opentext™

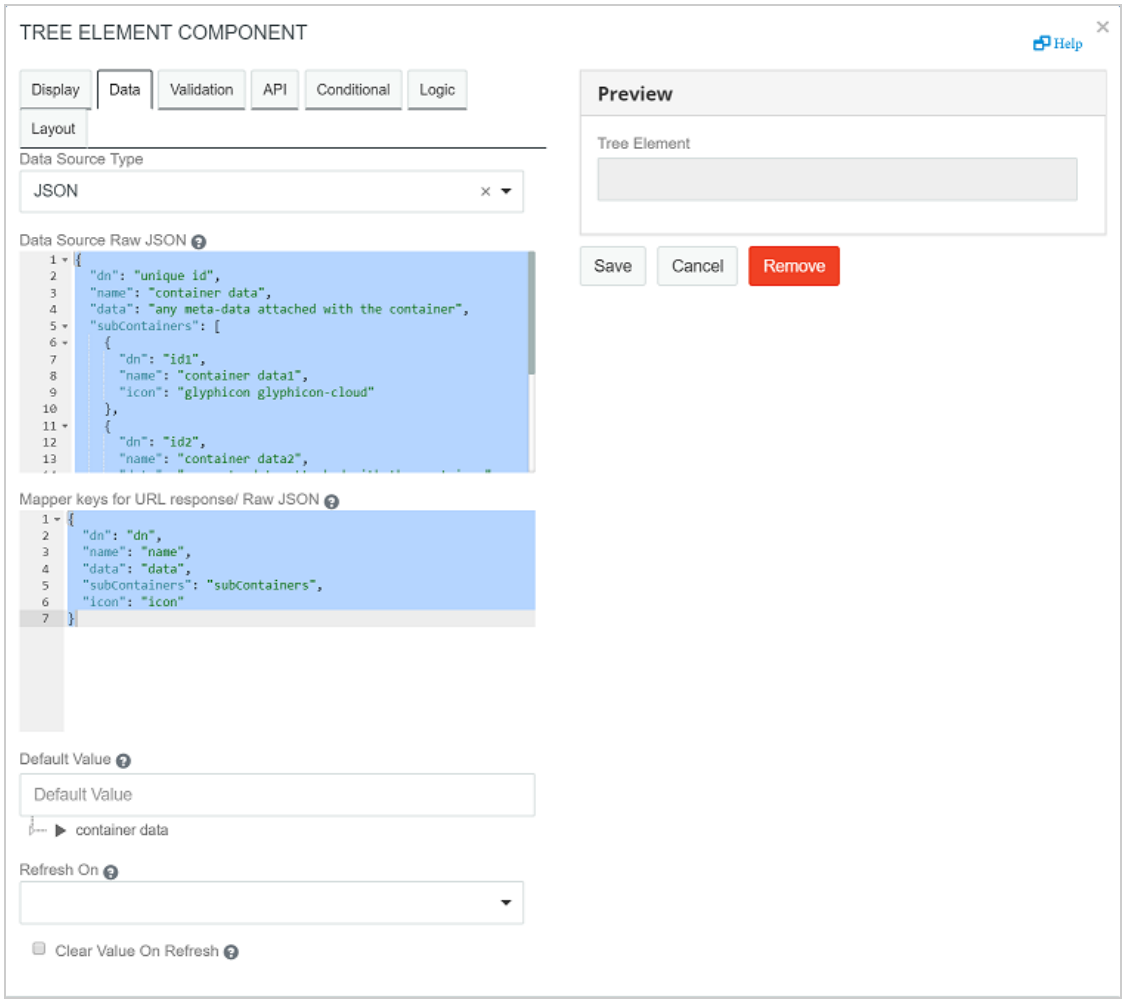
OpenText™ Formio Help

December 2024

Tree Element

The Tree element is used to design a hierarchy or a category. There are options to configure nodes that allows you to change the design of the form in the desired format.

Figure 1 *Tree Element - Data*



NOTE: The **Preview** area displays how the form would render on making the changes or edits to any field.

The following fields are populated in the **Data** tab.

Data Source Type: Select the type of data source, that is JSON or URL.

The following fields appear when the **Data Source Type** is **JSON**:

Data Source Raw JSON: Each node configuration is displayed in a JSON file. You must have the following details mentioned:

dn: It is a unique ID that defines the selected node. The value must be a string. This is the data value for the tree element that you add and is passed to the workflow.

name: Enter the name you wish to be displayed on the User Interface. The value must be a string.

data: (Optional) This saves the metadata to the existing field.

subcontainers: It defines the child node.The value must be an array of JSON. Each element of the array is an object following the same structure as that of a node.

icon: The image class to be used for displaying icon on each node. The value must be a string. If you do not define the value, the node uses the image class selected in **Default Icon Class** under the **Display** tab.

NOTE: All nodes must follow either JSON or customized data structure. The parent and child cannot be of different structure.

If JSON does not follow the default structure, it must be mapped accordingly.

Mapper keys for URL response: Allows you to map the JSON parameters with the URL element parameters.

You can use the **Mapper keys for URL response/ Raw JSON** setting to customize the field names provided under **Data Source RAW JSON**.

For example, if you want the field names as ["id","display","description","child","icon"], then modify the details under **Mapper keys for URL response/ Raw JSON** as follows:

```
{
    dn: "id",
    name: "display",
    data: "description",
    subContainers: "child",
    icon: "icon"
}
```

The following fields appear when the **Data Source Type** is **URL**:

HTTP method: Select the required http method. For example, GET, POST.

Service ID: Select the appropriate service ID. For example, IDM, IG.

Data Source URL: The URL that returns a JSON array to use as the data source.

Mapper keys for URL response: Allows you to map the JSON parameters with the URL element parameters.

Request Headers: Set any headers that should be sent along with the request to the URL. This is useful for authentication.You can have multiple keys for a parameter, and you can have multiple values for a key.

request payload: Enter the request body for the root node in the request payload field. The request body contains the node details that will be used to load the node. It applies in case of POST method.

lazy parameter: Enter the parameter name that will be used to load a sub-node. The lazy parameter such as `nodeid` is appended in the data source URL provided by the user. It applies in case of GET method (Lazy Loading).

Default Value: The entered value is displayed in the field before user interaction. Having a default value overrides the placeholder text.

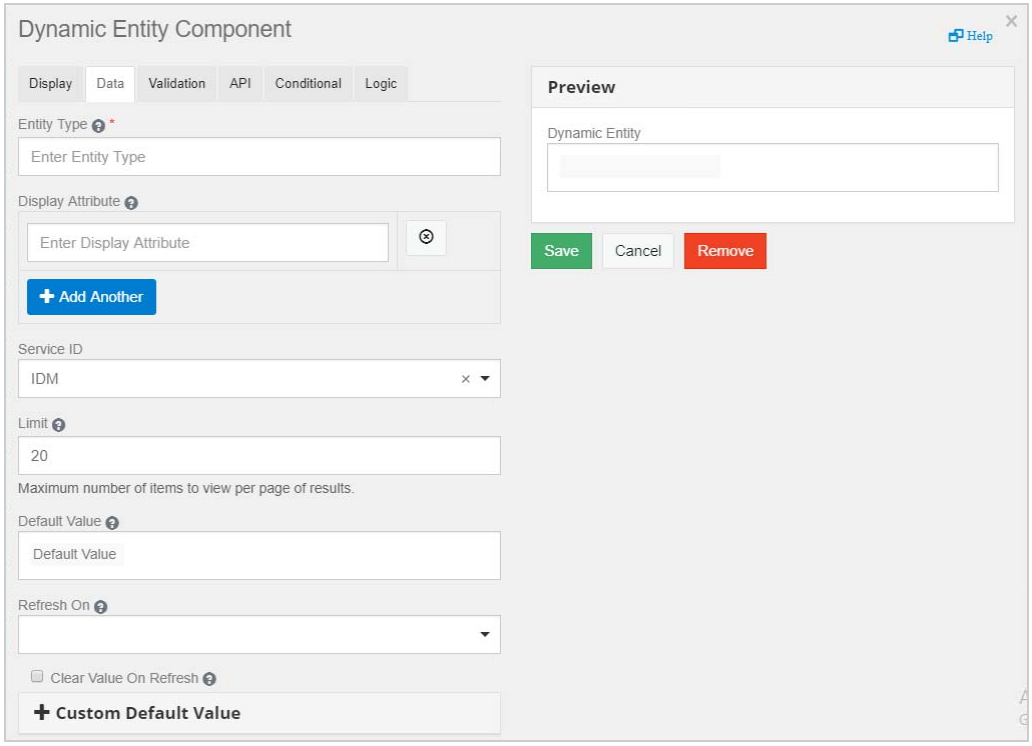
Refresh On: Refreshes data when another field changes.

Clear Value On Refresh: This text appears below the input field.

Dynamic Entity

This is a multi select drop-down field. This widget allows you to select more than one entity in an entity type. For example, user, group

Figure 2 *Dynamic Entity*



The following fields are populated in the **Data** tab:

Entity Key: Specify the entity key for the entity type. For example, for the entity type User, the entity key is user.

Display Expression Attribute: Specify the attributes of the entity type you want to be displayed. You can add multiple display attributes. For example, for the entity type User, the **Display Expression Attribute** can be FirstName, LastName.

Entity Type	Display Expression Attribute
User	FirstName, LastName

Service ID: Select the service ID. For example, OpenText Identity Lifecycle Manager or OpenText Identity Governance.

Limit: Specify the number of entities that you want to be displayed. The default value is 20.

Default Value: Specify the value to be displayed in the field before user interaction. Having a default value overrides the placeholder text.

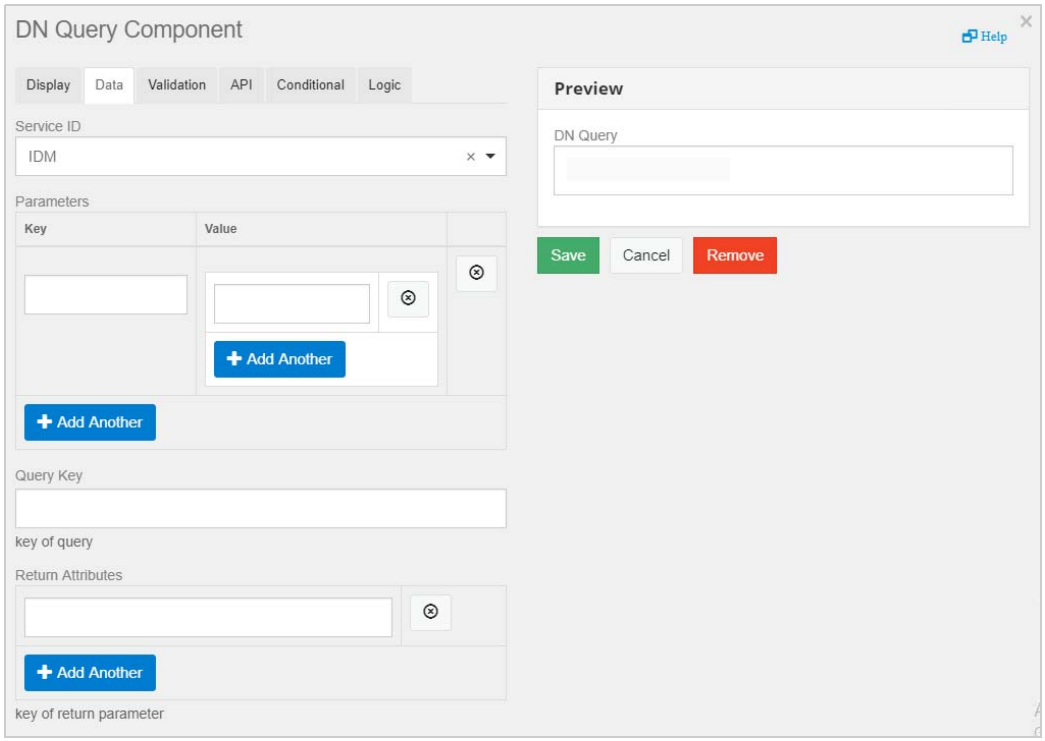
Refresh On: Refreshes data when another field changes.

DN Query

Allows you to search and retrieve DNs from the Identity Vault. However, with the DNQuery, the object selector content can be driven by the result of a directory abstraction layer Queries object rather than from properties.

NOTE: You must ensure that the query, parameter, and key you specify is present in Designer.

Figure 3 *DN Query*



The following fields are populated in the **Data** tab:

Service ID: Select the required service ID. For example, OpenText Identity Lifecycle Manager or OpenText Identity Governance.

Parameters: Specify the parameter key and its value. You can have multiple keys for a parameter, and you can have multiple values for a key.

NOTE: You must provide the parameter value (static value) while configuring this component. Dynamic parameter value such as `data.<other dynamic value>` is not supported. To use the dynamic parameter value, you can use the `IDVault.globalquery` in the **Select** component.

Query Key: Specify the key of the DAL Queries object.

Return Attributes: Specify the attributes of the entity type you want to be displayed. You can add multiple display attributes.

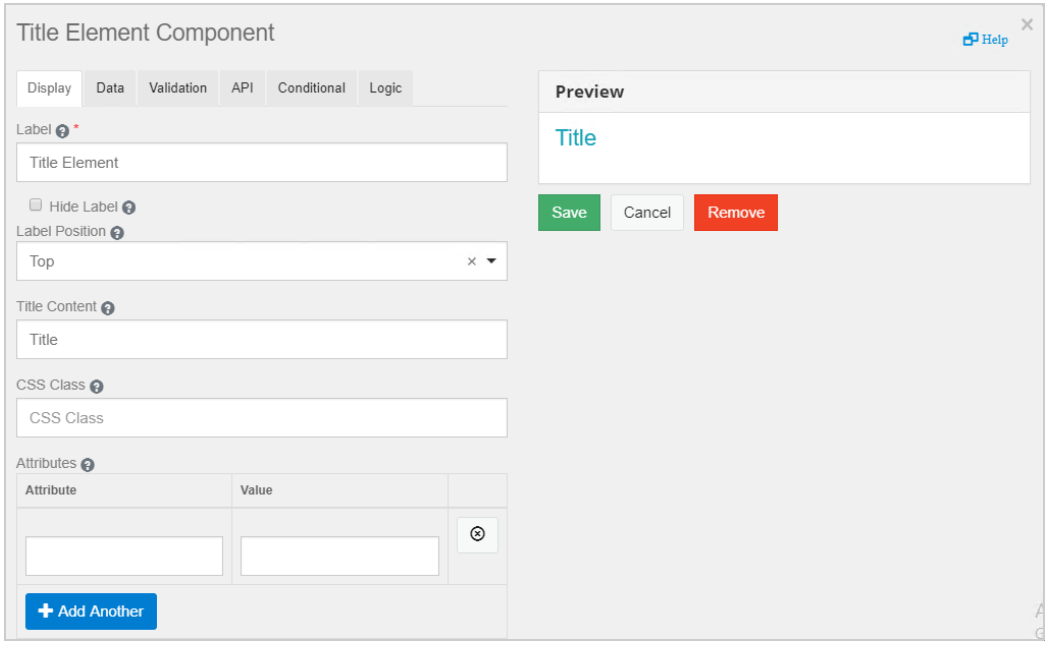
Default Value: Specify the value to be displayed in the field before user interaction. Having a default value overrides the placeholder text.

Refresh On: Refreshes data when another field changes.

Title Element

This widget is used to design a title for the form.

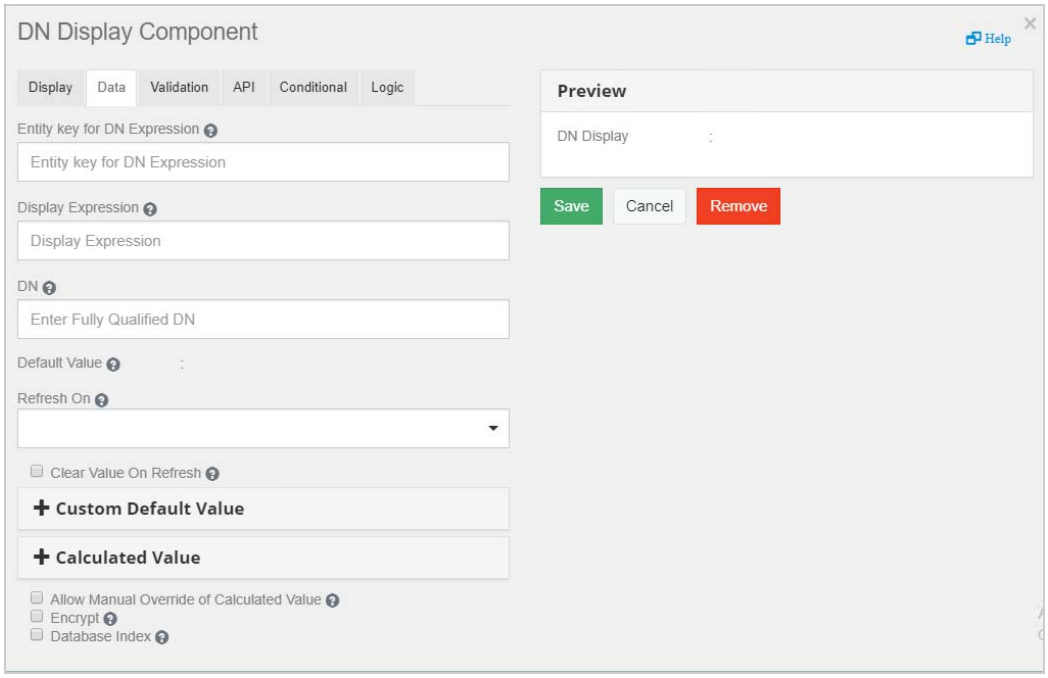
Figure 4 *Title Element*



DN Display

This is used to display a read-only DN. It can display the full DN or a set of attributes associated with the DN depending on the properties you set.

Figure 5 *DN Display*



The following fields are populated in the **Data** tab:

Entity key for DN Expression: Represents the type of entity used for DN display. Leave this value blank if you want to display the full DN or CN value retrieved from the Identity Vault, else enter an entity.

The entity you choose must:

- Have the directory abstraction layer View property set to **True**.
- Be the entity of the DN you are working with.

Display Expression: Represents the attribute used for DN display. It also displays multiple attributes, provided the attribute is separated by comma. Leave this value blank if you want to display the full DN or CN value. If you want to mask the DN by displaying attributes, you must first specify an **Entity key for DN expression.**

For example, to show the user entity's first and last name attributes, construct an expression like this: `FirstName LastName`.

Ensure the attribute's View, Read, Search, and Required properties are set to True in the directory abstraction layer. For more information, see Attribute Properties in the *OpenText™ Identity Manager - Administrator's Guide to Designing the Identity Applications*.

DN: Enter the respective DN.

Default Value: The entered value is displayed in the field before user interaction. Having a default value overrides the placeholder text.

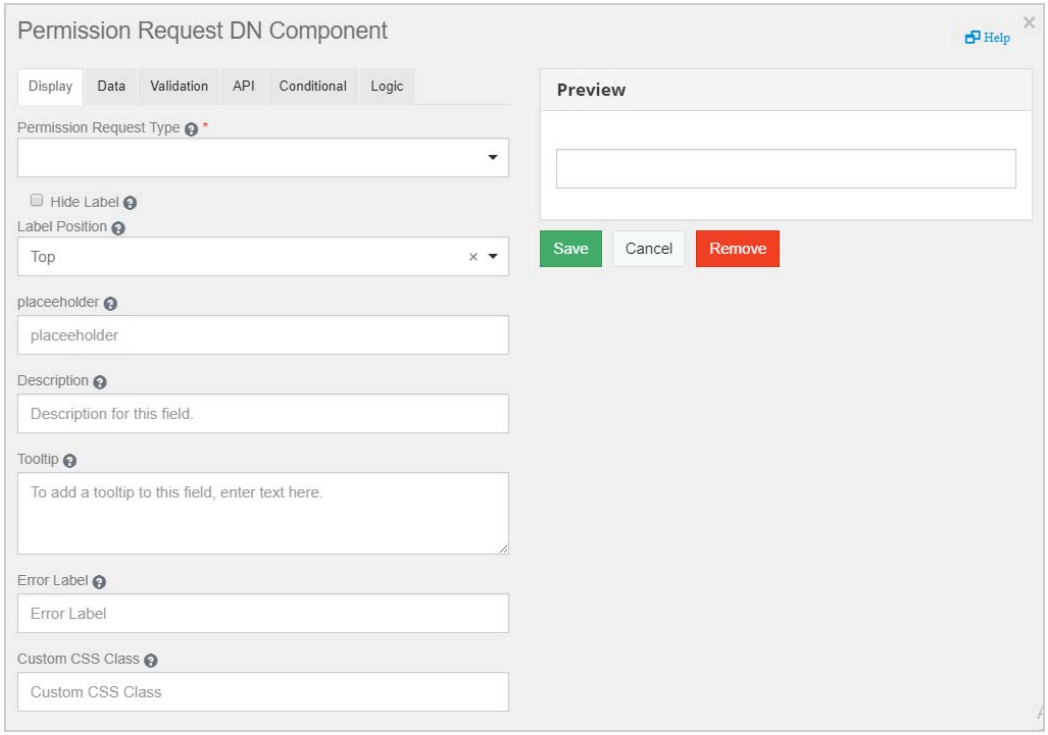
Refresh On: Refreshes data when another field changes.

Clear Value On Refresh: When Refresh On field is changed, clear this components value.

Permission Request DN

This is used to provide permission for a default role and resource approval template.

Figure 6 *Permission Request DN*

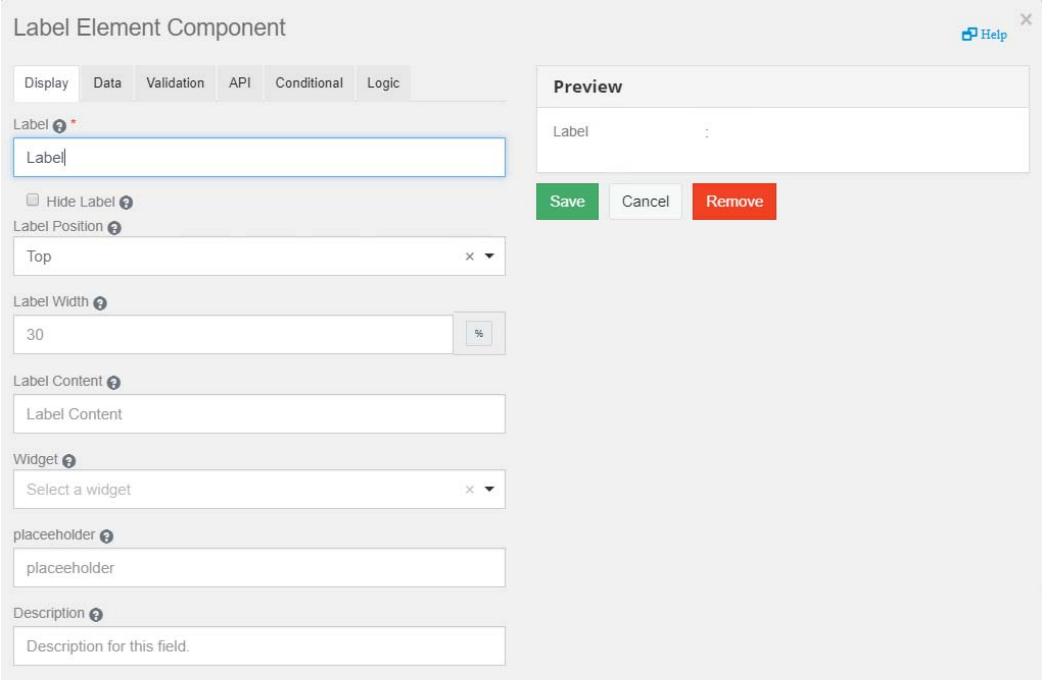


In the **Display** tab, based on the selected **Permission Request type**, the label is automatically populated.

Label Element

This widget allows you to create and design the labels used in the forms.

Figure 7 *Label Element*



Data Item Mapping Element

This widget allows you to map data from the data flow into fields in a form (pre-activity mapping) and to map data from the form back to the data flow (post-activity mapping). If you have specified the Data Item Mapping Value in OpenText Designer, the **Custom Default Value** will overwrite the **Data Mapping Value**. In other words, the **Custom Default Value** takes precedence.

Figure 8 *Data Item Mapping Element*