkbox35?string('yes','no')}</p>
                    </td>
                </tr>
            </table>
        <footer>
            <#if formData.brand=="rbs">
                <table cellpadding="0" cellspacing="0" border="0" role="presentation"
                    style="width:650px;color:grey;font-family:Calibri;font-size: 12px">
                    <tr>
                        <hr />
                        <td>
                            <p>The Royal Bank of Scotland plc Registered in Scotland No 83026.
                              Registered
                              Office: 36 St Andrew Square, Edinburgh EH2 2YB. <br /><br />
                              Authorised by the Prudential Regulation Authority and regulated by the
                              Financial Conduct Authority and the Prudential Regulation Authority.
                            </p>
                        </td>
```

In line highlighted section, <p class="elementValue"> ${body.checkbox35?string('yes','no')}</p> illustrates how check box is added using ternary operator.

If the checkbox value is ***false,*** value evaluated is ***"No"*** and if it is true, ***"Yes"*** is evaluated**.**

The following snippet uses the same syntax for formatting date.

7.3. Date Handling

- *.now* represents the current date-time
- *date*, *time* and *datetime* can return the date and time sections of the date-time object
- *string* will convert date-times to strings – we can also pass the desired format or use a pre-defined one

We're going to now get the current time and format the output to a string containing only the hours and minutes:

```
<p>${.now?time?string('HH:mm')}</p>
```

We are using the Date Handling to achieve check box as above.

FTL provides various other syntax to add logic to the template. Please refer ftl documentation available online to explore more.

How to check null values in FTL?

Sometimes, when we use show and hide functionality in a form, based on input for joint or solo account, the next page should show either one applicant or both applicants. If the user fills the form as joint and go back to change it as solo before submitting the form, the email may show all details of joint account also. We can handle it in FTL using a conditional statement of if (jointOrSoloAccount= "Yes") then <.....>, and this can work fine in single brand forms or where the if condition is NOT brand specific, but in some forms we may have to give condition based on all brands, and when we used the above condition, email may not be generated and error may be thrown.

This happens due to null values from the elements used for other brands, so we need to check null for all the elements before the conditional statement is executed and condition needs to be set for all three brands separately.

Syntax to check null:

```
 <#if body.elementName??>

<#if>
```

Example:

```
<#if body.isThisAJointAccountNatWest??>

<#if body.isThisAJointAccountNatWest=="Yes">

<#if body.firstNameApplicantTwo??>

<tr>

                    <td>

                       <p class="elementName">Your first name in full</p>

                    </td>

                    <td>

                       <p class="elementValue"> ${body.firstNameApplicantTwo}</p>

                    </td>

                     </tr>


<#if>

<#if >

<#if >


<#if body.isThisAJointAccountRBS??>

<#if body.isThisAJointAccountRBS=="Yes">

<#if body.firstNameApplicantTwo??>

<tr>

                    <td>

                       <p class="elementName">Your first name in full</p>

                    </td>

                    <td>

                       <p class="elementValue"> ${body.firstNameApplicantTwo}</p>

                    </td>
```

```
                              </tr>


<#if>

<#if >

<#if >


<#if body.isThisAJointAccountUlser??>

<#if body.isThisAJointAccountUlster=="Yes">

<#if body.firstNameApplicantTwo??>

<tr>

                            <td>

                                <p class="elementName">Your first name in full</p>

                            </td>

                            <td>

                                <p class="elementValue"> ${body.firstNameApplicantTwo}</p>

                            </td>

                              </tr>


<#if>

<#if >

<#if >
```

How to define forms-rule-definition ?

About: Eforms tool provide the power of creating reusable forms with dragging and dropping components over the form area. These components are the building blocks of the form. Values of these components are structured into a single data i.e *form-data*. But components in a form don't have the capability to *talk to each other* and are unaware of each others presence. There are cases where we want one component A to be aware of component B's value and should behave depending on B's value or vice versa.

Few scenarios where we want one component/field to behave according to another component's/field's value are given below :-

Sheldon is a bank's customer

1. **Scenario I**
   Sheldon is filling a form, where he is asked "Do you have a PAN card ?" Depending on Sheldon owning a PAN card, he will select "Yes" or "No". Lets assume Sheldon do not have a PAN card.
   Sheldon selects "No" and "Enter PAN Card details" disappears from the screen.

2. **Scenario 2**
   Sheldon wants Plastic Money (Debit/ credit Card). The bank uses a common stepped form for Debit & Credit Cards application.
   First step is for basic details. Second page is for details specific to Credit Card & Third page is for details specific to Debit Card.
   On first page (first step) Sheldon is asked
   Request For
   A. Credit Card
   B. Debit Card
   C. Both
   Depending on Sheldon's choice of request, form is expected to behave in 3 different ways.
   - If Credit Card is selected , Debit Card step should disappear.
   - If Debit Card is selected, Credit Card step should disappear.
   - If both is selected, both should remain.

3. **Sheldon is filling a form for fuel reimbursement.**
   If the value to be reimbursed exceeds a threshold value, **"Number of vehicles owned"** becomes a mandatory field.

All the above mentioned features can be added to a form using the **"Rules definitions"**.

Rules-definition is a code editor, available in **“Developer Mode”** right sidebar. The code it takes is a JSON object that can have a **single or multiple “condition-event” rules.**

The highlighted area in green is the editor textarea where the rules are defined.

How to check null values in FTL?

Scenario: Sometimes, based on input for joint or solo account, the next page should show either one applicant or both applicants. If the user fills the form as joint and go back to change it as solo before submitting the form, the email may show all details of joint account also. We used to handle it in FTL using a conditional statement of if (jointOrSoloAccount= “Yes”) then <.....>, and this can work fine in single brand forms or where the if condition is NOT brand specific, but in some forms, we may have to give condition based on all brands, and when we use the above condition, email may not be generated and error may be thrown.

Solution: This can happen due to null values from the elements used for other brands. We need to check null for all the elements before the conditional statement is executed and condition needs to be set for all three brands separately.

Example:

```
<#if body.isThisAJointAccountNatWest??>

<#if body.isThisAJointAccountNatWest=="Yes">

<#if body.firstNameApplicantTwo??>

<tr>

            <td>

              <p class="elementName">Your first name in full</p>
```

```
                    </td>
                    <td>
                        <p class="elementValue"> ${body.firstNameApplicantTwo}</p>
                    </td>
                      </tr>

<#if>
<#if >
<#if >

<#if body.isThisAJointAccountRBS??>
<#if body.isThisAJointAccountRBS=="Yes">
<#if body.firstNameApplicantTwo??>
<tr>
                    <td>
                        <p class="elementName">Your first name in full</p>
                    </td>
                    <td>
                        <p class="elementValue"> ${body.firstNameApplicantTwo}</p>
                    </td>
                      </tr>

<#if>
<#if >
<#if >

<#if body.isThisAJointAccountUlser??>
<#if body.isThisAJointAccountUlster=="Yes">
<#if body.firstNameApplicantTwo??>
<tr>
                    <td>
                        <p class="elementName">Your first name in full</p>
                    </td>
                    <td>
```

<p class="elementValue"> ${body.firstNameApplicantTwo}</p>

</td>

</tr>

<#if>

<#if >

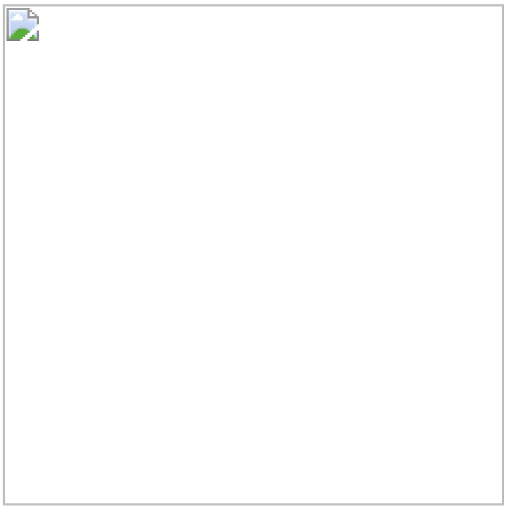
<#if >

Solution Scenario 1: If customer do not have a pan card, the pan card details field hide.

1.Identifying the components.

In the given scenario, selecting **Radiobutton "Yes"** or **"No"** as option is one of the component, that is pretty obvious. The another component is the **Enter Pan Card Details:** field that gets hidden when **"No"** is selected.

Now we know the components, but we want our form to know about them, Each component in a form has a unique identifier i.e its ***element name***.



Clicking on gear icon associated with every component, opens up the details of each component in right sidebar. Following this procedure we can find element name of our components involved.

In our case it is:

Type	Label	identifier
RadioButton	Do you have a Pan Card ?	exampleRadioButton00
StringInput	Enter Pan Card Details	newStringField01

2. Defining a condition

A rule consists of a condition and an event object. Condition object follows a JSON structure

```
{
  "conditions": {
    "exampleRadioButton00": {
      "equal": "No"
    }
  },
  "event": { }
}
```

It says, when **exampleRadioButton00** component has **No** value then the associated event (an empty event) is triggered. We will define event in next step.

3. Defining event

The event we want is, hiding the Pan Card Details component, in first step we identified it with name newStringField01

We tell this field to disappear using the following syntax.

```
  "event": {
    "type": "remove",
    "params": {
      "field": [
        "newStringField01"
      ]
    }
  }
```

The whole rule looks like following:

```
[
  {
    "conditions": {
      "exampleRadioButton00": {
        "equal": "No"
      }
    },
    "event": {
      "type": "remove",
      "params": {
        "field": [
          "newStringField01"
        ]
      }
    }
  }
]
```

Following is the completed rules-definition for scenario 1.