Register form structure

The application forms are generated by a json file named register.json which contains a large object with properties registerForm and certificateCollectorDefinition. The property registerForm holds the field definitions which are used only for application forms where the latter one holds the data for certificate collector forms. Each of those two properties has definitions for forms separated by properties of corresponding events. That means, `registerForm.birth` holds the form field definitions only for application forms of the birth event whereas certificateCollectorDefinition.death stands for certificate collector forms of death event.

So, the overall hierarchy looks like,

```
{
  "registerForm": {
    "birth": {
      // Birth application form definitions
    },
    “death”: {
      // Death application form definitions
    }
  },
  “certificateCollectorDefinition”: {
    "birth": {
      // Certificate collector form definitions for birth
    },
    “death”: {
      // Certificate collector form definitions for death
    }
  }
}
```

Before proceeding, let us have a brief idea about the data types we are using in the forms.

Data types

We are using some data types and they maintain a hierarchical relation between themselves. These types are currently defined in `/packages/client/src/forms/index.ts`

1. **IForm:** `IForm` lies in the highest level in the hierarchy of form field definitions. It has an array of "`IFormSection`" in the `sections` property. For instance, `registerForm.birth` is a `IForm`.
2. **IFormSection:** `IFormSection` holds all the field definitions of a corresponding "section". A form has several sections and they are important for structuring form data. The sections are ordered in accordance with a visual representation. So, they play an important role in both the data and the view layer of the form. An `IFormSection` has several properties.

   Note: Question mark beside the property means it is optional, otherwise the property is mandatory.

   a. **id:** Unique identifier of a section within a `IForm`. Usually it indicates the property where all section data should consist when submitting. It can also be named otherwise with the help of a section-wise mapping.
   b. **viewType:** `viewType` describes whether the form should be represented as a form input or only view (in our case, `RegisterForm` / `CollectorForm` or `ReviewForm` / `PreviewForm`). It has four possible values: form, `review`, `preview` and `hidden`.
   c. **title:** Title shown above the form section.
   d. **optional?:** A section can be optional. It indicates whether a section is optional or not.
   e. **name:** It is used to hold the title of the corresponding tab if the section has any.
   f. **groups:** Consists of an array of groups. A group is used as a view group for fields and displays a part of a section. It only has presentational significance and no impact on section data. Groups have a type of `IFormSectionGroup`.
   g. **mapping?:** Stands for section-wise mapping and has a type of `IFormSectionMapping`. More will be discussed in the mapping section.
   h. **disabled?:** Indicates whether the section is disabled or not.
   i. **hasDocumentSection?:** It tells whether the section has a corresponding document section or not.
   j. **notice?:** Used to display a warning text under the label.
3. **IFormSectionGroup:** As earlier mentioned, a group is used as a view group of fields and shows a part of a section. A group is visually equivalent to a partial section. It holds field definitions of that part of the section. So, a section does not have fields as its direct children rather it has to have groups that contain several fields as direct children. It also has some features like `previewGroup`, `conditionals`. An IFormSectionGroup has `id` as its unique identifier. It has a property named `fields` in which an array of `IFormField` is provided. Actually, `IFormField` is a union of all fields. That means any field is an `IFormField`; it can be `TEXT`, `TEL`, `RADIO_GROUP`, etc. `IFormField` inherits `IFormFieldBase` type.

It has a `preventContinueIfError` prop. It has the boolean value `true` users cannot get past this view group with a validation error or left out a required field. This group consists of several fields which are mentioned below.

4. **`IFormFieldBase`:** This type is the base type of all fields. That means any property of this type is applicable to all fields. It has the following properties.
    a. **`name`:** Identifier of the field. Typically unique for one group but sometimes it is okay to have a group that can have fields having the same names if they do not appear at the same time.
    b. **`type`:** Input type of the field. It can be `TEXT`, `SELECT_WITH_OPTIONS`, etc.
    c. **`label`:** The label for the input field.
    d. **`tooltip?`:** Tooltip text for the field.
    e. **`validate`:** This prop holds an array of validator function references from the codebase. A reference is just an object with the `operation` and optional `parameters` prop. That means the field value is processed through that function and gives errors if validation fails.
    f. **`required?`:** If the field is required then set this to true. So, when the field is left unfilled, it gives a validation error.
    g. **`prefix?`:** When there is a text to show at the start of input, this prop is used.
    h. **`postfix?`:** It is used when there is a text to show at the end of the input. For instance, for taking input of a person's weight we have to show the unit, which is one of the use cases where `postfix` can be applied.
    i. **`disabled?`:** The field is disabled when it is set to true.
    j. **`initialValue?`:** When the field requires a value during generation before input, this prop is used.
    k. **`extraValue?`:** Actually this property is needed when we need more information for mapping/ tracking with the original value of the field. Currently, we are using it in the document field, nested field and reasons not applying fields. The reasons for extraValue is like -
        - Suppose we need to upload a national id doc for mother/father but we have only the value like national_id. How could we track whose dock is this? So we need to pass an extraValue like mother father/vice-versa with the field itself so the corresponding mapping could recognize whose document is this.
        - For nestedFields with `nestedRadioFieldTransformer` mapping, we need to pass optionValue as extraValue to recognize which option is selected. The reason is as all the options in nestedField are processed during the mutation mapping so we need to stop processing other options that are not selected.
        - The same sort of logic for reasons fields also because we need to know who is not applying for what. That is why we need to pass father/mother/etc as extraValue.

l. **`conditionals?:`** It keeps an array of `IConditionals`. `IConditional` is an object where it has a prop named `expression` which is a text containing a javascript expression. The condition lies in that expression. It is evaluated (with `eval`) when rendering the fields and conditionally shows/hides the field.
m. **`description?:`** Description text to show under the label.
n. **`placeholder?:`** Placeholder text for the input.
o. **`mapping?:`** It contains an object of `IFormFieldMapping` type. As the name goes, the property does the job of mapping data when submission (mutation) and retrieval (query) at field level. The reason for the mapping function is that the format of the data in graphql is different than the format of the form and the data of the form needs to be transformed. It has two properties: `query` and `mutation`.
    i. `query:` Name of the query transformer method in `operation` property.
    ii. `Mutation:` Name of the mutation transformer method in `operation` property.
p. **`hideAsterisk?:`** Required fields always show a red asterisk. This prop hides the asterisk even if the field is required.
q. **`hideHeader?:`** It hides the label. It can be used when a group has only one field and we want to show the group title only.
r. **`mode?:`** This prop indicates theme mode. The container component where the input field is implemented may be in a dark or light theme. This property enables theme mode to be passed to the field to adapt styles accordingly.
s. **`hidden?:`** This property is set to true if we want the field to be hidden.
t. **`previewGroup?:`** This prop takes the id of the preview group. An IFormSectionGroup can have multiple preview groups in its previewGroups property. Each preview group object has its respective ids.
u. **`nestedFields?:`** This field accepts an object map of field arrays that and the keys of the object are all possible values of that field. That means nested fields appear below the field when the corresponding key is the value of the field.
v. **`hideValueInPreview?:`** Hides the value on the review page.
w. **`ignoreBottomMargin?:`** If we need to remove space between any 2 fields then we need to set this prop value to true in the first field. If there are 3 fields we need to set the value to true in the first two fields. Actually it set value fields bottom margin to 0px if set to true.
x. **`readonly?:`** If this is set true, users cannot change its value in the review page.
y. **`hideInPreview?:`** Setting this property true hides this field in the review page, although the value still persists in the form data.
z. **`ignoreNestedFieldWrappingInPreview?:`** If it is set to true, it ignores nested field wrapping when transforming the value.
aa. **`ignoreFieldLabelOnErrorMessage?:`** It is set to true, the fields of a preview group do not show their respective error messages if they have, they show a collective error message instead.

Properties of `register.json`

Going through all properties of `register.json` below

`registerForm`

This object has all the form definitions of corresponding events i.e. `birth`, `death`. This object only applies for event application forms.

`birth`

This object holds sections for birth application forms. All sections are contained in `sections` prop. The sections are ordered as shown below.

1. `registration`

   The registration section has several groups as listed below.

   - `applicant-relation`

     It is a conditional group and it is only visible when someone else is applying for birth (in other words, `presentAtBirthRegistration` equals "`OTHER`" in draft). This condition is defined in the `conditional` prop.

     - `applicant`

       This field has a type of `RADIO_GROUP_WITH_NESTED_FIELDS`. It has a property `size` and its value `large` makes it appear bigger than the regular radio option. Similar to radio group but it has an additional feature that shows or hides nested input fields upon selected radio option. The map of the fields against selected values is defined in `nestedFields` property. For instance, for selected value `OTHER_FAMILY_MEMBER` or `OTHER` it has an array of fields and `otherRelationShip` is the only member of the array. For other selected options as there are no fields to show, they have an empty array on the map.

       - `OTHER_FAMILY_MEMBER -> otherRelationShip`
         This is a text input field. This field value is filled up when the applicant has any other relationship with the child in the family but not in the provided options. This field value is mapped to `informant.otherRelationShip` in graphql. Having a look at `changeHirerchyMutationTransformer` (for mutation) and

changeHirerchyQueryTransformer (for query) which are mentioned in the mapping prop will help to get the idea of how the field value gets transformed to graphql payload.

        - OTHER -> otherRelationShip
          This field is applicable when the applicant has any other relationship with the child outside the family. This field also maps to informant.otherRelationShip in graphql. The transformation methods are defined in the mapping prop for both query and mutation.

- primary-applicant

  This is also a conditional view group of fields which is visible only when the applicant is both parents (presentAtBirthRegistration equals "BOTH_PARENTS" in draft). The purpose of this group is to choose the primary applicant between both parents. It has two fields under it.

  - applicant

    This is also a radio group with nested fields. It has only two options (MOTHER and FATHER) which indicates the primary applicant. It maps to informant.relationship in graphql.

- contact-view-group

  This view group is intended for deciding the contact point of the application. The contact point of the application means the person who will be notified when the birth registration has been sent for review or registered. The fields under this view group are mentioned below.

  - contactPoint

    This is a radio group with nested fields. It has some conditional options. When the applicant is a legal guardian (the user selects option LEGAL_GUARDIAN in the applicant field), the options only APPLICANT, OTHER are displayed. If the applicant is one of the parents (the user selects MOTHER or FATHER in the applicant field), the options MOTHER, FATHER and OTHER displayed. If the applicant is anyone else, all options are displayed (APPLICANT, MOTHER, FATHER, OTHER). All these conditions are defined in conditional prop in respective options. This field maps to registration.contact in graphql.
    The nested fields after selecting each option is described below.

        - APPLICANT -> registrationPhone
          When the applicant is a legal guardian or anyone other than the parents and contact point is selected to the applicant, the registrationPhone field is displayed to

take input of the phone number. It maps to `registration.contactPhoneNumber` in graphql.

- `MOTHER` -> `registrationPhone`
  When the primary contact point is `MOTHER`, the field shows up. It also maps to `registration.contactPhoneNumber` in graphql.
- `FATHER` -> `registrationPhone`
  When the primary contact point is `FATHER`, the field shows up. It also maps to `registration.contactPhoneNumber`.
- `OTHER` -> `contactRelationshipOther`
  When the primary contact point is anyone other than the applicant or parent, to keep a record of relationship, this text input appears. Maps to `registration.contactRelationshipOther` in graphql.
- `OTHER` -> `registrationPhone`
  When the primary contact point is another person, below `contactRelationshipOther` field, this `registrationPhone` is shown. As like before, it also maps to `registration.contactPhoneNumber` in graphql.

2. `child`

   This section takes input for the information of the child. It has only one view group `child-view-group`.

   - child-view-group

     This view group consists of several fields.

     - `firstNames`

       First name in the locale of the selected country environment.

     - `firstNamesEng`

       First name in English.

     - `familyName`

       Family name or surname in the locale of the selected country environment.

- `familyNameEng`

  Family name or surname in English.

- `gender`

  Gender of the child.

- `childBirthDate`

  Birthdate of the child (DD-MM-YYYY).

- `attendantAtBirth`

  Person who attended the birth event.

- `birthType`

  Type of birth. I.e. single or twin etc.

- `multipleBirth`

  Order of birth among children of the parents.

- `placeOfBirth`

  Place of the birth event. It can be a health facility (hospital/clinic), private home or any other place.

- `birthLocation`

  It is a search field which appears if the `placeOfBirth` is a health facility. This field searches among the facilities data we have and enables us to select the right place for the birth event.

Address fields

The following fields are address fields. If the `placeOfBirth` is a private home or any other location, we need to get the address of that location. Addresses might be layered by several address lines. So, we have separate inputs for address lines. They maintain a cascading appearance by their own `conditional` properties.

- `country`

  Country of the place of birth.

- `state`

  State of the place of birth.

- `district`

  District of the place of birth.

- `addressLine4`

  Highest level of address lines of the place of birth address, i.e. city.

- `addressLine3`

  This is a conditional field that only appears if the `addressLine4` is not an urban address. It usually stands for municipalities.

- `addressLine3CityOption`

  This is a conditional field that only appears if the `addressLine4` is an urban address. It usually stands for which ward the address has.

- `addressLine2`

  This is a conditional field that appears only when `addressLine4` is a rural address. Stands for area/region names.

- `addressLine1CityOption`

  Applicable only for urban addresses. Means the primary addresses i.e.plot number or house number.

- `postCodeCityOption`

  Postcode number for urban address.

- `addressLine1`

  Primary address for rural address. It can be a house name, village name etc.

- `postCode`

  Postcode for rural address.

The following address fields function in a similar way as above. When the application environment country is not the same as the default country, these fields appear instead of the above-mentioned fields.

    - internationalState
    - internationalDistrict
    - internationalCity
    - internationalAddressLine1
    - internationalAddressLine2
    - internationalAddressLine3
    - internationalPostcode

3. informant

   This section takes information of the applicant if he/she is a legal guardian. This section has only one view group `informant-view-group`.

   - informant-view-group

     This is a conditional view group and it only appears when the applicant is a legal guardian (the user selects option `LEGAL_GUARDIAN` in the applicant field). This is the only view group of the `informant` section, so if the condition is not fulfilled the entire section is not visible.

     - idType

       This is a drop-down for taking input of which type of ID the applicant is going to provide. I.e. national ID, driver's license etc.

     - applicantID

       ID number of the selected ID type.

     - fetchButton

       It is a conditional button which only appears when `idType` is `NATIONAL_ID`. It fetches all the information of the person with the ID provided in input and fills up all fields of the form.

     - nationality

       Nationality of the informant.

     - applicantFirstNames

       First names of the informant in locale.

     - applicantFamilyName

       Family name of the informant in locale.

    - `applicantFirstNamesEng`

      First names of the informant in english.

    - `applicantFamilyNameEng`

      The family name of the informant in English.

      Below are the address fields, work similarly like mentioned earlier. All these fields stand for the address where the informant resides.

    - `permanentAddress`
    - `statePermanent`
    - `districtPermanent`
    - `addressLine4Permanent`
    - `addressLine3Permanent`
    - `addressLine3CityOptionPermanent`
    - `addressLine2Permanent`
    - `addressLine1CityOptionPermanent`
    - `postCodeCityOptionPermanent`
    - `addressLine1Permanent`
    - `postCodePermanent`
    - `internationalState`
    - `internationalDistrict`
    - `internationalCity`
    - `internationalAddressLine1`
    - `internationalAddressLine2`
    - `internationalAddressLine3`
    - `internationalPostcode`

4. `primaryCaregiver`

   This is a conditional section which is only visible when the `applicant` is `LEGAL_GUARDIAN` or `OTHER`. Although the sections do not have `conditional` prop if all the groups of that section share the same condition in their `conditional` properties, it will make the entire section conditional.

   - `parent-details-view-group`

     This view group takes input on whether the applicant has the mother or father details. It has only one field.

- `parentDetailsType`

  It is a radio group for taking input whose details the applicant has between the parents. The options are `MOTHER_ONLY`,`FATHER_ONLY`, `MOTHER_AND_FATHER`, `NONE`. The options are self-descriptive.

- `parent-not-applying-view-group`

  This view group takes the reason for the parents not applying.

  - `reasonMotherNotApplying`

    Text input, to take input of the reason for the mother not applying if the mother is still alive.

  - `motherIsDeceased`

    Checkbox, checked if the reason for the mother is not applying is that she is deceased.

  - `reasonFatherNotApplying`

    Text input, to take input of the reason for the father not applying if the father is still alive.

  - `motherIsDeceased`

    Checkbox, checked if the reason for the father is not applying is that he is deceased.

- `caregiver-details-view-group`

  This view group stands for the primary caregiver of the child. It has only one field.

  - `primaryCaregiverType`

    It is a radio option with nested fields. The options are `MOTHER`, `FATHER`, `LEGAL_GUARDIAN`, `OTHER`, `INFORMANT`. If the mother or the father or both are absent because of death, the option `MOTHER`,`FATHER` or both do not appear.

    If options `LEGAL_GUARDIAN` or `OTHER` is selected the fields `name`, `phone`, `reasonNotApplying` appear as nested fields.

5. `mother`

   This section takes information about the mother of the child. It has only one conditional view group which is `mother-view-group`, causing the entire section conditional.

- `mother-view-group`

  This view group appears only when the applicant has mother details (in other words, the applicant selects `MOTHER_ONLY` or `MOTHER_AND_FATHER` in `parentDetailsType` field). This view group consists of several fields

  - `motherBirthDate`

    It is a date field, which takes the input of the birth date of the mother.

  - `iDType`

    Dropdown for id type selection of the mother.

  - `iD`

    Input for the id number of the selected id type of the mother.

  - `fetchButton`

    Button to fetch all information about the mother by id.

  - `nationality`

    Dropdown input for the mother's nationality selection.

  - `firstNames`

    Input for the first names in the country locale of the mother.

  - `familyName`

    Input for the family name in the country locale of the mother.

  - `firstNamesEng`

    Input for the first names in English of the mother.

  - `familyNameEng`

    Input for the family name in English of the mother.

  - `maritalStatus`

    Input for the marital status of the mother.

  - `dateOfMarriage`

    Date of marriage of the mother.

- `educationalAttainment`

  Level of the educational qualification of the mother.

Below are the address fields. There are some classifications of the addresses.

Following address fields stand for permanent address.

- `permanentAddress`

  Label for the permanent address subsection.

- `countryPermanent`
- `statePermanent`
- `districtPermanent`
- `addressLine4Permanent`
- `addressLine3Permanent`
- `addressLine3CityOptionPermanent`
- `addressLine2Permanent`
- `addressLine1CityOptionPermanent`
- `postCodeCityOptionPermanent`
- `addressLine1Permanent`
- `postCodePermanent`

Below are international fields for a permanent address. As mentioned earlier, these international fields only appear when the country's environment is not the same as the default country.

- `internationalStatePermanent`
- `internationalDistrictPermanent`
- `internationalCityPermanent`
- `internationalAddressLine1Permanent`
- `internationalAddressLine2Permanent`
- `internationalAddressLine3Permanent`
- `internationalPostcodePermanent`

- `currentAddressSameAsPermanent`

  This is a radio group input that takes the choice of whether the current address is the same as a permanent address. If it is the same, all values from the permanent address are set into the current address.

  Following are the fields for current address

    - currentAddress
    - country
    - state
    - district
    - addressLine4
    - addressLine3
    - addressLine3CityOption
    - addressLine2
    - addressLine1CityOption
    - postCodeCityOption
    - addressLine1
    - postCode

  Following are the international fields for the current address.

    - internationalState
    - internationalDistrict
    - internationalCity
    - internationalAddressLine1
    - internationalAddressLine2
    - internationalAddressLine3
    - internationalPostcode

  .

6. father

   This section takes input from the father of the child. It has only one view group.

   - father-view-group

     This is a conditional group. When the applicant has the father details (that means, the applicant selects FATHER_ONLY or MOTHER_AND_FATHER in parentDetailsType field), this view group appears.

     It has several fields like following.

       - fathersDetailsExist

         This is a conditional radio option which checks if the father details exists or not if the parent is the applicant. If the applicant is a legal guardian and he/she does not have father details (MOTHER_ONLY or NONE in parentDetailsType).

- `fatherBirthDate`

  It is a date field, for the birthdate of the father

- `iDType`

  Input for the id type of the father.

- `iD`

  Input for the id number of selected id type of the father.

- `fetchButton`

  Fetches information of the father by id number.

- `nationality`

  Nationality of the father.

The following are the name fields of the father and work in a similar way like the mother.

- `firstNames`
- `familyName`
- `firstNamesEng`
- `familyNameEng`

- `maritalStatus`

  Marital status of the father.

- `dateOfMarriage`

  Date of marriage of the father, pre-populated if mother has given a marriage date.

- `educationalAttainment`

  The level of the educational qualification of the father.

- `addressSameAsMother`

  This is a radio option field, which checks whether the father's address is the same as mother's. If so, then the mother's address is copied into the father's address.

Following are the address fields for the father which appear only if the father has a different address than the mother's.

- currentAddress
- country
- state
- district
- addressLine4
- addressLine3
- addressLine3CityOption
- addressLine2
- addressLine1CityOption
- postCodeCityOption
- addressLine1
- postCode
- internationalState
- internationalDistrict
- internationalCity
- internationalAddressLine1
- internationalAddressLine2
- internationalAddressLine3
- internationalPostcode
- permanentAddressSameAsMother
- permanentAddress
- countryPermanent
- statePermanent
- districtPermanent
- addressLine4Permanent
- addressLine3Permanent
- addressLine3CityOptionPermanent
- addressLine2Permanent
- addressLine1CityOptionPermanent
- postCodeCityOptionPermanent
- addressLine1Permanent
- postCodePermanent
- internationalStatePermanent
- internationalDistrictPermanent
- internationalCityPermanent
- internationalAddressLine1Permanent
- internationalAddressLine2Permanent
- internationalAddressLine3Permanent
- internationalPostcodePermanent

7. `documents`

   This section is responsible for taking input of the supporting documents. It has only one view group.

   - `documents-view-group`

     This view group holds the fields for taking input of the

     - `paragraph`

       This is a paragraph which holds a short description about supporting documents and their requirements the applicant is going to provide. The supporting document requirement varies according to the child age. So, there are multiple conditional `paragraph` fields which appear in accordance with child birthdate.

     - `uploadDocForMother`

       This is a document uploader with options field which is intended to get the documents for the mother. The options are birth registration, national id (front and back) and many more as mentioned in the options prop. This uploader field only appears when the applicant has the mother's details. It can be verified by the conditional prop of this field. Fortunately, we have descriptions in the conditionals which explains a bit about the reason behind the conditional.

     - `uploadDocForFather`

       Document uploader field for the father. The options prop has the possible options for documents. Similarly, this field appears only if the applicant has father details.

     - `uploadDocForApplicant`

       Document uploader field for the applicant if someone else.

     - `uploadDocForProofOfLegarGuardian`

       If the applicant is a legal guardian, we need the proof of the legal guardianship. This field takes documents for legal guardianship.

     - `uploadDocForProofOfAssignedResponsibility`

       If the applicant is someone else and responsible for the child, the proof of assigned responsibility is needed. This field is used to take that documents.

    - `uploadDocForParentPermanentAddress`

      If the child age is between 46 days to 5 years, the parent permanent address proof is needed. This conditional field appears only when the child age is within the above mentioned limit.

    - `uploadDocForChildDOB`

      If the child age is not between 46 days to 5 years, this conditional field appears and is used to take the document of the date of birth of the child.

    - `uploadDocForChildAge`

      This conditional field appears only when the child age is more than 45 days. It used to take the document for the proof of child age.

    - `uploadDocFromCounsilor`

      This document uploader has only one option of the letter from the ward councilor. As the name goes it is used to take ward councillor proof.

`death`

This object holds sections for death application forms. All sections are in `sections` prop (like above). The order is maintained as mentioned below.

1. `registration`
    - `other-relationship-with-deceased`

      This view group appears only when the informant is other than the officer in charge, the owner of the house, driver of the vehicle or the head of the institute. To determine the other relationship this view group shows up.

        - `relationship`

          It is a radio group with nested fields. It has five options: the owner of the house, driver of the vehicle, head of the institute, office-in-charge and someone else. If someone else is selected (`OTHER` is the selected value), we need to know the relationship the informant has. So, the `otherRelationShip` nested text input appears when someone else is selected.

    - `point-of-contact`

      This view group is used to take the point of contact. It has only one field.

- `contactPoint`

  It is a radio group with nested fields. It has two options: the applicant or someone else (APPLICANT and OTHER). If the applicant is the primary contact and then only the phone number input (`registrationPhone`) pops up as a nested field and if the someone else is selected two fields show up (1. `contactRelationship` - the relationship between contact person and the deceased, 2. `registrationPhone` - phone number of that contact person).

2. `deceased`

This section covers all the information about the deceased. It has only one view group.

- `deceased-view-group`

  This view group takes the details of the deceased person.

  - `birthDate`

    Birthdate of the deceased.

  - `iDType`

    Id type of the deceased.

  - `iD`

    Id number of the selected id type of the deceased.

  - `fetchButton`

    Fetch button to fetch the details of the deceased with his/her id number.

  - `nationality`

    Nationality of the deceased.

  - `firstNames`

    First names in the country locale of the deceased.

  - `familyName`

    Family name in the country locale of the deceased.

  - `firstNamesEng`

    First names in English of the deceased.

- `familyNameEng`

  The family name of the deceased.

- `gender`

  Gender of the deceased.

- `maritalStatus`

  Marital status of the deceased.

Below are the address fields, which takes the address the deceased was used to reside.

- currentAddress
- country
- state
- district
- addressLine4
- addressLine3
- addressLine3CityOption
- addressLine2
- addressLine1CityOption
- postCodeCityOption
- addressLine1
- postCode
- internationalState
- internationalDistrict
- internationalCity
- internationalAddressLine1
- internationalAddressLine2
- internationalAddressLine3
- internationalPostcode
- permanentAddressSameAsMother
- permanentAddress
- countryPermanent
- statePermanent
- districtPermanent
- addressLine4Permanent
- addressLine3Permanent
- addressLine3CityOptionPermanent
- addressLine2Permanent
- addressLine1CityOptionPermanent
- postCodeCityOptionPermanent
- addressLine1Permanent
- postCodePermanent
- internationalStatePermanent
- internationalDistrictPermanent
- internationalCityPermanent
- internationalAddressLine1Permanent
- internationalAddressLine2Permanent
- internationalAddressLine3Permanent
- internationalPostcodePermanent

3. deathEvent

This section covers all the information about the death event. It has several groups and some of them are conditional.

- deathEvent-deathDate

  This view group is used to take input from the death date, having only one field.

  - deathDate

    This is a date field. Takes the date of the death occurrence.

- deathEvent-deathManner

  It concerns the manner of death.

  - manner

    It is a radio group. To provide information about how the death occurred, the informant should provide this field. It has five options.

    1. NATURAL_CAUSES: If the death occurred by natural causes this radio option is selected.
    2. ACCIDENT: Selected if the death occurred by accident.
    3. SUICIDE: Selected if the manner is suicide.
    4. HOMICIDE: Selected if the manner is homicide.
    5. MANNER_UNDETERMINED: Selected if the manner is not determined yet.

- deathEvent-deathPlaceAddress

  This view group concerns where the death occurred. It has only one field.

  - deathPlaceAddress

    It is a radio group. It has five options.

    1. PERMANENT: If the death occurred in the deceased's permanent address, this option is selected.
    2. CURRENT: If the death occurred in the deceased's current address, this option is selected.
    3. PRIVATE_HOME: If the death occurred in the deceased’s private home, this option is selected.
    4. HEALTH_FACILITY: If the death occurred in a health facility, this option is selected.
    5. OTHER: It is selected when the death occurred in none of the above places.

- deathEvent-deathLocation

  This conditional view group appears only when the death occurred in a health facility.

  - deathLocation

    It is a search field. It allows users to search among the offline facilities data to find out the actual health facility where death occurred.

- deathEvent-deathAtPrivateHome

  If death occurred at a private home, we need the address of the private home. So this conditional view group appears only when the earlier mentioned condition is met. This view group fields are address fields.

  - country
  - state
  - district.
  - addressLine4
  - addressLine3
  - addressLine3CityOption
  - addressLine2
  - addressLine1CityOption
  - postCodeCityOption
  - addressLine1
  - postCode
  - internationalState
  - internationalDistrict
  - internationalCity
  - internationalAddressLine1
  - internationalAddressLine2
  - internationalAddressLine3
  - internationalPostcode
- deathEvent-deathAtOtherLocation

  If death occurred at any other location, we need the address of that location. So this conditional view group appears only when the earlier mentioned condition is met. This view group fields are address fields.

    - country
    - state
    - district.
    - addressLine4
    - addressLine3
    - addressLine3CityOption
    - addressLine2
    - addressLine1CityOption
    - postCodeCityOption
    - addressLine1
    - postCode
    - internationalState
    - internationalDistrict
    - internationalCity
    - internationalAddressLine1
    - internationalAddressLine2
    - internationalAddressLine3
    - internationalPostcode

4. causeOfDeath

   This section concerns the cause of death.

   - causeOfDeath-causeOfDeathEstablished

     This view group checks if a cause of death is established. It contains only one field.

     - causeOfDeathEstablished

       This is a radio group field. This field queries whether a cause of death established or not having two options: yes, no.

   - causeOfDeath-methodOfCauseOfDeathSection

     This conditional view group shows up only if there is an established cause of death.

     - causeOfDeathCode

       It is a text input for taking the cause of death code as the cause of death is established.

5. informant

   This section concerns the informant of the death event which has only one group.

- `informant-view-group`

  This view group has the fields for taking the information about the informant.

  - `applicantBirthDate`

    Date field, stands for the birth date of the informant.

  - `iDType`

    Id type of the informant.

  - `applicantID`

    Id number of the selected id type of the informant.

  - `fetchButton`

    Fetch button for fetching information of the informant by the person's id.

  - `nationality`

    Nationality of the informant.

  - `applicantFirstNames`

    First names in the country locale of the informant.

  - `applicantFamilyName`

    Family name in the country locale of the informant.

  - `applicantFirstNamesEng`

    First names in English of the informant.

  - `applicantFamilyNameEng`

    Family name in English of the informant.

  - `relationship`

    Relationship of the informant to the deceased.

  Rest is the address fields, meaning the address of the informant's residence.

    - country
    - state
    - district.
    - addressLine4
    - addressLine3
    - addressLine3CityOption
    - addressLine2
    - addressLine1CityOption
    - postCodeCityOption
    - addressLine1
    - postCode
    - internationalState
    - internationalDistrict
    - internationalCity
    - internationalAddressLine1
    - internationalAddressLine2
    - internationalAddressLine3
    - internationalPostcode

6. father

   This section has only one conditional view group making the entire section conditional.

   - father-view-group

     This conditional view group only shows up when the informant is not the father of the deceased. This view group has four fields.

     - fatherFirstNames

       First names in the country locale of the deceased's father.

     - fatherFamilyName

       Family name in the country locale of the deceased's father.

     - fatherFirstNamesEng

       First names in English of the deceased's father.

     - fatherFamilyNameEng

       Family name in English of the deceased's father.

7. mother

   Like the `father` section, this section also has only one conditional view group making the entire section conditional.

- `mother-view-group`

  This conditional view group only shows up when the informant is not the mother of the deceased. This view group has four fields too.

  - `motherFirstNames`

    First names in the country locale of the deceased's mother.

  - `motherFamilyName`

    Family name in the country locale of the deceased's mother.

  - `motherFirstNamesEng`

    First names in English of the deceased's mother.

  - `motherFamilyNameEng`

    First names in English of the deceased's mother.

8. `spouse`

   This section concerns the deceased's spouse. It has only one conditional view group. So, it is a conditional section.

   - `spouse-view-group`

     This view group only shows up when the informant is not a spouse. It has only one field.

     - `hasDetails`

       It is a radio group with nested fields. It queries whether the deceased person had a spouse or not. So the options are yes and no. If selected no, the user can move on to the next section. If selected yes, the name fields (`spouseFirstNames`, `spouseFamilyName`, `spouseFirstNamesEng`, `spouseFamilyNameEng`) will show up as nested fields to take input of the spouse's name.

9. `documents`

   This is the section that takes the responsibility of the supporting documents concerning the death event application.

   - `documents-view-group`

     It is the only view group of the `documents` section. This view group contains mostly fields of document uploader with options type.

- `paragraph`

  This is a paragraph, holds a descriptive message about the supporting documents requirements.

- `uploadDocForDeceased`

  This is a document uploader with options and is used to take the documents for the deceased. The document of this field actually stands for the deceased id proof. So, the `extraValue` of this is `DECEASED_ID_PROOF`. The options are `National ID (front)`, `National ID (back)`, `Birth Registration`, `Passport`.

- `uploadDocForApplicant`

  This “document uploader with options” field stands for the documents for the applicant as proof of his/her id. So the `extraValue` of this field is `APPLICANT_ID_PROOF`. The options are the same as the deceased.

- `uploadDocForApplicantAthorityToApply`

  This field is conditional and only appears when the informant is someone else. The `extraValue` of this field is `APPLICANT_ATHORITY_TO_APPLY_PROOF`. This has only one option which is `Signed Affidavit`.

- `uploadDocForDeceasedDeath`

  This field takes proof of the death of the deceased. It has several options, which are: `Certified Post Mortem Report`, `Hospital Discharge Certificate`, `Attested Letter of Death`, `Attested Certificate of Death`, `Certified Copy of Burial Receipt`, `Certified Copy of Funeral Receipt`. The `extraValue` of this field `DECEASED_DEATH_PROOF`, which is self-descriptive.

- `uploadDocForDeceasedPermanentAddress`

  This field stands for the proof of the deceased’s permanent address having `extraValue` of `DECEASED_PARMANENT_ADDRESS_PROOF`. The options are the same as the uploadDocForDeceased.

- `uploadDocForDeceasedDOB`

  This field stands for the deceased birth proof, having only one option `Proof of Date of Birth of Deceased`.

- `uploadDocFromCounsilor`

  This field stands for the ward councilor proof having only one option `Letter from ward councillor`.