

E-Form DB Collections

1. **forms-editor-form-definition**- Form details in editor/authorizing environment. Once form created in editor, new entry gets created.
2. **runtime-form-definition**- Form details of runtime environment. Once form published in runtime, new entry gets created.

uuid- Unique Form ID.

version- Version of the form. If any changes in the form, version will get updated.

name- Name of the form.

description- Description of the form.









dateCreated- Date on which form creation.

createdBy- Name of the user who created form.

franchise- Franchise name under which form created.

team- Team name under which form created.

owner- Owner name.

UUID		Name		Owner		Version	dateCreated	Action
Add + Import Import Publish								
Form ID		Form Name		Status	Owner	Version	Date Created	
<input type="checkbox"/>	RB_ExtensionEmailTemplate	Rb Extension Email Template		DRAFT	EFormsMockUser	38	18/08/2022	 
<input type="checkbox"/>	RB_ATMClaim	ATM/CDM Claim		DRAFT	M01EUROPA_kumpaaf	223	05/08/2022	 
<input type="checkbox"/>	sampleextensiontemplate	Sample Extn Template		DRAFT	RBFORMSUSER006	73	01/11/2022	 
<input type="checkbox"/>	RBSTEPPEDFORM			DRAFT	RBFORMSUSER006	133	08/11/2022	 

isTemplate- Tells whether it's a template or a form. False represents form, True represents template.

isSteppedForm- Tells whether the form is step form or simple form. False represents Simple form, True represents stepped form.

lastModified- Date in which the form modified.

lastModifiedBy- Name of the user who modified the form.

stepDefinitions- Details of each step in the Form.

stepDefinitions.name- Name of the step.

stepDefinitions.sequenceNumber- Order number of the step.

stepDefinitions.dataSchemaReference- To control the details of the component (like mandatory, title, pattern) in the form.

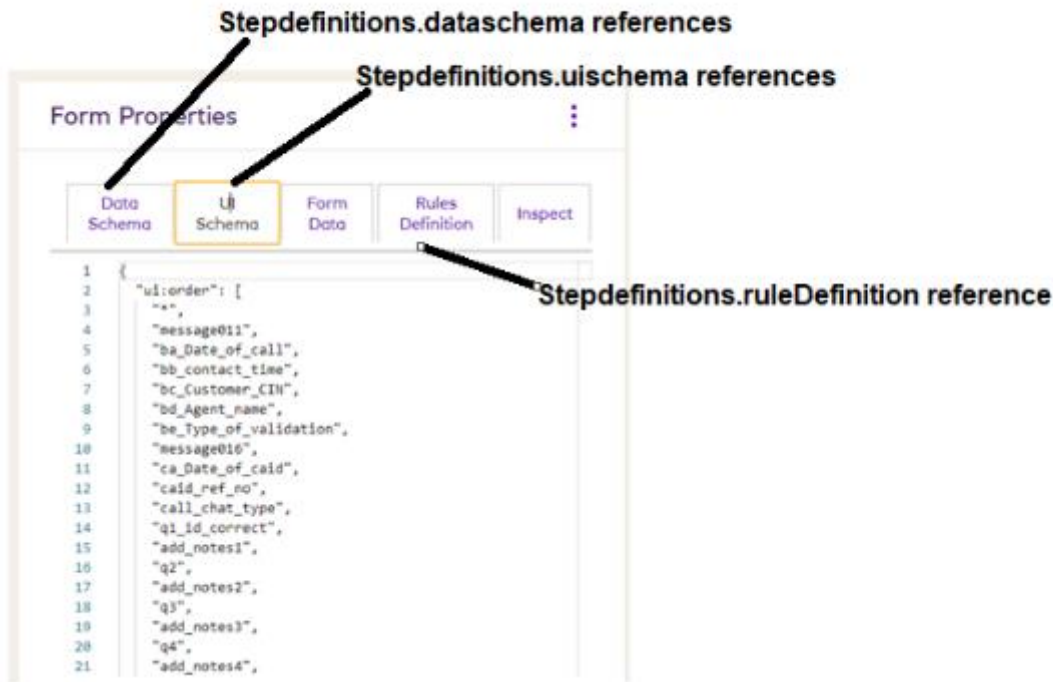
stepDefinitions.dataSchemaReference.uuid- Unique id to refer in the collection **forms-editor-data-schema** (To refer the form in editor), **runtime-data-schema** (To refer the form in runtime).

stepDefinitions.dataSchemaReference.version- Version of the component details.

stepDefinitions.uiSchemaReference- To control the order of the component in the form.

stepDefinitions.uiSchemaReference.uuid- Unique id to refer in the collection **forms-editor-ui-schema** (To refer the form in editor), **runtime-ui-schema** (To refer the form in runtime).

stepDefinitions.uiSchemaReference.version- version of the order of the component.



stepDefinitions.ruleDefinitionReference- Interaction (Show & hide) of each step of the forms. It says how the components should behave(show/hide) in the steps. To make the form dynamic/responsive.

stepDefinitions.ruleDefinitionReference.uuid- Unique id to refer in the collection **rule-definition** (To refer the form in editor), **runtime-rule-definition** (To refer the form in runtime).

stepDefinitions.ruleDefinitionReference.version- version of the interaction of the step.

extensionDefinition- Extension attribute of the form.

extensionDefinition.templateFormDefinitionReference.uuid- Unique id of the extension template.

extensionDefinition.templateFormDefinitionReference.version- Version of the extension template.

extensionDefinition.templateFormDefinitionReference.dateCreated- Date on which extension template created.

extensionDefinition.lastestTemplateVersion- Version of the extension template.

extensionDefinition.extensionData.fromEmailPath- From Mail Id.

extensionDefinition.extensionData.submitterEmailPath- Form Submitter mail id.

extensionDefinition.extensionData.ftlNamePath- Name of the ftl file attached.

extensionDefinition.extensionData.ToEmailPath- To mail id.

5. Add extension attributes

Template Name: RB Ela Email Template
Multiple50
Version: 120

Change template Refresh version Remove Template

RB Ela Email Template
Multiple50

Email Content

FTL Template Name
The FTL file attached to this form, please include the file extension (.ftl)

Call_Quality_Validation_Assessment.ftl

From
Mailbox that will issue email containing formdata.

SERVICE-RBFORMS@rbs.co.uk

To Email
Mailbox that will receive the submitted formdata

ruleDefinitionReference- Interaction (Show & hide) of the form. It says how the steps should behave(show/hide). Linked with the rule-definition collection.

ruleDefinitionReference.uuid- Unique id to refer in the collection **rule-definition** (To refer the form in editor), **runtime-rule-definition** (To refer the form in runtime).

ruleDefinitionReference.version- Version of the interaction of the form.

ruleDefinitionReference.dateCreated- Date on which form created.

ruleTemplateDataReference- Integration rule of the form. Emails, event-enrichment and kafka-event- partitioning are the 3 integration rules available in the form. Those are stored as array. Linked with **forms-editor-rule-template-data** (To refer the form in editor), **forms-rule-template-data** collection (To refer the form in runtime).

ruleTemplateDataReference.uuid- Unique id of the integration rule.

ruleTemplateDataReference.version- Version of the integration rule.

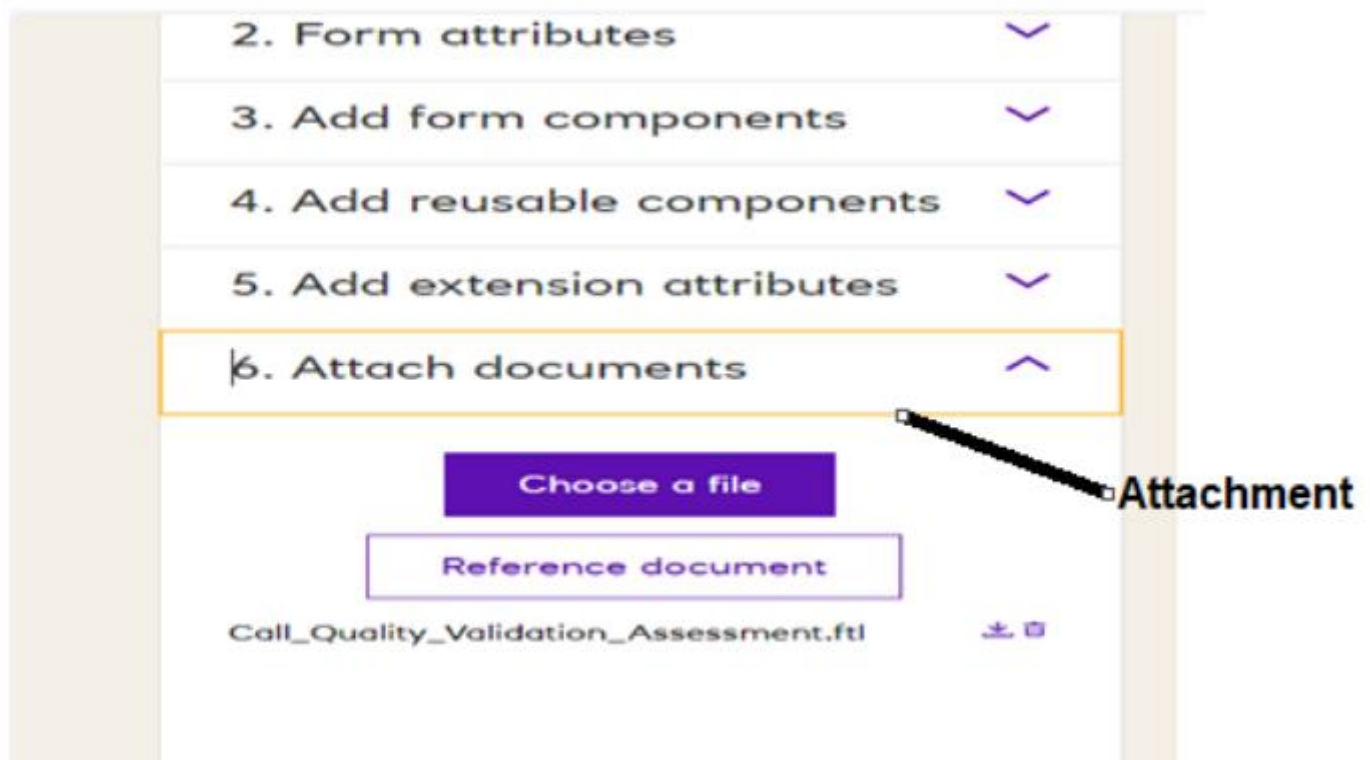
ruleTemplateDataReference.dateCreated- Date on which the integration rule created.

attachmentReferences- FTL attachment details are stored as array.

attachmentReferences.uuid- Unique id to refer the ftl file. Uuid can be mapped to Id of fs.files and files_id of fs.chunks collection.

attachmentReferences.filename- Name of the attached file.

attachmentReferences.mimeType- Type of file form should accept.



3.forms-editor-form-definition-revs- Archive of the form details in editor/authorizing environment.

4.runtime-form-definition- Archive of the form details in runtime environment.

5 fs.files- FTL attachment details.

_id- Id mapped to the files_id in fs.chunks collection to fetch the file.

filename- Name of the file.

length- Length of the file.

uploadDate- Date on which FTL file uploaded on the form editor.

metadata.parentUuid- Form Reference no (id of the form).

metadata.contentType- Type of the file.

6.fs.chunks- Attached FTL file in form editor.

files_id- Id to reference a form. Linked to collection fs.files.

data- Attached FTL file.

7. forms-editor-data-schema- Editor version. It stores the component details of the form, like component name, component type, component pattern.

uuid- Unique id to refer the components.

version- Version of the form component details. If any component is added/ removed/ modified, then version will get update.

dateCreated- Date on which the form created.

















schema.properties- List of component in the form.

Title- Title/Label of the component.

Text- Content to be displayed.

Widget- Type of component. HtmlToTextWidget, notification, CheckboxWidget, TextareaWidget,radio.

3. Add form components

Input	Drop Down	Date Field	Check Box
			
Radio	List	Post Code	Head
			
Signature	Group	eSign	Attachment
			
Summary	Direct Debit	Feedback	Textarea
			
Html to text	Dependency	Link Button	Feedback Comment

elementName- Unique name to the component.

componentType- Type of component.

Placeholder- Sample content for the component.

Pattern- It is the regex pattern to accept the input.

patternError- It is the error message to be displayed when the user input not match with pattern.

Pattern

^(((0-1)[0-9]|2[0-3]):[0-5][0-9])|24:00)\$

Pattern Error text

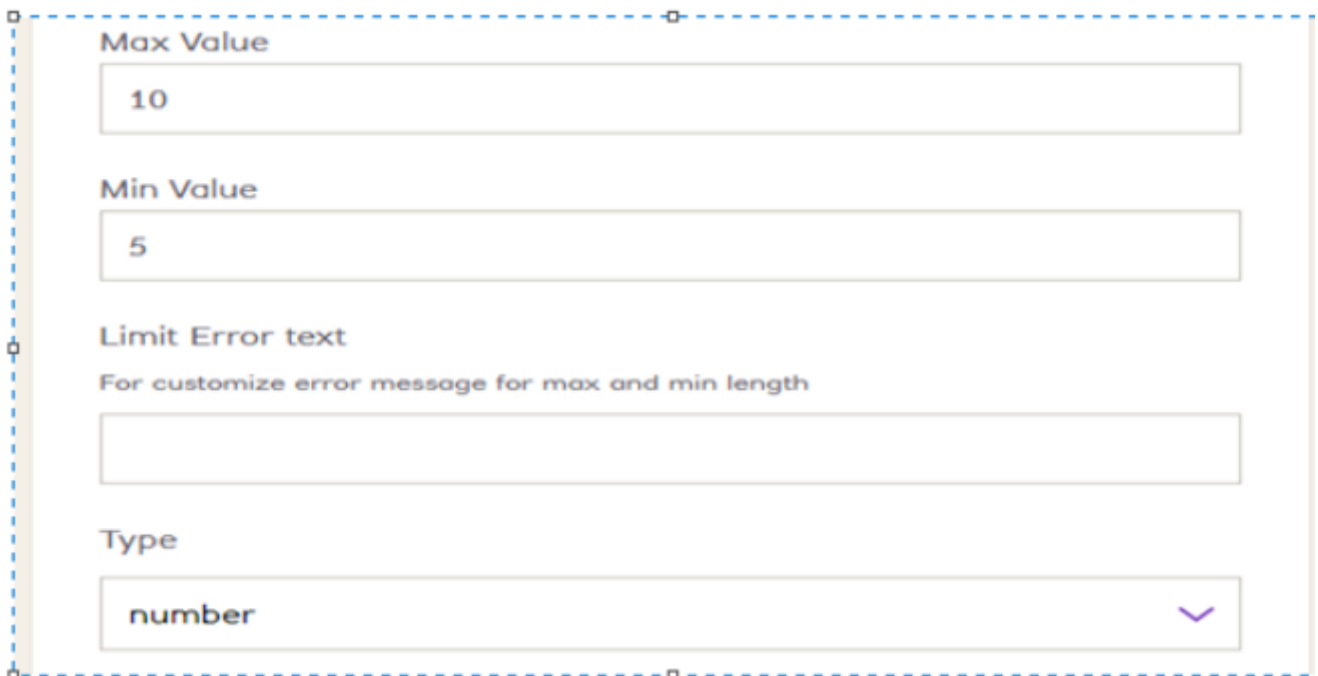
For a pattern provide the customizable error text

'Enter Contact Time as HH:MM'

maxLength- Maximum number of character input should accept.

minLength- Minimum number of character input should accept.

Type- Type of the component. String, Boolean, number.



Max Value

10

Min Value

5

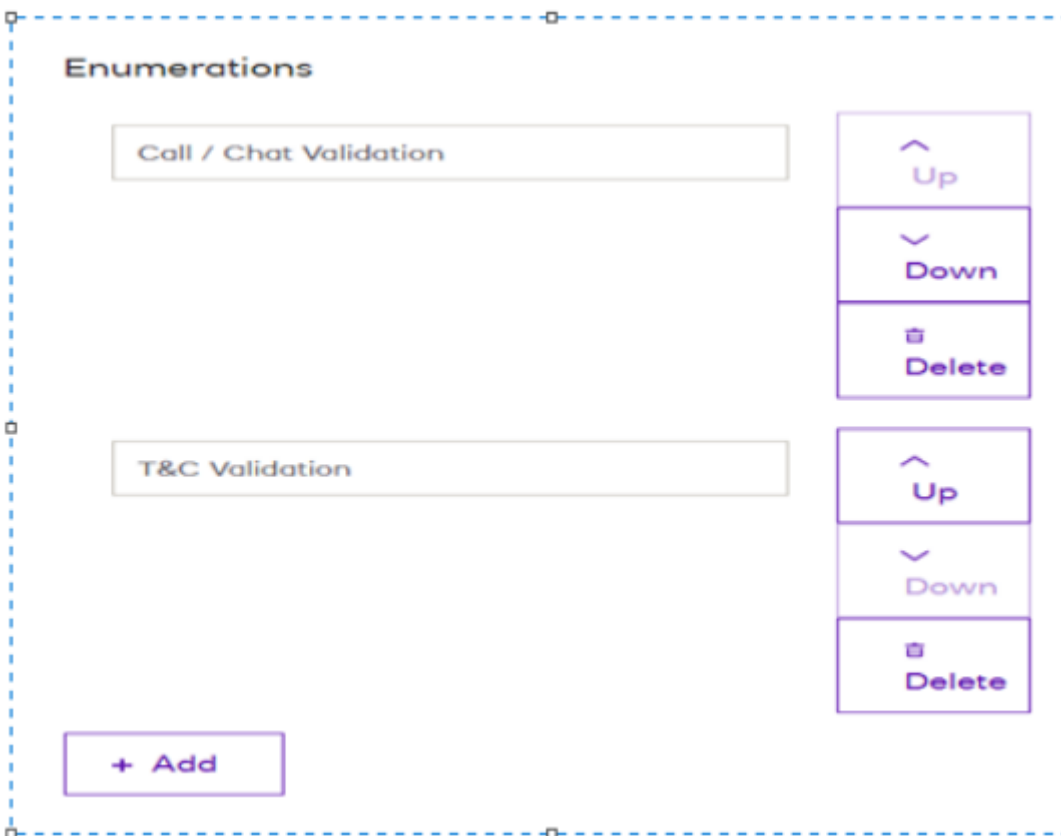
Limit Error text

For customize error message for max and min length

Type

number

Enum- It is the list options for the radio button, dropdown.



Enumerations

Call / Chat Validation

Up

Down

Delete

T&C Validation

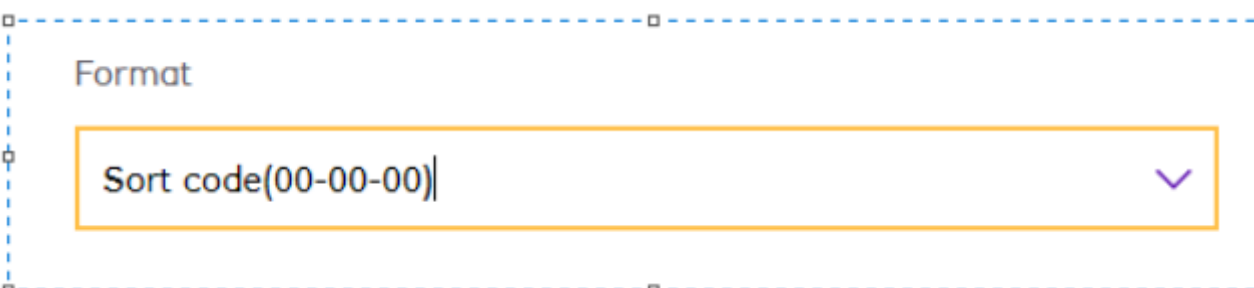
Up

Down

Delete

+ Add

format- It is the inbuilt format of input user should provide, like sort code should accept 6 digits, phone number should start with 07 and with 11 digits, date, email.



Format

Sort code(00-00-00)

requiredField- It says whether the component is mandatory or not.

Default- Its default value for the component.

schema.required- List of component which are all mandatory.

Form Properties



Inspect

Interactions

Element Name **elementName**

ba_Date_of_call

Label

Date of Call **title**

Default value (if defined) **default**

00-00-0000

Collect Data & Analytics **i**



Yes



No



Required Field ?

requiredField

8. **forms-editor-data-schema-revs**- Editor version. Archive of the collection **forms-editor-data-schema**.

9. **runtime-data-schema**- Runtime version. It stores the component details of the form, like component name, component type, component pattern.

10. **runtime-data-schema-revs**- Runtime version. Archive of the collection **runtime-data-schema**.

11. **forms-editor-ui-schema**- Editor version. Order of the components in the form.

uuid- Unique id to refer the component order.

version- Version of the component order. If any changes in the order of the component or any component is added/ removed, version will get update.

dateCreated- Date on which the form created.

schema:ui-order- It is the order of component in the form.

12. **forms-editor-ui-schema-revs**- Editor version. Archive of the forms-editor-ui-schema.

13. **runtime-ui-schema**- Runtime version. Order of the components in the form.

14. **runtime-ui-schema-revs**- Runtime version. Archive of the collection **runtime-ui-schema**.

15. **rule-definition**- Editor version. Interaction (Show & hide) of the form step. To make the form dynamic/responsive.

uuid- Unique id to refer the interaction of the step.

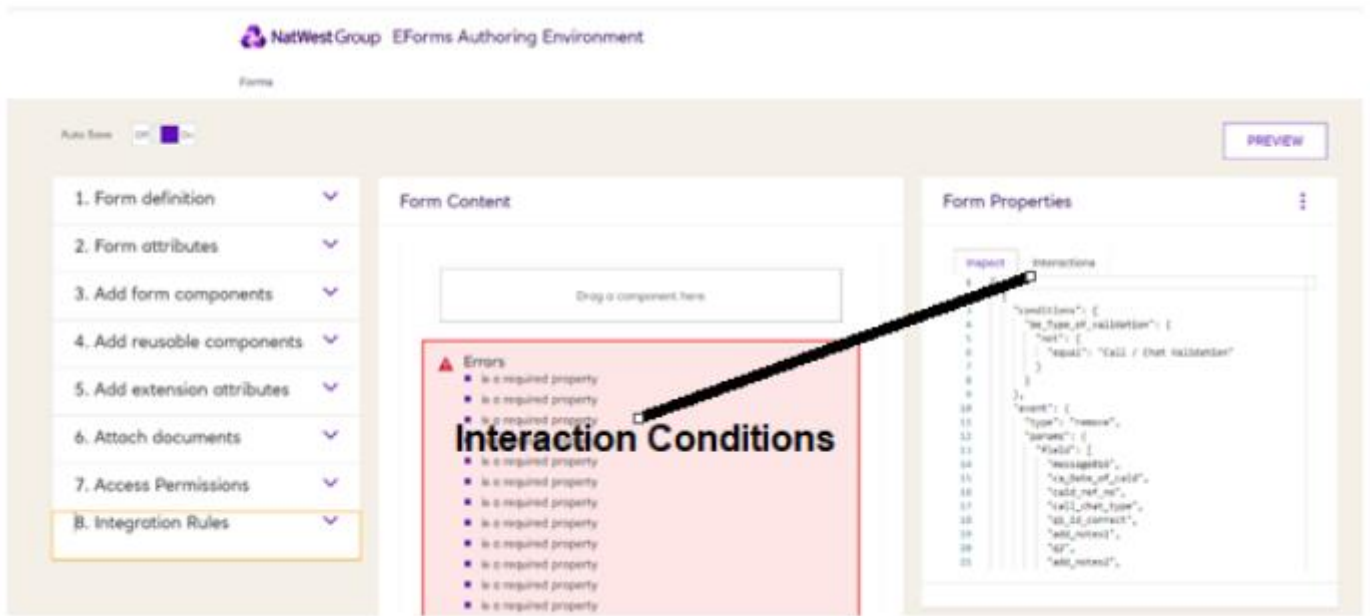
version- Version of the interaction.

dateCreated-

rules.condition - Condition of the show and hide.

rules.condition.conditions – Component condition based on that it works.

rules.condition.event- Component which needs to be hidden or shown.



Refer to understand more on interaction. [JSON \(Show/Hide etc.\) - Design a Dynamic/Responsive eForm - Colleague Capability CoE - Enablement & Effectiveness - Confluence \(atlassian.net\)](#)

16. **rule-definition-revs**- Editor version. Archive of the collection **rule-definition**.

17. **runtime-rule-definition**- Runtime version. Interaction (Show & hide) of the form step. To make the form dynamic/responsive.

18. **runtime-rule-definition-revs**- Runtime version. Archive of the collection **runtime-rule-definition**.

19. **forms-rule-template**- Editor integration data.

ruleTemplateName- It can be emails, kafka-event-partitioning, event-enrichment.

1. Emails- when data need to send through mail, 2. kafka-event-partitioning- when data need to be sent through the kafka and it is to configure the kafka topic, 3. event-enrichment- when data need to be sent through the kafka and it is to configure the component element name and the brand.

schemaFields.title- Title of the integration rule.

schemaFields.description- Description of the integration rule.

schemaFields.required- Fields which are all mandatory.

schemaFields.properties- List of fields available in the integration excel.

Ruleset for the runtime service that allows designers to enrich all events generated by adding values mapped from the form to the events themselves. For each rule select a Form Definition Attribute OR a Form Field to map the value from into the event.

Description

Download Rules To Edit Upload New Rules

No rules currently defined. Click Download Rules to Edit to add new rules.

Properties

Brand	Form Definition Attribute	Form Field
Brands *	[REQUIRED]	The list of brands that this rule is applicable for (Select from the drop down multiple times to get multiple values)

schemafields.properties.brand.description

20. **forms-editor-rule-template-data**- Editor version. Integration rules of the form. Emails, event-enrichment and kafka-event- partitioning are the 3 integration rules available in the form.

21. forms-editor-rule-template-data-revision- Editor version. Archive of the collection **forms-editor-rule-template-data**.

22. forms-rule-template-data- Runtime version. Integration rules of the form. Emails, event-enrichment and kafka-event-partitioning are the 3 integration rules available in the form.

23. forms-rule-template-data-revision- Runtime version. Archive of the collection **forms-rule-template-data**.

uuid- Unique id to refer the integration rules.

version- Version of the integration rules. Any addition/ deletion/ modification in the integration rule will affect the version.

dateCreated- Date on which form created.

dateUpdated- Date on which integration rules are modified.

ruleTemplateName- It can be emails, kafka-event-partitioning, event-enrichment.

1. Emails- when data need to send through mail, 2. kafka-event-partitioning- when data need to be sent through the kafka and it is to configure the kafka topic, 3. event-enrichment- when data need to be sent through the kafka and it is to configure the component element name and the brand.

formDefinitionReference.uuid- Form ID.

formDefinitionReference.version- Form version.

rows- 1. Emails- It says the mail id, brand and ftl file name, 2. kafka-event-partitioning- It is to say on which kafka topic data to be sent, 3. event-enrichment- It is to configure the component element name and the brand. It says to which brand which data to be sent in kafka.



24. forms-editor-user- List of users who can access the editor.

UserId- Racf ID of the users.

dateCreated- Date on which user added.

25. runtime-form-data- It stores the data of form submission.

uuid- UUID (Unique ID) of form submission. It is a reference no of the form submission.

version- It is the no of time user saves the form. Version of the form submission.

formDefinitionReference.uuid- It is the form ID.

formDefinitionReference.version- It is the form version in runtime.

status- It is the status of the form submission. It tells whether the form is submitted or not.

brand- It say on which brand the form is getting submitted like ulsterbank/ natwest/ rbs etc.

data- It is the user submitting data.

data.isSubmitted- It tells whether form submitted or not. True represents form submitted and False represents form is not submitted.

createdBy- User racf, who submitted the form data.

updatedBy- User racf, who updated the form data.

dateCreated- Date on which user submits the data of form.

dateUpdated- Date on which user update the data of form.

26. runtime-data-schema-revs- Archive of the **runtime-form-data**.

27. event-store- It has the status of the form submission. It tells whether form submitted, mail sent successfully or not and kafka sent or not.

header.created- Date on which event created.

header.schemaId- It's an event type. RULE_FIRED_EVENT. FORM_DEFINITION_LIFECYCLE_EVENT- When form submitted/ form saved.

MAIL_LIFECYCLE_EVENT- When mail event triggered.

reference.uuid- UUID of form submission or UUID of form. It is a reference no of the form submission or id of the form.

reference.type- Form submitted internally or externally.

reference.version- Form version.

body.eventClass- Status of the event. Submitted- Form submission, Created- Form saved for first time, Updated- Form saved and it is saved again and again, SEND_SUCCESS- mail sent successfully, SEND_FAILURE- mail failed to send, SENT- kafka sent successfully.

body.ruleName- When event is related to kafka rule_fired_event then it tells the fields, and to which brand the data should flow to kafka.

body.uuid- form reference id.

body.from- From mail id.

body.to- To mail id.

body.cc- CC mail id.

body.bcc- BCC mail id.

body.subject- Subject of the mail.

body.mailBody- Mail content.

28. event-schema-store- It has the list of events available in eform and its details.

29. event-store-resume-token- For every form submission JWT token is generated for each event which is stored here.

startAtOperationTime- Time token created.

token- Generated token.

service- forms-kafka-integration-service, forms-email-integration-service are the services.

eventId- mapped to _id in event-store collection.

30. shedLock- Locking time of the token. It is locking for 10 seconds.

_id- mapped to combination of service and eventId of the event-store-resume-token collection.

lockedAt- Token locking start time.

lockedUtil- Token locking end time.

Short description

Mongo Qu

Text of the article

The Below queries will provide the form names as output from Editor environment. This was created to find all the form names easily which uses a particular component.

Note : To get the output from runtime collection rename the word **editor** to **runtime** in the query like below:

["forms-editor-data-schema"] to ["forms-runtime-data-schema"]

Check a dropdown widget in a form and get the form name.

```
db["forms-editor-data-schema"].aggregate([
  {$match:{"schema.properties":{$exists:true}}},
  {$project :{_id:0,uuid:1,properties:{$objectToArray:"$schema.properties"}}},
  {$match:{"properties.v.componentType":"'dropdown'"}},
  {$lookup:
    {
      from: "forms-editor-form-definition",
      localField: "uuid",
      foreignField: "stepDefinitions.dataSchemaReference.uuid",
      as: "Dropdown"
    }
  },
  {$group :{_id:{Form:'$Dropdown.uuid'}}}
]);
```

Check a radio widget in a form and get the form name.

```
db["forms-editor-data-schema"].aggregate([
{$match:{"schema.properties":{$exists:true}}},
{$project :{_id:0,uuid:1,properties:{$objectToArray:"$schema.properties"}}},
{$match:{"properties.v.widget":"'radio'"}},
{ $lookup:
  {
    from: "forms-editor-form-definition",
    localField: "uuid",
    foreignField: "stepDefinitions.dataSchemaReference.uuid",
    as: "radio"
  }
},
{$group :{_id:{Form:'$radio.uuid'}}}
]);
```

Check a Textarea widget in a form and get the form name.

```
db["forms-editor-data-schema"].aggregate([
{$match:{"schema.properties":{$exists:true}}},
{$project :{_id:0,uuid:1,properties:{$objectToArray:"$schema.properties"}}},
{$match:{"properties.v.widget":"'TextareaWidget'"}},
{ $lookup:
  {
    from: "forms-editor-form-definition",
    localField: "uuid",
    foreignField: "stepDefinitions.dataSchemaReference.uuid",
    as: "textarea"
  }
},
{$group :{_id:{Form:'$textarea.uuid'}}}
]);
```

Check a Checkbox widget in a form and get the form name.

```
db["forms-editor-data-schema"].aggregate([
{$match:{"schema.properties":{$exists:true}}},
{$project :{_id:0,uuid:1,properties:{$objectToArray:"$schema.properties"}}},
{$match:{"properties.v.widget":"'CheckboxWidget'"}},
{ $lookup:
  {
    from: "forms-editor-form-definition",
    localField: "uuid",
```

```

        foreignField: "stepDefinitions.dataSchemaReference.uuid",
        as: "checkbox"
    }
},
{$group :{_id:{Form:'$checkbox.uuid'}}}
]);

```

Check a HtmlToTextWidget widget in a form and get the form name.

```

-----

db["forms-editor-data-schema"].aggregate([
{$match:{"schema.properties":{$exists:true}}},
{$project :{_id:0,uuid:1,properties:{$objectToArray:"$schema.properties"}}},
{$match:{"properties.v.widget":"'HtmlToTextWidget'"}},
{ $lookup:
    {
        from: "forms-editor-form-definition",
        localField: "uuid",
        foreignField: "stepDefinitions.dataSchemaReference.uuid",
        as: "html"
    }
},
{$group :{_id:{Form:'$html.uuid'}}}
]);

```

Check a Currency widget in a form and get the form name.

```

-----

db["forms-editor-data-schema"].aggregate([
{$match:{"schema.properties":{$exists:true}}},
{$project :{_id:0,uuid:1,properties:{$objectToArray:"$schema.properties"}}},
{$match:{"properties.v.widget":"'CurrencyWidget'"}},
{ $lookup:
    {
        from: "forms-editor-form-definition",
        localField: "uuid",
        foreignField: "stepDefinitions.dataSchemaReference.uuid",
        as: "currency"
    }
},
{$group :{_id:{Form:'$currency.uuid'}}}
]);

```

Check a Signature widget in a form and get the form name.

```

-----

```

```
db["forms-editor-data-schema"].aggregate([
{$match:{"schema.properties":{$exists:true}}},
{$project :{_id:0,uuid:1,properties:{$objectToArray:"$schema.properties"}}},
{$match:{"properties.v.widget":'signature'}},
{ $lookup:
  {
    from: "forms-editor-form-definition",
    localField: "uuid",
    foreignField: "stepDefinitions.dataSchemaReference.uuid",
    as: "signature"
  }
},
{$group :{_id:{Form:'$signature.uuid'}}}
]);
```

Check a Address/postcode widget in a form and get the form name.

```
-----

db["forms-editor-data-schema"].aggregate([
{$match:{"schema.properties":{$exists:true}}},
{$project :{_id:0,uuid:1,properties:{$objectToArray:"$schema.properties"}}},
{$match:{"properties.v.widget":'AddressLookupWidget'}},
{ $lookup:
  {
    from: "forms-editor-form-definition",
    localField: "uuid",
    foreignField: "stepDefinitions.dataSchemaReference.uuid",
    as: "address"
  }
},
{$group :{_id:{Form:'$address.uuid'}}}
]);
```

Check a Input String widget in a form and get the form name.

```
-----

db["forms-editor-data-schema"].aggregate([
{$match:{"schema.properties":{$exists:true}}},
{$project :{_id:0,uuid:1,properties:{$objectToArray:"$schema.properties"}}},
{$match:{"properties.v.type":'string'}},
{ $lookup:
  {
    from: "forms-editor-form-definition",
    localField: "uuid",
    foreignField: "stepDefinitions.dataSchemaReference.uuid",
    as: "input"
  }
},
{$group :{_id:{Form:'$input.uuid'}}}
]);
```

```
    }
  },
  {$group :{_id:{Form:'$input.uuid'}}}
]);
```

Check a Date widget in a form and get the form name.

```
db["forms-editor-data-schema"].aggregate([
  {$match:{"schema.properties":{$exists:true}}},
  {$project :{_id:0,uuid:1,properties:{$objectToArray:"$schema.properties"}}},
  {$match:{"properties.v.format":"'date'"}},
  { $lookup:
    {
      from: "forms-editor-form-definition",
      localField: "uuid",
      foreignField: "stepDefinitions.dataSchemaReference.uuid",
      as: "Date"
    }
  },
  {$group :{_id:{Form:'$Date.uuid'}}}
]);
```

Check a List widget in a form and get the form name.

```
db["forms-editor-data-schema"].aggregate([
  {$match:{"schema.properties":{$exists:true}}},
  {$project :{_id:0,uuid:1,properties:{$objectToArray:"$schema.properties"}}},
  {$match:{"properties.v.elementName":"'List'"}},
  { $lookup:
    {
      from: "forms-editor-form-definition",
      localField: "uuid",
      foreignField: "stepDefinitions.dataSchemaReference.uuid",
      as: "List"
    }
  },
  {$group :{_id:{Form:'$List.uuid'}}}
]);
```

Check a Group widget in a form and get the form name.

```
db["forms-editor-data-schema"].aggregate([
{$match:{"schema.properties":{"$exists:true}}},
{$project :{_id:0,uuid:1,properties:{$objectToArray:"$schema.properties"}}},
{$match:{$and:
[{"properties.v.type":'object'},{"properties.v.widget":{"$ne:'summary'}},{"properties.v.elementName":{"$ne:'phoneNumber'}}]}},
{ $lookup:
    {
        from: "forms-editor-form-definition",
        localField: "uuid",
        foreignField: "stepDefinitions.dataSchemaReference.uuid",
        as: "group"
    }
},
{$group :{_id:{Form:'$group.uuid'}}}
]);
```