# Contents

## PART 1  Working with Jobs  1

**Chapter 1 / Managing Jobs**

- About Jobs .................................................. 3
- Create a Job Definition ................................. 4
- Viewing the Properties of a Job ..................... 5
- Adding Parameters ....................................... 5
- Accessing a Job via a URL .............................. 7
- Associating a Form ....................................... 8
- Schedule a Job ............................................. 10

## PART 2  Task Prompting Interface  11

**Chapter 2 / Overview of the Task Prompting Interface**  
About the Task Prompting Interface .................. 13
The Task Element ........................................ 13

**Chapter 3 / The Registration Element**  
About the Registration Element for the Prompting Interface........ 15
Example: Registration Element for Prompting Interface ............ 15

**Chapter 4 / The Metadata Element**  
About the Metadata Element ................................ 17
Working with the DataSources Element .................... 17
Working with the Options Element ....................... 26

**Chapter 5 / The UI Element**  
About the UI Element ....................................... 75
Option References ......................................... 75
Containers .................................................. 76
Example: UI Element from Sample Task .................. 77

**Chapter 6 / The OptionDependencies Element**  
About the OptionDependencies Element .................. 79
OptionDependency Element ................................ 79
Example: OptionDependencies Element ................... 80
PART 1

Working with Jobs

Chapter 1
Managing Jobs
Managing Jobs

About Jobs

A SAS Viya job consists of a program and its definition. The job definition includes information such as the job name, the author, and the creation date and time. After you have created a job definition, you have an execution URL that you can share with others at your site. The execution URL can be entered into a web browser and run without opening SAS Studio.

You can use jobs for web reporting, performing analytics, building web applications, and delivering content to clients such as SAS Studio. This client works with jobs that contain SAS code. These jobs can access any SAS data source or external file and create tables, files, or other data targets that are supported by SAS.

You can create an HTML form or task prompt to provide a user interface to the job. When the user selects an option to submit the information, the data that is specified in the form or task prompt is passed to a SAS session as global macro variables. The SAS program runs, and the results are returned to the web browser.

Note: In SAS Studio, jobs are available only in the Standard Perspective. For more information, see “Understanding Perspectives” in SAS Studio: User’s Guide.

When running a SAS job, SAS Studio uses the SAS Job Execution Web App. For more information about jobs, see SAS Job Execution Web Application: User’s Guide.
Create a Job Definition

To create a job definition in SAS Studio:

1. Select New ➔ Job ➔ Definition.

2. On the Code tab, enter the SAS code for the job definition. For example, enter the following code:

   ```sas
   proc print data=sashelp.class;
   run;
   ```

3. Save the job definition to a folder where you have Write access, such as My Folders in SAS Content.

   Note: You must save the job definition before you can run it.

4. Run the job. The output appears on the Preview tab.
Viewing the Properties of a Job

To view the properties of a job, click **E**.

From the Properties pane, you can view the name of the job, the location where the job is saved, when the job was created and the creator, and when the job was last modified and who modified it. You can also add or edit a description. The Job **submit** field lists the URL that you can share with other users so that they can run the job from a web browser. For more information, see “Accessing a Job via a URL” on page 7.

Adding Parameters

Using SAS Studio, you can also add parameters. These parameters become macro variables that are available to use in your SAS program.

1. On the **Code** tab, specify the code for the job definition. Include macro variables for each parameter that you plan to create.

Here is an example of a job definition with a macro variable:

```sas
%global title;
title "&title";
```
proc print data=sashelp.class;
run;

2 Click \(\square\) to open the Parameter pane.

3 In the **Name** field, enter the name of the parameter. For this example, enter **TITLE**.

4 In the **Default value** field, enter the default value for the parameter. For this example, enter **Listing of Class Data Set**.

5 Save the job definition, and then run the job to see the “Listing of Class Data Set” title in the output.

The SAS Job Execution Service Web App has several predefined parameters that you can use. For more information, see *SAS Job Execution Web Application: User's Guide*. 
Accessing a Job via a URL

You might want to share your job definition with other users at your site. These job consumers only need to run the job. They do not need to edit it. You can share a job with others by providing them with the Job Submit URL. You paste this URL in the address bar of a web browser, press Enter, and the results are returned in the web browser.

To determine the URL for a job, click to open the Properties pane. The URL is in the **Job submit** field and has this format: http://host/SASJobExecution/?_program=/folder-name/job-name.

The _PROGRAM parameter specifies the directory path and name for the job definition.
Associating a Form

About Forms

By default, no form is associated with the job definition. You do not need to associate a form with a definition to run a job.

However, you might want to create an interface so that the end user can specify the values to use when the job runs. At run time, the specified values are assigned to the macro variables that you define. In SAS Studio, you can associate two types of forms with a job definition: an HTML form and a task prompt.

HTML Forms

You can write HTML to create an HTML form. You can create this form in SAS Studio or it can be a separate file. The location of the HTML input form can be specified using the _FORM parameter. This parameter can be specified in the URL as an input parameter.

Here is an example of specifying the HTML input form using an input parameter:
http://host:port/SASJobExecution/form/?_program=/Folder1/myJob&_form=/Folder2/myJobForm.

In this example, the _FORM parameter identifies that the HTML input form named MyJobForm is stored in /Folder2. Note that the HTML input form does not need to be in the same folder as the job or have the same name as the job.

If the _FORM parameter is not specified, an HTML input form with the same name and location as the corresponding job is displayed.

As the author of the job definition, you can specify _FORM as a preset job parameter. This technique is useful when you have multiple jobs that need to use the same HTML input form.

For more information about how to create an HTML form and examples of HTML forms, see SAS Job Execution Web Application: User’s Guide.

Task Prompt

You can use the common task model (CTM) in SAS Studio to define job prompts.

Use this example to get started with writing your own prompts. This example creates a data source prompt. When the job runs, the user sees a prompt and can select the input data source for the job.

1 In SAS Studio, select New ➔ Job ➔ Definition.
2 Click and select **Task prompt**.

3 On the **Task prompt** tab, enter this code:

```xml
<?xml version="1.0" encoding="UTF-16"?><Task schemaVersion="7.2">
  <Registration>
    <Name>Data Source Prompt</Name>
    <Description>Example of a Data Source Prompt</Description>
    <Version>5.2</Version>
  </Registration>
  <Metadata>
    <DataSources>
      <DataSource name="DATASOURCE" where="true">
      </DataSource>
    </DataSources>
    <Options>
      <Option inputType="string" name="DATATAB">DATA</Option>
      <Option inputType="string" name="DATAGROUP">DATA</Option>
    </Options>
  </Metadata>
  <UI>
    <Container option="DATATAB">
      <Group open="true" option="DATAGROUP">
        <DataItem data="DATASOURCE"/>
      </Group>
    </Container>
  </UI>
</Task>
```

4 On the **Code** tab for the job definition, create the macro variable `&datasource` for the data source prompt. Add this code:

```plaintext
%global datasource;
proc print data=&datasource;
run;
```

5 Save the job definition and run it.

When the job runs, here is what appears on the **Task prompt**.
After you select the input data source, click Submit. The job runs and the results appear on the Preview tab.

Schedule a Job

To schedule a job:

1. In the Explorer, right-click the name of the job definition and select Schedule job.

2. In the New Trigger window, specify how frequently to run the job. You can also specify the time at which the job should run and the start date and the end date for running the job.

To view all the scheduled jobs, select View ⇒ Selected Jobs.
Chapter 2
  Overview of the Task Prompting Interface .......................... 13

Chapter 3
  The Registration Element ........................................... 15

Chapter 4
  The Metadata Element ............................................... 17

Chapter 5
  The UI Element ....................................................... 75

Chapter 6
  The OptionDependencies Element ................................. 79
About the Task Prompting Interface

The task prompting interface is an XML definition that enables you to create custom prompts.

The common task model (CTM) defines the template for the prompt. In the CTM file, you define how the prompt appears to the user. In SAS Studio, a prompt is defined by the Task element.

The Task Element

The Task element uses only one attribute:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>schemaVersion</td>
<td>Specifies the schemaVersion that is associated with the prompt.</td>
</tr>
</tbody>
</table>

The Task element has these children:

Registration

The Registration element identifies the type of prompt. In this element, you define the prompt name, description, and version.

Metadata

The Metadata element can specify whether an input data source is required to run the task, any role assignments, and the options in the task.

- The Roles element specifies the types of variables that are required by the task. Here is the information that you would specify in this element:
- type of variable that the user can assign to this role (for example, numeric or character)
- the minimum or maximum number of variables that you can assign to a role
- the label or description of the role that appears in the user interface

The Options element specifies how to display the options in the user interface.

UI

The UI element describes how to present the user interface to the user. Only a top-down layout is supported.
The Registration Element

About the Registration Element for the Prompting Interface

The Registration element represents a collection of metadata for the prompt interface. This element is required in order to know the type of prompt.

Here are the child elements for the Registration element:

<table>
<thead>
<tr>
<th>Element Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>The name of the task. This name is used throughout the application to represent the task.</td>
</tr>
<tr>
<td>Description</td>
<td>A description of the task. This text could appear in the task properties or in tooltips for the task.</td>
</tr>
<tr>
<td>Version</td>
<td>A number that represents the version of the task.</td>
</tr>
</tbody>
</table>

Example: Registration Element for Prompting Interface

```
<Registration>
  <Name>Connection Prompts</Name>
  <Description>Prompts used to connect to a CAS server.</Description>
  <Version>5.2</Version>
</Registration>
```
The Metadata Element

About the Metadata Element

The Metadata element comprises two parts: the DataSources element and the Options element.

Working with the DataSources Element

About the DataSources Element

The DataSources and DataSource elements create a simple grouping of the data that is required for the task. If these elements are not specified, then no input data is needed to run the task.

The DataSource element is the only child of the DataSources element. Most tasks need only one data source, but multiple data sources can be defined. The DataSource element specifies the information about the data set for the task.
## Attribute Description

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Specifies the name assigned to this role.</td>
</tr>
<tr>
<td>defaultValue</td>
<td>Specifies the default data source to use. If this value is not specified, SAS Studio uses the most recently used data source. If the most recently used data source cannot be determined, this value is empty. To change this behavior, use the <code>defaultValue</code> attribute.</td>
</tr>
<tr>
<td>libraryEngineExclude</td>
<td>Specifies the engine types that are not valid for the data source. The engine types should be a comma-separated list.</td>
</tr>
<tr>
<td>libraryEngineInclude</td>
<td>Specifies the engine types that are valid for the data source. Possible values include CAS and V9. Multiple engine types should be in a comma-separated list.</td>
</tr>
<tr>
<td>readOnly</td>
<td>Specifies whether the user can change the input data source. The default value is <code>false</code>, so the data can be changed. If the value is set to <code>true</code>, the value can be displayed but not changed.</td>
</tr>
<tr>
<td>where</td>
<td>Specifies whether a filter is allowed for the data. The default value is <code>false</code>, and the user cannot filter the task from the task interface.</td>
</tr>
</tbody>
</table>

**Note:** If you do not specify either the `libraryEngineExclude` parameter or the `libraryEngineInclude` parameter, all engine types are available for the data source control. If you need to limit the engine type, use either the `libraryEngineExclude` parameter or the `libraryEngineInclude` parameter. Do not specify both.

## Working with the Roles Element

### About the Roles Element

The **Roles** element is the only child of the **DataSource** element. The **Roles** element identifies the variables that must be assigned in order to run the task. This element groups the individual role assignments that are needed for a task.

The **Role** tag, which is the only child of the **Roles** element, describes one type of role assignment for the task.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Specifies the name assigned to this role.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>defaultValue</td>
<td>Specifies the default roles for this control. By default, the Role control is empty. Use the defaultValue attribute to specify a comma-separated list of variable names. If the variables do not exist in the current data source, the Role control does not show the missing variables.</td>
</tr>
<tr>
<td>type</td>
<td>Specifies the type of column that can be assigned to this role. Here are the valid values:</td>
</tr>
<tr>
<td></td>
<td>A: All column types are allowed. In the user interface, all columns are identified by the icon.</td>
</tr>
<tr>
<td></td>
<td>N: Only numeric columns can be assigned to this role. In the user interface, numeric columns are identified by the icon.</td>
</tr>
<tr>
<td></td>
<td>C: Only character columns can be assigned to this role. In the user interface, character columns are identified by the icon.</td>
</tr>
<tr>
<td>minVars</td>
<td>Specifies the minimum number of columns that must be assigned to this role. If minVars=&quot;0&quot;, the role is optional. If minVars=&quot;1&quot;, a column is required to run this task and a red asterisk appears next to the label in the user interface.</td>
</tr>
<tr>
<td>maxVars</td>
<td>Specifies the maximum number of columns that can be assigned to this role. If maxVars=&quot;0&quot;, users can assign an unlimited number of columns to this role.</td>
</tr>
<tr>
<td>exclude</td>
<td>Specifies the list of roles that are mutually exclusive to this role. If a column is assigned to a role in this list, the column does not appear in the list of available columns for this role.</td>
</tr>
<tr>
<td>order</td>
<td>Specifies that the user can order the columns that are assigned to this role. Valid values are true and false. If order=&quot;true&quot;, the user can use the up and down arrows in the user interface to modify the order.</td>
</tr>
<tr>
<td>fetchDistinct</td>
<td>Specifies whether to retrieve the distinct information for columns assigned to this role. The default value is false.</td>
</tr>
<tr>
<td>readOnly</td>
<td>Specifies whether the value can be edited. By default, the value is false, and the role can be edited. If this value is true, the current value is displayed but cannot be edited by the end user.</td>
</tr>
</tbody>
</table>

Example: DataSources and Roles Elements from the Sample Task

Here is an example of the DataSources and Roles elements from the Sample Task:

```
<DataSources>
  <DataSource name="DATASOURCE">
    <Roles>
      <Role maxVars="1" minVars="1" name="VAR">
```

When you run this code, you get the Data and Roles sections in this example:

```
<DATA>
  <SASHELP.BASEBALL/>
</DATA>

<ROLES>
  <Role name="OPTNVAR" order="true" type="N">
    Numeric variable:
  </Role>
  <Role name="OPTCVAR" order="true" type="C">
    Character variable:
  </Role>
  <Role exclude="VAR" maxVars="0" minVars="0">
    Required variable:
  </Role>
</ROLES>
```

A red asterisk appears for the **Required variable** role because you must assign a column to this role. In the code, this requirement is indicated by `minVars="1"`.

### Working with the Filters Element

#### About the Filters Element

The **Filters** tag can have one or more **Filter** children elements. The **Filters** tag is a grouping mechanism for the individual **Filter** definitions. There are no attributes associated with the **Filters** tag.

Filters enable you to populate option controls with the values from a column in the data source. A simple filter might return values from a specific column in the data source. If this column is specified in the **Filter** element, the filter is static, which means that the end user of the task is forced to use that column. If the **Filter** element is written so that the user selects the column using a Role control, the filter is dynamic, which means that the end user can choose the column before running the task.
To create a more complex filter, add expressions. An expression specifies the condition that is applied to the filter to return specific values. The expression can use values from other options or columns to dynamically create the condition. For example, you could create a filter for the value of state. This filter defines that the value of the combobox control determines which observations in the data source are returned. At run time, the end user selects the state from the list of possible values in the combobox control.

Here is the XML hierarchy for a filter:

```xml
Filters (0..1)
  Filter(1..n)
    Column (1)
    Where (0..1)
      Expressions (1..n)
        Expression (1..n)
          Column (1)
          Value (1)
```

**Filter Element**

Each filter is a source of possible items. These items are values of a variable from the `DataSource` element. All of the values for a variable are returned unless a `Where` element is specified. Each `Filter` element must have a single `Column` element and can have a single `Where` element.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Specifies the name assigned to the filter.</td>
</tr>
</tbody>
</table>

These option types can use the `filter` attribute to populate their items from the `Filter`.
- combobox
- dualselector
- select

**Column Element**

The `Column` element specifies which column in the data source contains the values that you want to use. You can use the attributes for this element to define how the values are returned. For example, only the distinct values are returned or how the values are presented to the end user.

If you want to define the column to use in the task, use the `column` attribute, which creates a static column. The end user of the task has to use this column in the task. To enable users to select the column to use, use the `role` attribute. At run time, the user sees a Role control and can select the column to use.

**Note:** Either the `column` attribute or the `role` attribute must be specified. You cannot specify both attributes.

The label for the `Column` element can be a formatted value or a value from another column in the data source. To specify a static value, use the `labelColumn` attribute. The user can use only this column in the task. To enable the user to select the
column from a Role control, use the `labelRole` attribute. By default, the formatted value of the selected column is used. To overwrite this formatting, use the `format` attribute.

**Note:** If neither the `labelColumn` or `labelRole` are specified, the value of the `column` attribute or the `role` attribute is used as the label.

This table describes the attributes for the `Column` element.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>column</code></td>
<td>Specifies the column in the data source that provides the values for the option control.</td>
</tr>
<tr>
<td><code>role</code></td>
<td>Specifies the role that identifies the column in the data source that provides the values for the option control. If the role can accept multiple columns, only the first column is used.</td>
</tr>
<tr>
<td><code>labelColumn</code></td>
<td>Specifies the column in the data source that provides the labels for the values in the option control. The return value is the formatted value of the column.</td>
</tr>
<tr>
<td><code>labelRole</code></td>
<td>Specifies the role that identifies the column to use for the formatted value. The formatted value is used when the <code>display</code> attribute is set to either <code>formatted</code> or <code>both</code>. If the role can accept multiple columns, only the first column is used. If the <code>labelRole</code> is not specified, the formatted value is derived from the column specified by the <code>role</code> attribute.</td>
</tr>
<tr>
<td><code>format</code></td>
<td>Specifies the SAS format to apply to the label. This format overrides the format associated with the column specified by the <code>labelColumn</code>, <code>labelRole</code>, <code>column</code>, or <code>role</code> attribute.</td>
</tr>
<tr>
<td><code>distinct</code></td>
<td>Specifies whether to return only unique values. By default, this value is <code>true</code>, and only unique values are returned. When this attribute is set to <code>false</code>, all values are returned.</td>
</tr>
<tr>
<td><code>display</code></td>
<td>Determines how the values are displayed. Here are the valid values:</td>
</tr>
<tr>
<td></td>
<td>- <code>formatted</code> uses the formatted value of the variable. If specified, the formatted value comes from the <code>labelRole</code> attribute. Otherwise, this value comes from the <code>role</code> attribute.</td>
</tr>
<tr>
<td></td>
<td>- <code>unformatted</code> uses the unformatted value and ignores any column format defined in the data source.</td>
</tr>
<tr>
<td></td>
<td>- <code>both</code> displays both the formatted and unformatted values. If specified, the formatted value comes from the <code>labelRole</code> attribute. Otherwise, this value comes from the <code>role</code> attribute.</td>
</tr>
<tr>
<td><code>max</code></td>
<td>Specifies the maximum number of values to return. The default value is 100.</td>
</tr>
<tr>
<td><code>sortBy</code></td>
<td>Specifies whether to sort by the unformatted or formatted values. By default, this value is <code>value</code>, which means the values are sorted by the unformatted values. If you specify <code>label</code>, the values are sorted by the formatted value.</td>
</tr>
</tbody>
</table>
### Attribute Description

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
</table>
| sortDirection | Specifies whether to sort the data. Here are the valid values:  
  - none does not sort the data.  
  - ascending sorts the values in ascending order.  
  - descending sorts the values in descending order. |

### Where Element

The optional `Where` element determines the values returned by the filter. Use the `Where` element to specify the values to return based on various options and how these variables compare to the specified variables in the data source.

When specified, the `Where` element must have a single `Expressions` element. There are no attributes associated with the `Where` element.

### Expressions Element

The `Expressions` element must have one or more `Expression` children. The `Expressions` element is a grouping mechanism for the expression definitions. There are no attributes associated with the `Expressions` element.

When an `Expressions` element contains multiple `Expression` children, all the expressions must be satisfied for the row in the data source to provide values to the `Filter` element's output. The expressions are joined by the AND operator.

### Expression Element

Each `Expression` element represents a part of the overall condition. For example, you could create an expression where the user sees only the values less than the number specified in the `numbertext` control.

Each `Expression` element must have a single `Column` element and a single `Value` element. The `Column` element specifies which column to use in the expression. Any column in the data source can be used.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>not</td>
<td>Specifies which values to retrieve. By default, this value is <code>false</code>. When this attribute is set to <code>true</code>, the result of the condition is negated.</td>
</tr>
</tbody>
</table>
| operator  | Specifies the operator to use in the expression. Here are the valid values: `eq`, `ne`, `gt`, `ge`, `lt`, `le`, `contains`, `in`, and `like`.  
  **Note:** Some operators work only with specific column or option types. For example, the contains and like operators work only with character variables and cannot be used with range values. |

**Note:** If an option returns an ALL value to an expression, the expression is evaluated in this way:
true if the operators are eq or in
false if the operators are ne, gt, or lt
an error is generated for all other operators

Column Element

The Column element specifies the variable to use in the comparison.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>column</td>
<td>Specifies the column in the data source that provides the values for the option.</td>
</tr>
<tr>
<td>role</td>
<td>Specifies the role that determines the column in the data source that provides the values for the option. Tip: Because the filter uses only the first variable, you might want to set maxVars=&quot;1&quot; in the Role element.</td>
</tr>
</tbody>
</table>

Note: Specify either the column attribute or the role attribute, but not both.

Value Element

The Value element specifies the value to compare to the Column element for the expression.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>Enables the user to specify a string value.</td>
</tr>
<tr>
<td>option</td>
<td>Specifies the option that will provide the values for the comparison.</td>
</tr>
</tbody>
</table>

Options that provide lists or ranges of strings, numbers, or dates can be used if these values are supported by the operator and column type.

Example of a Filter

In this example, the user selects the value of Make from the Sashelp.Cars data set.

```xml
<Metadata>
  <DataSources>
    <DataSource name="CARDATA" defaultValue="SASHELP.CARS" active="true">
      <Roles>
        <Role type="A" maxVars="1" order="true" minVars="1" name="VAR">
          Choose a column to retrieve values for:
        </Role>
      </Roles>
      <Filters>
        <Filter name="filterMake">
          <Column column="make"/>
```
<Filter name="filterVar">
  <Column role="VAR" sortDirection="descending"/>
  <Where>
    <Expressions>
      <Expression operator="ge">
        <Column column="cylinders"/>
        <Value value="6"/>
      </Expression>
      <Expression operator="eq">
        <Column column="make"/>
        <Value option="comboMake"/>
      </Expression>
      <Expression operator="in">
        <Column column="msrp"/>
        <Value option="priceRange"/>
      </Expression>
    </Expressions>
  </Where>
</Filter>
</Filters>
</DataSource>
</DataSources>

<!--- Define the task options.    -->
<Options>
  <Option name="comboMake" defaultValue="default" inputType="combobox" filter="filterMake">Select a car make:</Option>
  
  <Option name="priceRange" inputType="numericrange">Specify a price range:</Option>
  
  <Option name="OPTIONSTAB" inputType="string">OPTIONS</Option>
  
  <Option name="comboVar" defaultValue="default" inputType="combobox" filter="filterVar">Select a value from Role:</Option>
  
  <Option name="message" inputType="string">This combobox shows values from the column chosen above. The values shown have been run through an expression where cylinders are greater than 6 and msrp is between the price range specified above.</Option>
</Options>
</Metadata>

<UI>
  
  <Container option="OPTIONSTAB">
    <OptionChoice option="comboMake"/>
    <OptionItem option="priceRange"/>
    <RoleItem role="VAR"/>
    <OptionItem option="message"/>
    <OptionChoice option="comboVar"/>
  </Container>
</UI>
1. The `DataSource` element specifies that Sashelp.Cars is the default data source for this task.

2. The `Filter` element (filterMake) specifies that the values for this filter come from the Make column in the Sashelp.Cars data source. Because the `Column` element is used, the column is static.

3. The second `Filter` element specifies the `role` attribute. In this case, the column is dynamic. The filter values come from the column that the user selects at runtime.

4. The first `Expression` element specifies that the value of the cylinder column must be greater than 6. Only column values that meet this criterion are considered for the next expression.

5. The second `Expression` element specifies that the value of the Make column must equal the value selected in the first `Filter` element (filterMake).

6. The third `Expression` element specifies that the value of MSRP must be in the range specified in the numericrange control called priceRange.

---

**Working with the Options Element**

**About the Option Element**

The `Options` element identifies the options that appear in the task’s user interface. The `Option` tag, which is the only child of the `Options` element, describes the assigned option.

Here are the attributes of the `Option` element.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>active</td>
<td>Specifies whether the option’s Velocity value is accessible. The default value is <code>false</code>. When this attribute is set to <code>true</code>, the option value is available even when hidden, disabled, or when the value does not appear in the user interface.</td>
</tr>
<tr>
<td>defaultValue</td>
<td>Specifies the initial value for the option.</td>
</tr>
<tr>
<td>helpMessageRef</td>
<td>Specifies whether to display help content. This content could be a string or markdown. When this attribute is defined, a Help icon appears to the right of the control’s label.</td>
</tr>
<tr>
<td>hide</td>
<td>Specifies whether to display the control in the user interface. By default, this attribute is <code>false</code>. If this attribute is set to <code>true</code>, the control is not displayed.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| indent       | Specifies the indentation for this option in the task interface. Here are the valid values:  
  1 – minimal indentation (about 17px)  
  2 – average indentation (about 34px)  
  3 – maximum indentation (about 51px)  |
| inputType    | Specifies the input control for this option. Here are the valid values:  
  checkbox  
  color  
  combobox  
  datepicker  
  distinct  
  dualselector  
  inputtext  
  mixedeffects  
  multientry  
  numstepper  
  numbertext  
  outputdata  
  passwordtext  
  radio  
  sasserverpath  
  select  
  slider  
  string  
  textbox  
  validationtext  |
| name         | Specifies the name assigned to this option.                                |
| readOnly     | Specifies whether the current value can be edited. The default value is false. If this attribute is true, the current value is displayed but cannot be edited. |
| returnValue  | Applies to strings that are used by input types (such as combobox and select) where the user has a selection of choices. If the returnValue attribute is specified in other contexts, this attribute is ignored.  
  For more information, see “Specifying a Return Value Using the returnValue Attribute” on page 67. |
### Attribute Description

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>width</td>
<td>Specifies the width of the control. The width can be specified in %, em, or px. The default behavior is to autosize the control based on available width and content.</td>
</tr>
</tbody>
</table>

Here is an example of an Options element:

```xml
<Options>
  <Option name="PRINTOPTIONS" inputType="string">List Data Options</Option>
  <Option name="BASICOPTIONS" inputType="string">Basic Options</Option>
  <Option name="OBS" defaultValue="1" inputType="checkbox">Print row number</Option>
  <Option name="OBSHEADING" defaultValue="Row number" inputType="inputtext">Column heading</Option>
  <Option name="LABEL" defaultValue="1" inputType="checkbox">Use variable labels as column headings</Option>
  <Option name="PRINTNUMROWS" defaultValue="0" inputType="checkbox">Print number of rows</Option>
</Options>
```

### Supported Input Types

#### checkbox

This input type does not have additional attributes. The default value for a checkbox is either 0 (unchecked) and 1 (checked).

Here is the example code in the Sample Task:

```xml
<Option name="GROUPCHECK" inputType="string">CHECK BOX</Option>
<Option name="LabelCheck" inputType="string">
  An example of a check box. Check boxes are either on or off.</Option>
<Option name="chkEXAMPLE" defaultValue="0" inputType="checkbox">
  Check box</Option>
```

Here is an example of a check box control in the user interface:

![Check box example](image)

#### color

This input type has one attribute:
### Attribute

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
</table>
| required     | Specifies whether a value is required. Valid values are true and false. The default value is false.  
Note: If the required attribute is set to true and no default value is specified, the user must select a color to run the task. |

Here is an example from the sample task definition:

```xml
<Option name="GROUPCOLOR" inputType="string">COLOR SELECTOR</Option>  
<Option name="labelCOLOR" inputType="string">An example of a color selector.</Option>  
<Option name="colorEXAMPLE" defaultValue="red" inputType="color">  
  Choose a color</Option>
```

Here is an example of a color control in the user interface:

![COLOR SELECTOR](image)

#### combobox

This input type has these attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>allowAllValues</td>
<td>Specifies whether to add an item to the list so that the user can select all possible values. By default, this attribute is false, and no item is added to the list. You might use this attribute when the list is generated by a filter.</td>
</tr>
<tr>
<td>allowMissingValues</td>
<td>Specifies whether to add an item to the list so that the user can select missing values. By default, this attribute is false, and no item is added to the list. You might use this attribute when the list is generated by a filter.</td>
</tr>
<tr>
<td>editable</td>
<td>Specifies whether the user can enter a value in the combobox control. By default, users cannot enter a new value in the combobox control.</td>
</tr>
</tbody>
</table>
| filter         | Specifies the filter to use for pulling data for this option.  
Note: If this attribute is specified, the sourceLink attribute and any children in the OptionChoice element are ignored. |
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>required</td>
<td>Specifies whether a value is required. Valid values are true and false. The default value is false. <strong>Note:</strong> If the <code>required</code> attribute is set to true and no default value is specified, the combobox control displays the text specified in the <code>selectMessage</code> attribute.</td>
</tr>
<tr>
<td>selectMessage</td>
<td>Specifies the message to display when a value is required for the combobox control and no default value has been set. The default message is Select a value.</td>
</tr>
<tr>
<td>sourceLink</td>
<td>Specifies that the data for the combobox should be pulled from another source option. For more information, see &quot;About Data Linking&quot; on page 68.</td>
</tr>
<tr>
<td>width</td>
<td>Specifies the width of the control. This value can be in percent (%), em, or px. By default, SAS Studio sizes the control based on the available width and content.</td>
</tr>
</tbody>
</table>

The code in the Sample Task creates a combination box called **Combobox**. This list contains three options: **Value 1**, **Value 2**, and **Value 3**.

```xml
<Option name="GROUPCOMBO" inputType="string">COMBOBOX</Option>
<Option name="labelCOMBO" inputType="string">An example of a combobox.</Option>
<Option name="comboEXAMPLE" defaultValue="value2" inputType="combobox" width="100%">Combobox:</Option>
<Option name="value1" inputType="string">Value 1</Option>
<Option name="value2" inputType="string">Value 2</Option>
<Option name="value3" inputType="string">Value 3</Option>
```

Here is an example of a combobox control in the user interface:

![Combobox example](image)

datepicker

This option returns all date values in the ISO8601 format (yyyy-MM-dd). This input type has these attributes:
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>displayFormat</td>
<td>Specifies the visual formatting of the date. The valid values are short (default), medium, and long.</td>
</tr>
<tr>
<td>required</td>
<td>Specifies whether a date is required. By default, no date is required.</td>
</tr>
<tr>
<td>minDate</td>
<td>Specifies the minimum threshold for date values. This attribute is not set by default.</td>
</tr>
<tr>
<td>maxDate</td>
<td>Specifies the maximum threshold for date values. This attribute is not set by default.</td>
</tr>
<tr>
<td>width</td>
<td>Specifies the width of the control. This value can be in percent (%), em, or px. By default, SAS Studio sizes the control based on the available width and content.</td>
</tr>
</tbody>
</table>

If you specify the `defaultValue`, `minDate`, and `maxDate` attribute for this input type, the value must be in ISO8601 format (yyyy-mm-dd).

The code in the Sample Task creates datepicker control with the label **Choose a date:**.

```xml
<Option name="GROUPDATE" inputType="string">DATE PICKER</Option>
<Option name="labelDATE" inputType="string">An example of a date picker.</Option>
<Option name="dateEXAMPLE" inputType="datepicker" format="monyy7."">Choose a date:</Option>
```

Here is an example of a datepicker control in the user interface:
The `daterange` control enables the user to choose a date and time period.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>defaultFromValue</td>
<td>Specifies the default value for the <code>From</code> input field. The default value is null.</td>
</tr>
<tr>
<td>defaultToValue</td>
<td>Specifies the default value for the <code>To</code> input field. The default value is null.</td>
</tr>
<tr>
<td>displayFormat</td>
<td>Specifies the visual formatting of the date. The valid values are short (default), medium, and long.</td>
</tr>
<tr>
<td>dateType</td>
<td>Specifies the type of control to display in the user interface. Here are the valid values: date (the default), week, month, quarter, year, time, and datetime.</td>
</tr>
<tr>
<td>fromLabel</td>
<td>Overrides the label for the <code>From</code> input field.</td>
</tr>
<tr>
<td>toLabel</td>
<td>Overrides the label for the <code>To</code> input field.</td>
</tr>
<tr>
<td>minValue</td>
<td>Specifies the minimum threshold for values. Any values less than this threshold are invalid. This attribute is not set by default.</td>
</tr>
<tr>
<td>maxValue</td>
<td>Specifies the maximum threshold for values. Any values greater than this threshold are invalid. This attribute is not set by default.</td>
</tr>
<tr>
<td>width</td>
<td>Specifies the width of the control. This value can be in percent (%), em, or px. By default, SAS Studio sizes the control based on the available width and content.</td>
</tr>
<tr>
<td>required</td>
<td>Specifies whether a date is required. By default, no date is required.</td>
</tr>
</tbody>
</table>

**Note:** When using the `daterange` element, consider these items:

- The standard attribute, `defaultValue`, is not a valid attribute for the `daterange` element. Instead, use the `defaultFromValue` and `defaultToValue` attributes.
- All date and time attributes should be in an ISO format.
- The `displayFormat` attribute is not valid for week and year.
- The `minValue` and `maxValue` attributes are not honored when the date range is time. If the date range is datetime, the date portion of the `minValue` and `maxValue` attributes is honored, but not the time portion.

```xml
<Option name="labelDATERANGE" inputType="daterange">
  Specify the date range:
</Option>

<Option name="labelDATETIMERANGE" inputType="daterange" displayFormat="short" dateType="DateTime" maxValue="2019-12-31T19:15:22" defaultFromValue="2018-12-31T19:15:22">
```
Specify the time range:

```xml
<Option name="labelYEARRANGE" inputType="daterange" dateType="year"
        maxValue="2050" defaultFromValue="2018" defaultToValue="2020">
    Specify the year range:
</Option>
```

An example of a date range picker. Note that the date type attribute can be used with the following date types: Date, Time, DateTime, Week, Month, Quarter, and Year.

Choose a date range:

```
From:              To:
    Select a date   Select a date
```

```
datetimepicker
```

The `datetimepicker` element enables you to choose a date and time.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>displayFormat</td>
<td>Specifies the visual formatting of the date. The valid values are short (default), medium, and long.</td>
</tr>
<tr>
<td>required</td>
<td>Specifies whether a date is required. By default, no date is required.</td>
</tr>
<tr>
<td>minDate</td>
<td>Specifies the start date. All previous dates are disabled. The default value is null.</td>
</tr>
<tr>
<td>maxDate</td>
<td>Specifies the end date. All future dates are disabled. The default value is null.</td>
</tr>
<tr>
<td>use24HourTime</td>
<td>Specifies whether to display the time using the 24-hour format instead of AM and PM. By default, this attribute is false, and the AM and PM format is used.</td>
</tr>
<tr>
<td>showSeconds</td>
<td>Specifies whether to show seconds in the time. By default, this attribute is false, and the seconds are not displayed.</td>
</tr>
<tr>
<td>width</td>
<td>Specifies the width of the control. This value can be in percent (%), em, or px. By default, SAS Studio sizes the control based on the available width and content.</td>
</tr>
</tbody>
</table>

If you specify the `defaultValue`, `minDate`, and `maxDate` attribute for this input type, the value must be in ISO8601 format: `yyyy-MM-ddTHH:mm:ss` and `yyyy-MM-dd`.

```xml
<Option name="labelDATETIME" inputType="datetimepicker" required="true"
        defaultValue="2018-09-05T19:15:22" displayFormat="long"
        minDate="2018-09-01" maxDate="2018-09-30" use24HourTime="true"
        helpMessageRef="helpMessage">
    Select a Date and Time:
</Option>
```
An example of a datetime picker.
Choose a date and time:

Select a date and time

February 2019

distinct
This input type has these attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>required</td>
<td>Specifies whether a value is required. The default value is false. Note: If the required attribute is set to true and no default value is specified, the combobox control displays the text specified in the selectMessage attribute.</td>
</tr>
<tr>
<td>selectMessage</td>
<td>Specifies the message to display when a value is required for the combobox control and no default value has been set. The default message is Select a value.</td>
</tr>
<tr>
<td>source</td>
<td>Specifies the role to use to get the distinct values. The maxVars control for the role must be set to 1. In other words, users can assign only one variable to this role.</td>
</tr>
</tbody>
</table>
### Attribute

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
</table>
| max       | Specifies the maximum number of distinct values to obtain and display in the UI. By default, the maximum value is 100. Larger maximum values might cause a long delay in populating the UI control.  
**Note:** Missing values are ignored, so missing values do not appear in the list of distinct values. |
| width     | Specifies the width of the control. This value can be in percent (%), em, or px. By default, SAS Studio sizes the control based on the available width and content. |

When using the `distinct` control, remember this information:

- Missing values are excluded from the list of returned values.
- The `distinct` control is affected by any filter that is applied to the data source.  
  For more information, see the `where` attribute in “About the `DataSources` Element” on page 17.

In this example, you want the user of this task to see the first 15 distinct values for the response variable.

In the code, you first specify the `DataSources` element because an input data set is required to run this task. Then, in the `Roles` element, you specify that only one response variable is required to run this task. The `name` attribute for this role is `VAR`.

Now, you want to create an option that lists the first 15 distinct values in the `VAR` variable. The code for the `distinct` input type includes these attributes:

- The `source` attribute specifies that the values that appear in the `Age of interest` option come from the `VAR` role (in this example, the `Age` variable).
- The `max` attribute specifies that a maximum of 15 values should be available for the `Age of interest` option.

```xml
<DataSources>
  <DataSource name="DATASOURCE">
    <Roles>
      <Role type="A" maxVars="1" order="true" minVars="1" name="VAR">Response variable</Role>
    </Roles>
  </DataSource>
</DataSources>

<Options>
  <Option name="values" inputType="distinct" source="VAR" max="15">Age of interest:</Option>
</Options>
```

Here is an example of the `distinct` control in the sample task:
**dualselector**

This input type has these attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>allowAllValues</td>
<td>Specifies whether to add an item to the list so that the user can select all possible values. By default, this attribute is <code>false</code>, and no item is added to the list. You might use this attribute when the list is generated by a filter.</td>
</tr>
<tr>
<td>allowMissingValues</td>
<td>Specifies whether to add an item to the list so that the user can select missing values. By default, this attribute is <code>false</code>, and no item is added to the list. You might use this attribute when the list is generated by a filter.</td>
</tr>
<tr>
<td>dataType</td>
<td>Specifies the type of data allowed in the dualselector control.</td>
</tr>
<tr>
<td></td>
<td>If the <code>editable</code> attribute is set to <code>true</code>, the edit field uses the value for <code>dataType</code>.</td>
</tr>
<tr>
<td></td>
<td>Here are the valid values:</td>
</tr>
<tr>
<td></td>
<td>• <code>auto</code> (default). If the <code>dataType</code> attribute is used with a filter, <code>auto</code> chooses the data type based on the column type. If a filter is not applied, <code>auto</code> is the same as text.</td>
</tr>
<tr>
<td></td>
<td>• <code>date</code></td>
</tr>
<tr>
<td></td>
<td>• <code>number</code></td>
</tr>
<tr>
<td></td>
<td>• <code>text</code></td>
</tr>
<tr>
<td></td>
<td>If you are using static values, the <code>returnValue</code> should be specified when the <code>dataType</code> is either <code>date</code> or <code>number</code>. Date value should be in the ISO format (<code>yyyy-mm-dd</code>). If the return value does not equal the data type or is forced into that data type, the value is not added to the list in the dualselector control.</td>
</tr>
<tr>
<td>editable</td>
<td>Specifies whether the user can enter a value. This attribute is used with the <code>dataType</code> attribute.</td>
</tr>
<tr>
<td>filter</td>
<td>Specifies the filter to use for pulling data for this option.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> If this attribute is specified, the <code>sourceLink</code> attribute and any children in the <code>OptionChoice</code> element are ignored.</td>
</tr>
<tr>
<td>height</td>
<td>Specifies the height of the control. This value can be in <code>em</code> or <code>px</code>. If a height is not specified, SAS Studio sizes the control based on a reasonable default.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>maxItems</td>
<td>Specifies the maximum number of values that can be selected from the list of available values.</td>
</tr>
<tr>
<td>minItems</td>
<td>Specifies the minimum number of values that must be selected from the list of available values.</td>
</tr>
<tr>
<td>reorderable</td>
<td>Specifies whether the list of values can be reordered. By default, this attribute is set to true, and the values can be reordered.</td>
</tr>
<tr>
<td>required</td>
<td>Specifies whether any input text is required. Valid values are true and false. The default value is false.</td>
</tr>
<tr>
<td>width</td>
<td>Specifies the width of the control. This value can be in percent (%), em, or px. By default, SAS Studio sizes the control based on the available width and content.</td>
</tr>
</tbody>
</table>

You can specify default values for the dualselector control by using the `defaultValue` attribute. Any default values that you specify are selected at run time. If you need to specify multiple default values, use a comma-separated list of values for the `defaultValue` attribute.

This example shows how the dualselector control works.

```xml
<Options>
  <Option name="ANOTHERLIST" inputType="dualselector"
    defaultValue="anothertest2, anothertest3">Test choices:</Option>
  <Option inputType="string" name="anothertest1">Another 1</Option>
  <Option inputType="string" name="anothertest2">Another 2</Option>
  <Option inputType="string" name="anothertest3">Another 3</Option>
  <Option inputType="string" name="anothertest4">Another 4</Option>
  <Option inputType="string" name="anothertest5">Another 5</Option>
  <Option inputType="string" name="anothertest6">Another 6</Option>
</Options>

<UI>
  <OptionChoice option="ANOTHERLIST">
    <OptionItem option="anothertest1"/>
    <OptionItem option="anothertest2"/>
    <OptionItem option="anothertest3"/>
    <OptionItem option="anothertest4"/>
    <OptionItem option="anothertest5"/>
    <OptionItem option="anothertest6"/>
  </OptionChoice>
</UI>
```

When you run this code, the **Test choices** option appears in the user interface. In this example, the `defaultValue` attribute specifies to use the values for `anothertest2` and `anothertest3` as the default values for this option. As a result, **Another 2** and **Another 3** are automatically selected for the **Test choices** option.
To change the selected values, click Edit. A new dialog box appears. From this dialog box, the user can see a list of all the available variables and then select which variables to use for the Test choices option.
When the user clicks **OK**, any variables in the **Selected items** pane now appear in the list of values for the **Test choices** option. To specify the order of the values in the **Test choices** option, use the up and down arrows for the **Selected** pane.
**inputtext**

This input type has these attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hintMessage</td>
<td>Specifies the text to display when the text box is empty.</td>
</tr>
<tr>
<td>missingMessage</td>
<td>Specifies the tooltip text that appears when the text box is empty but input text is required. No message is displayed by default.</td>
</tr>
<tr>
<td>promptMessage</td>
<td>Specifies the tooltip text that appears when the text box is empty and the user has selected the text box.</td>
</tr>
<tr>
<td>required</td>
<td>Specifies whether any input text is required. Valid values are true and false. The default is false.</td>
</tr>
<tr>
<td>width</td>
<td>Specifies the width of the control. This value can be in percent (%), em, or px. By default, SAS Studio sizes the control based on the available width and content.</td>
</tr>
</tbody>
</table>

The code in the Sample Task creates a text box called **Input text**. The default value is “Text goes here.” If the user removes this text, the message “Enter some text” appears because a value is required.

```xml
<Option name="textEXAMPLE" defaultValue="Text goes here" inputType="inputtext" required="true" promptMessage="Enter some text." missingMessage="Missing text.">Input text:</Option>
```

Here is an example of an inputtext control in the user interface:

![Text field example](image)

**markdown**

This input type is used to display formatted text to the user. Standard markdown is accepted. You should use markdown’s syntax of punctuation of characters rather than an HTML tag because the HTML tags are removed for security reasons.

Use the markdown attribute to display formatted text to the user. You can also refer to the markdown attribute from an option’s helpMessageRef attribute.

```xml
<Option name="markdownTextEXAMPLE" inputType="markdown">Heading

======

## Sub-heading

Horizontal rule:
```
Here is a subset of the contents of the Markdown tab in the Sample Task.
A model is an equation that consists of a dependent or response variable and a list of effects. The user creates the list of effects from variables and combinations of variables.

Here are examples of effects:

main effect
For variables Gender and Height, the main effects are Gender and Height.

interaction effect
For variables Gender and Height, the interaction is Gender * Height. You can have two-way, three-way, ...n-way interactions.

The order of the variables in the interaction is not important. For example, Gender * Height is the same as Height * Gender.
nested effect
For variables Gender and Height, an example of a nested effect is Gender(Height).

polynomial effect
You can create polynomial effects with continuous variables. For the continuous variable $X$, the quadratic polynomial effect is $X^2$. You can have second-order, third-order, \ldots nth-order polynomial effects.

The mixedeffects control enables users to create various model effects. You can define fixed effects, random effects, repeated effects, means effects, and zero-inflated effects. For the control to work properly, you must specify at least one of the role attributes, roleContinuous or roleClassification. If no roles are specified, the control is displayed but the user has no variables to work with.

Here are the attributes for the mixedeffects input type:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>effects</td>
<td>Specifies the list of effects that you want available from the task interface:</td>
</tr>
<tr>
<td>fixed</td>
<td>only fixed effects. This is the default value.</td>
</tr>
<tr>
<td>fixedrandom</td>
<td>fixed effects and random effects.</td>
</tr>
<tr>
<td>fixedrandomrepeated</td>
<td>fixed effects, random effects, and repeated effects.</td>
</tr>
<tr>
<td>fixedrepeated</td>
<td>fixed and repeated effects.</td>
</tr>
<tr>
<td>meanszero</td>
<td>means and zero-inflated effects.</td>
</tr>
<tr>
<td>roleContinuous</td>
<td>Specifies the role that contains the continuous variables.</td>
</tr>
<tr>
<td>roleClassification</td>
<td>Specifies the role that contains the classification variables.</td>
</tr>
<tr>
<td>excludeTools</td>
<td>Specifies the effect and model buttons to exclude from the user interface.</td>
</tr>
<tr>
<td>fixedInterceptVisible</td>
<td>Specifies whether the intercept option is available for fixed effects or mean effects. Valid values are true and false. The default value is true.</td>
</tr>
<tr>
<td>fixedInterceptDefaultValue</td>
<td>Specifies the default value for the intercept option if fixedInterceptVisible = true. Valid values are 0 and 1. The default value is 1.</td>
</tr>
<tr>
<td>randomInterceptVisible</td>
<td>Specifies whether the intercept option is available for random effects. Valid values are true and false. The default value is true.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>--------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>randomInterceptDefaultValue</td>
<td>Specifies the default value for the intercept option if randomInterceptVisible = true. Valid values are 0 and 1. The default value is 1.</td>
</tr>
<tr>
<td>width</td>
<td>Specifies the width of the control. The width value can be specified in percent, em, or px. By default, the control is automatically sized based on the available width and content.</td>
</tr>
</tbody>
</table>

Here is an example of the mixedeffects control from the Advanced Task:

```xml
<Options>
  <Option name="MECTAB" inputType="string">MIXED EFFECTS CONTROL</Option>
  <Option name="MECTEXT" inputType="string">This tab shows an example of the Mixed Effects control. The variables come from both the Variables and Numeric Variables roles.</Option>
  <Option name="mixedEffects" inputType="mixedeffects"
    roleContinuous="dataVariablesNumeric" roleClassification="dataVariables"
    excludeTools="POLYEFFECT,TWOFACT,THREEFACT,NFACTPOLY"></Option>
  ...
</Options>

<UI>
  <Container option="MECTAB">
    <OptionItem option="MECTEXT"/>
    <OptionItem option="mixedEffects"/>
  </Container>
</UI>
```

If you run the Advanced Task, here is the resulting **Mixed Effects Control** tab:

This tab shows an example of the Mixed Effects control. The variables come from both the Variables and Numeric Variables roles.

If you click **Edit**, the Model Effects Builder appears.
The component opens, but there are no variables available. You must assign a variable to the continuous variable or classification variable role. You can assign variables to both roles.

In the Advanced Task, close the Model Effects Builder and click the Data tab. Select an input data source (such as Sashelp.Pricedata) and assign variables to the Variables and Numeric Variables roles.
Return to the Model Effects Control tab and click Edit. Now, the price and sale variables are available from the Variables pane.
The `monthpicker` attribute creates a control that enables the user to choose a month and year.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>displayFormat</td>
<td>Specifies the visual formatting of the date. The valid values are short (default), medium, and long. If you are using static values, the <code>returnValue</code> should be specified when the <code>dataType</code> is either date or number. Date value should be in the ISO format (yyyy-mm-dd). If the return value does not equal the data type or be forced into that data type, the value is not added to the list in the dualselector control.</td>
</tr>
<tr>
<td>required</td>
<td>Specifies whether a date is required. By default, no date is required.</td>
</tr>
<tr>
<td>minValue</td>
<td>Specifies the minimum threshold for the month values. This attribute is not set by default.</td>
</tr>
<tr>
<td>maxValue</td>
<td>Specifies the maximum threshold for month values. This attribute is not set by default.</td>
</tr>
</tbody>
</table>
### Attribute table

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>width</td>
<td>Specifies the width of the control. This value can be in percent (%), em, or px. By default, SAS Studio sizes the control based on the available width and content.</td>
</tr>
</tbody>
</table>

If you specify the `defaultValue`, `minDate`, and `maxDate` attribute for this input type, the value must be in ISO8601 format: `yyyy-MM`.

```xml
<Order inputType="monthpicker" name="monthEXAMPLE" defaultValue="2020-07"
    minValue="2019-15" maxValue="2020-25" displayFormat="long">
    Select a month:
</Option>
```

An example of a month picker. Choose a month:

- **Month:** October
- **Year:** 2019

You can assign default values to the `monthpicker` control by using the `OptionsChoice` element.

### multientry

This input type has these attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dataType</td>
<td>Specifies the data type that is allowed for user-specified values. The edit field matches the data type. The valid values are <code>date</code>, <code>number</code>, and <code>text</code> (default).</td>
</tr>
<tr>
<td>required</td>
<td>Specifies whether a value is required. Valid values are <code>true</code> and <code>false</code>. The default value is <code>false</code>.</td>
</tr>
<tr>
<td>width</td>
<td>Specifies the width of the control. This value can be in percent (%), em, or px. By default, SAS Studio sizes the control based on the available width and content.</td>
</tr>
<tr>
<td>reorderable</td>
<td>Specifies whether the user can reorder the values in the list. Valid values are <code>true</code> and <code>false</code>. The default value is <code>false</code>.</td>
</tr>
</tbody>
</table>

You can assign default values to the `multientry` control by using the `OptionsChoice` element.
The code in the Sample Task creates the **Multiple entry** option.

```xml
<Options>
  <Option name="labelMULTIENTRY" inputType="string">An example of a multiple entry. This control allows the user to add their own values to create a list.</Option>
  <Option name="multientryEXAMPLE" inputType="multientry">Multiple entry:</Option>
</Options>

<UI>
...
</UI>
```

In this example, the **Multiple entry** option has three values: **Value 1**, **Value 2**, and **Value 3**. To add additional values to the list, enter the name of the new value in the text box and click +.

To enable users to reorder the values in this list, set the reorderable attribute to true, as shown in this example.

```xml
<Options>
  <Option name="labelMULTIENTRY" inputType="string">An example of a multiple entry. This control allows the user to add their own values to create a list.</Option>
  <Option name="multientryEXAMPLE" inputType="multientry" reorderable="true">Multiple entry:</Option>
</Options>

<UI>
...
</UI>
```

Now, the multientry control includes up and down arrows.
## numbertext

This input type has these attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>decimalPlaces</td>
<td>Specifies the number of decimal places to display. Valid values include a single value or a range. To create a field that allows 0 to 3 decimal places, specify decimalPlaces=&quot;0,3&quot;. The maximum number of decimal places is 15.</td>
</tr>
<tr>
<td>formatValue</td>
<td>Specifies whether the number should be formatted to include locale-specific delimiters in the user interface. The default value is true. Note: Setting this attribute does not change the numeric value in the Velocity variable.</td>
</tr>
<tr>
<td>hintMessage</td>
<td>Specifies the placeholder text when the control is empty.</td>
</tr>
<tr>
<td>invalidMessage</td>
<td>Specifies the tooltip text that appears when the content is invalid.</td>
</tr>
<tr>
<td>maxValue</td>
<td>Specifies the maximum value that is allowed. If the user tries to exceed this value, a message appears. The default value is 9000000000000.</td>
</tr>
<tr>
<td>maxInclusive</td>
<td>Specifies whether to include the maximum value in the range of values. By default, the maxInclusive attribute is set to true.</td>
</tr>
<tr>
<td>minValue</td>
<td>Specifies the minimum value that is allowed. If the user specifies a value that is below the minimum value, a message appears.</td>
</tr>
</tbody>
</table>
### Attribute Description

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>minInclusive</td>
<td>Specifies whether to include the minimum value in the range of values. By default, the <code>minInclusive</code> attribute is set to <code>true</code>.</td>
</tr>
<tr>
<td>missingMessage</td>
<td>Specifies the tooltip text that appears when the text box is empty, but a value is required.</td>
</tr>
<tr>
<td>promptMessage</td>
<td>Specifies the tooltip text that appears when the text box is empty, and the field has focus.</td>
</tr>
<tr>
<td>rangeMessage</td>
<td>Specifies the tooltip text that appears when the value in the text box is outside the specified range.</td>
</tr>
<tr>
<td>required</td>
<td>Specifies whether a value is required. Valid values are <code>true</code> and <code>false</code>. The default value is <code>false</code>.</td>
</tr>
<tr>
<td>width</td>
<td>Specifies the width of the control. This value can be in percent (%), em, or px. By default, SAS Studio sizes the control based on the available width and content.</td>
</tr>
</tbody>
</table>

This example code creates a field called **Number text**.

```xml
<Option name="labelNUMBERTEXT" inputType="string">An example of a number text. The minimum value is set to 0 and the maximum value is set to 100.</Option>
<Option name="numberTextEXAMPLE" defaultValue="1" inputType="numbertext"
    minValue="0"
    maxValue="100"
    promptMessage="Enter a number between 0 and 100."
    missingMessage="Enter a number between 0 and 100."
    rangeMessage="This number is out of range. Enter a number between 0 and 100."
    invalidMessage="Invalid value. Enter a number between 0 and 100.">
    Number text:
</Option>
```

Here is an example of the numbertext control in the user interface. In this example, 110 is higher than the maximum value allowed for this field. A tooltip with an error message appears in the user interface and an error message appears in the Task Console.

![Error message in the user interface and Task Console](image)

According to the code, the minimum value for this field is **0**, and the maximum value is **100**. Because 110 exceeds the maximum value, the default out of range message appears.
numericrange

This input type has these attributes:

Table 4.1  Attributes for numericrange

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>decimalPlaces</td>
<td>Specifies the range of acceptable decimal places (minimum, maximum). For example, decimalPlaces=&quot;0,3&quot;.</td>
</tr>
<tr>
<td>defaultFromValue</td>
<td>Specifies the default value for the From input field. The default value is null.</td>
</tr>
<tr>
<td>defaultToValue</td>
<td>Specifies the default value for the To input field. The default value is null.</td>
</tr>
<tr>
<td>fromLabel</td>
<td>Specifies the label for the From input field.</td>
</tr>
<tr>
<td>toLabel</td>
<td>Specifies the label for the To input field.</td>
</tr>
<tr>
<td>minInclusive</td>
<td>Specifies whether the minimum value is in the range of values in the From input field. By default, the minInclusive attribute is set to true, and these values can be equal.</td>
</tr>
<tr>
<td>maxInclusive</td>
<td>Specifies whether the maximum value is in the range of values in the To input field. By default, the maxInclusive attribute is set to true, and these values can be equal.</td>
</tr>
<tr>
<td>maxValue</td>
<td>Specifies the highest value that the user can specify. Any values greater than the maximum value are invalid. By default, the maxValue attribute is not set.</td>
</tr>
<tr>
<td>minValue</td>
<td>Specifies the lowest value that the user can specify. Any values less than the minimum value are invalid. By default, the minValue attribute is not set.</td>
</tr>
<tr>
<td>required</td>
<td>Specifies whether a value is required. By default, this attribute is false.</td>
</tr>
<tr>
<td>width</td>
<td>Specifies the width of the control. This value can be in percent (%), em, or px. By default, SAS Studio sizes the control based on the available width and content.</td>
</tr>
</tbody>
</table>

Here is an example:

```xml
<Option name="labelNUMERICRANGE" inputType="numericrange" minValue="-10" maxValue="100000" minInclusive="true" maxInclusive="true" decimalPlaces="0,3" defaultFromValue="10" defaultToValue="20" helpMessageRef="helpMessage" required="false" fromLabel="Low:" toLabel="High:">
  Specify the range of numbers
</Option>
```
numstepper
This input type has these attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>decimalPlaces</td>
<td>Specifies the number of decimal places to display. Valid values include a single value or a range. To create a field that allows 0 to 3 decimal places, specify decimalPlaces=&quot;0,3&quot;.</td>
</tr>
<tr>
<td>formatValue</td>
<td>Specifies whether the formatted number should include locale-specific delimiters. The default value is true. Note: The numeric value in the Velocity variable is not formatted.</td>
</tr>
<tr>
<td>increment</td>
<td>Specifies the number of values that the option increases or decreases when a user clicks the up or down arrow. The default value is 1.</td>
</tr>
<tr>
<td>invalidMessage</td>
<td>Specifies the tooltip text that appears when the content in the field is invalid.</td>
</tr>
<tr>
<td>maxValue</td>
<td>Specifies the maximum value that is allowed. If the user tries to exceed this value, a message appears. The default value is 9000000000000.</td>
</tr>
<tr>
<td>maxInclusive</td>
<td>Specifies whether to include the maximum value in the range of values. By default, this attribute is true.</td>
</tr>
<tr>
<td>minValue</td>
<td>Specifies the minimum value that is allowed. If the user specifies a value that is below the minimum value, a message appears.</td>
</tr>
<tr>
<td>minInclusive</td>
<td>Specifies whether to include the minimum value in the range of values. By default, this attribute is true.</td>
</tr>
<tr>
<td>missingMessage</td>
<td>Specifies the tooltip text that appears when the field is empty but a value is required.</td>
</tr>
<tr>
<td>promptMessage</td>
<td>Specifies the tooltip text that appears when the field is empty and the mouse is positioned over the field.</td>
</tr>
<tr>
<td>rangeMessage</td>
<td>Specifies the tooltip text when the value in the text box is outside the specified range.</td>
</tr>
<tr>
<td>required</td>
<td>Specifies whether a value is required. Valid values are true and false. The default value is false.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td>width</td>
<td>Specifies the width of the control. This value can be in percent (%), em, or px. By default, SAS Studio sizes the control based on the available width and content.</td>
</tr>
</tbody>
</table>

The first example in the Sample Task creates an option with an assigned default value of 5.

```xml
<Option name="labelNumStepperEXAMPLE1" inputType="string">
  An example of a basic numeric stepper.</Option>
<Option name="basicStepperEXAMPLE" defaultValue="5" inputType="numstepper" indent="1">Basic numeric stepper:</Option>
```

Here is an example of a numstepper control in the user interface:

```
An example of a basic numeric stepper.

Basic numeric stepper:

  ▼ 5 ▲
```

The second example in the Sample Task creates an option with a specified minimum value, maximum value, and increment.

```xml
<Option name="labelNumStepperEXAMPLE2" inputType="string">
  An example of a numeric stepper with a minimum value of -10, a maximum value of 120, and an increment of 2.</Option>
<Option name="advancedStepperEXAMPLE" defaultValue="80" inputType="numstepper" increment="2" minValue="-10" maxValue="120" decimalPlaces="0,2" width="8em" indent="1">Advanced numeric stepper:</Option>
```

When you run the code, here is the resulting user interface:

```
An example of a numeric stepper with a minimum value of -10, a maximum value of 120, and an increment of 2.

Advanced numeric stepper:

  ▼ 80 ▲
```

outputdata

The `outputdata` input type creates a text box where the user can specify the name of the output data set that is created by a task. The `outputdata` element enforces a two-level name in the format `library-name.data-set-name`. These names must follow SAS naming conventions. For more information, see “Names in the SAS Language” in *SAS Language Reference: Concepts*.

This input type has these attributes:
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>required</td>
<td>Specifies whether a name is required. The default value for this attribute is false, which means that no name is required.</td>
</tr>
<tr>
<td>width</td>
<td>Specifies the width of the control. The width can be specified in (percent) %, em, or px. By default, SAS Studio determines the size of the control based on the available width and content.</td>
</tr>
<tr>
<td>libraryEngineExclude</td>
<td>Specifies the engine types that are not valid for the data source. The engine types should be a comma-separated list. Possible values are V9 and CAS.</td>
</tr>
<tr>
<td>libraryEngineInclude</td>
<td>Specifies the engine types that are valid for the data source. The engine types should be a comma-separated list. Possible values are V9 and CAS.</td>
</tr>
<tr>
<td>unique</td>
<td>Specifies whether the value of the output control is unique. This is a Boolean value with a default value of false. If unique=true, the task ensures that the value of the outputdata control is unique when compared to all other outputdata control where unique=true.</td>
</tr>
</tbody>
</table>

**Note:** If you do not specify either the libraryEngineExclude attribute or the libraryEngineInclude attribute, all engine types are available for the data source control. If you need to limit the engine type, use either the libraryEngineExclude attribute or the libraryEngineInclude attribute. Do not specify both.

The defaultValue attribute contains the initial value for the output data set that is created by the task. SAS Studio checks to see whether this name is unique when you open the task. If the name is unique, the outputdata control in the task uses the default name specified. If the name is not unique, a suffix (starting with 0001) is added to the default name.

In this code example, the defaultValue attribute is Work.MyData. If no existing data sets use this name, Work.MyData appears as the name in the outputdata control. If a Work.MyData data set already exists, SAS Studio uses the suffix to create a unique name, such as Work.MyData0001. Using this technique prevents SAS Studio from overwriting an existing data set.

```xml
<Option defaultValue="Work.MyData" indent="1" inputType="outputdata" name="outputDSName" required="true">Data set name:</Option>
```

Here is an example of the outputdata control from the Sample Task:

An example of an output data selector.

Data set name: [WORK.MYDATA0007]

passwordtext

This input type has these attributes:
The Metadata Element

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>required</td>
<td>Specifies whether a name is required. The default value for this attribute is false, which means that no name is required.</td>
</tr>
<tr>
<td>width</td>
<td>Specifies the width of the control. The width can be specified in (percent) %, em, or px. By default, SAS Studio determines the size of the control based on the available width and content.</td>
</tr>
</tbody>
</table>

When using this control, remember these restrictions:

- When you save the task file, the password that is currently entered is not saved.
- The defaultValue attribute is not supported by the password control.
- The value of the password control cannot be set by a dependency.

Here is an example:

```xml
<Option name="pswd" inputType="passwordtext">Password:</Option>
```

An example of the password control. The password will be SAS002 encoded in the SAS code and will not be saved with the task.

Password:

---

**quarterpicker**

The quarterpicker attribute enables the user to choose a quarter and year.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>displayFormat</td>
<td>Specifies the visual formatting of the date. The valid values are short (default), medium, and long.</td>
</tr>
<tr>
<td>required</td>
<td>Specifies whether a date is required. By default, no date is required.</td>
</tr>
<tr>
<td>minValue</td>
<td>Specifies the minimum threshold for the quarter values. This attribute is not set by default.</td>
</tr>
<tr>
<td>maxValue</td>
<td>Specifies the maximum threshold for quarter values. This attribute is not set by default.</td>
</tr>
<tr>
<td>width</td>
<td>Specifies the width of the control. This value can be in percent (%), em, or px. By default, SAS Studio sizes the control based on the available width and content.</td>
</tr>
</tbody>
</table>

If you specify the defaultValue, minValue, and maxValue attribute for this input type, the value must be in yyyyQq format.

```xml
<Option name="qtrEXAMPLE" inputType="quarterpicker" defaultValue="2020Q2" minValue="2019Q1" maxValue="2020Q4" displayFormat="long">
Select the quarter:
```
radio

This input type has these attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>returnValue</td>
<td>Specifies that the value of the Velocity variable is the return value rather than the name of the radio control.</td>
</tr>
<tr>
<td>variable</td>
<td>Specifies the Velocity variable that contains the name (or return value) of the currently selected radio button.</td>
</tr>
</tbody>
</table>

One radio button in a group must be selected. If none of the values for the radio button list include the `defaultValue` attribute, the first button in the user interface is selected.

The example in the Sample Task creates an option called **Radio button group label** with **Radio button 1** selected by default.

```xml
<Options>
  <Option name="labelRADIO" inputType="string">An example of radio buttons. One radio button can be selected at a time.</Option>
  <Option name="radioButton1" variable="radioEXAMPLE" defaultValue="1" inputType="radio">Radio button 1</Option>
  <Option name="radioButton2" variable="radioEXAMPLE" inputType="radio">Radio button 2</Option>
  <Option name="radioButton3" variable="radioEXAMPLE" inputType="radio">Radio button 3</Option>
  ...
</Options>
```

Here is how this radio control appears in the user interface:
The `sasserverpath` control enables the user to choose a file or folder location on the SAS server. The default folder location is the user’s SAS home directory.

This input type has these attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>pathType</code></td>
<td>Specifies the type of selector for this control. Here are the valid values:</td>
</tr>
<tr>
<td></td>
<td>- <code>file</code> (default) enables the user to select a file from the SAS server.</td>
</tr>
<tr>
<td></td>
<td>- <code>folder</code> enables the user to select a folder. The folder must already exist. The user can create a new folder by using the Folder Selection window.</td>
</tr>
<tr>
<td><code>defaultExtension</code></td>
<td>Specifies the default extension for the file. If no value is specified, the default extension is <code>sas</code>.</td>
</tr>
<tr>
<td></td>
<td>Note: Available only if <code>pathType=&quot;file&quot;</code>.</td>
</tr>
<tr>
<td><code>defaultName</code></td>
<td>Specifies the default filename. If no value is specified, the default filename is <code>program</code>.</td>
</tr>
<tr>
<td></td>
<td>Note: Available only if <code>pathType=&quot;file&quot;</code>.</td>
</tr>
<tr>
<td><code>required</code></td>
<td>Specifies whether a selection is required. The default value is <code>false</code>.</td>
</tr>
<tr>
<td><code>width</code></td>
<td>Specifies the width of the control in percent (%), em, or px.</td>
</tr>
</tbody>
</table>

Here is an example of the `sasserverpath` control:

```xml
<Option name="fileSelector" inputType="sasserverpath" defaultFileName="myProgramFile" pathType="file" defaultExtension="sas">An example of a SAS Server Path control. This example allows the user to select a SAS program file.</Option>
```

An example of a SAS Server Path control. This example allows the user to select a SAS program file.

`/Public/score.sas`

---

The `select` control

This input type has these attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>allowAllValues</code></td>
<td>Specifies whether to add an item to the list so that the user can select all possible values. By default, this attribute is <code>false</code>, and no item is added to the list. You might use this attribute when the list is generated by a filter.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>allowMissingValues</td>
<td>Specifies whether to add an item to the list so that the user can select missing values. By default, this attribute is false, and no item is added to the list. You might use this attribute when the list is generated by a filter.</td>
</tr>
<tr>
<td>filter</td>
<td>Specifies that the data for this option should come from the specified filter. Note: If you specify the filter attribute, the sourceLink attribute and children of the OptionChoice element are ignored.</td>
</tr>
<tr>
<td>height</td>
<td>Specifies the height of the control in em or px.</td>
</tr>
<tr>
<td>multiple</td>
<td>Specifies whether users can select one or multiple items from the list. Valid values are true and false. The default value is true.</td>
</tr>
<tr>
<td>required</td>
<td>Specifies whether the user must select a value from the list. Valid values are true and false. The default value is false.</td>
</tr>
<tr>
<td>sourceLink</td>
<td>Specifies that the data for this control should come from another option. For more information about this attribute, see “Populating the Values for a Select Control from a Source Control” on page 68.</td>
</tr>
<tr>
<td>width</td>
<td>Specifies the width of the control in percent (%), em, or px.</td>
</tr>
</tbody>
</table>

Use the defaultValue attribute to specify the items that should be selected at run time. If you need to specify multiple items, use a comma-separated list.

The Sample Task creates an option called Select.

```xml
<Option name="labelSELECT" inputType="string">An example of a select.
   This example is set up for multiple selection.</Option>

<Option name="selectEXAMPLE" inputType="select" multiple="true">Select:</Option>

<UI>
...
<OptionItem option="labelSELECT" />
<OptionChoice option="selectEXAMPLE">
   <OptionItem option="value1"/>
   <OptionItem option="value2"/>
   <OptionItem option="value3"/>
</OptionChoice>
```

An example of a select. This example is set up for multiple selection.

Select:
- [ ] Value 1
- [ ] Value 2
- [ ] Value 3
slider

This input type has these attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>discreteValues</td>
<td>Specifies the number of discrete values in the slider. For example, if <code>discreteValues=&quot;3&quot;</code>, the slider has three values: a minimum value, a maximum value, and a value in the middle.</td>
</tr>
<tr>
<td>maxValue</td>
<td>Specifies the maximum value for this option.</td>
</tr>
<tr>
<td>minValue</td>
<td>Specifies the minimum value for this option.</td>
</tr>
<tr>
<td>showButtons</td>
<td>Specifies whether to show the increase and decrease buttons for the slide. Valid values are <code>true</code> and <code>false</code>. The default value is <code>true</code>.</td>
</tr>
</tbody>
</table>

The first example in the Sample Task creates a slider option with buttons.

```xml
<Option name="labelSliderEXAMPLE1" inputType="string">
    An example of a slider.</Option>
<Option name="labelSliderEXAMPLE1" defaultValue="80.00" inputType="slider" discreteValues="14" minValue="-10" maxValue="120">Slider</Option>
```

When you run the code, here is the resulting user interface:

![Slider](image)

string

The string input type can be used to display informational text to the user, to define strings for the `OptionChoice` element, to define string values that are used by the Velocity code, and to define text values to use for the Help Message feature.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>returnValue</td>
<td>Is the string that is returned in the control's Velocity variable (instead of the control's name). This attribute applies only when the string is used in an <code>OptionChoice</code> tag.</td>
</tr>
</tbody>
</table>

The code for the Sample Task contains several examples of the string input type. In the code for the slider option, the explanatory text (An example of a slider.) is created by the string input type.

```xml
<Option name="labelSliderEXAMPLE1" inputType="string">
    An example of a slider.</Option>
```
<Option name="labelSliderEXAMPLE1" defaultValue="80.00" inputType="slider" discreteValues="14" minValue="-10" maxValue="120">Slider</Option>

When you run the code, here is the resulting user interface:

An example of a slider with buttons.
Slider with buttons

An example of a slider.
Slider:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>required</td>
<td>Specifies whether any input text is required. Valid values are true and false. The default value is false.</td>
</tr>
<tr>
<td>width</td>
<td>Specifies the width of the control. This value can be in percent (%), em, or px. By default, SAS Studio sizes the control based on the available width and content.</td>
</tr>
<tr>
<td>height</td>
<td>Specifies the height of the control. This value can be in em or px. By default, SAS Studio sizes the control based on the available height and content.</td>
</tr>
<tr>
<td>splitLines</td>
<td>Specifies whether to split the text into an array of lines. The split is determined by the newline character. The default value is false.</td>
</tr>
</tbody>
</table>

If you specify the defaultValue attribute with this input type, you can specify the initial string to display in the text box. In this example, the text ‘Enter text here’ appears in the text box by default. Note the use of single quotation marks around the text. This example shows how you would include single quotation marks in your default text. These quotation marks are not required.

<Option name="textSimple" required="true" inputType="textbox" defaultValue="'Enter text here'">Text Box</Option>

Here is an example of a textbox control in the user interface. Note this example uses the default text. When the user types in the textbox control, this text disappears.
timepicker

The timepicker element enables you to choose a time.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>required</td>
<td>Specifies whether a date is required. By default, no date is required.</td>
</tr>
<tr>
<td>use24HourTime</td>
<td>Specifies whether to display the time using the 24-hour format instead of AM and PM. By default, this attribute is false, and the AM and PM format is used.</td>
</tr>
<tr>
<td>showSeconds</td>
<td>Specifies whether to show seconds in the time. By default, this attribute is false, and the seconds are not displayed.</td>
</tr>
<tr>
<td>width</td>
<td>Specifies the width of the control. This value can be in percent (%), em, or px. By default, SAS Studio sizes the control based on the available width and content.</td>
</tr>
</tbody>
</table>

If you specify the defaultValue attribute for this input type, the value must be in ISO8601 format (HH:mm:ss).

```xml
<Option name="timeEXAMPLE" inputType="timepicker" required="true"
  defaultValue="23:15:22" use24HourTime="true"
  helpMessageRef="timePickerLabel">Select the time:
</Option>
```

An example of a time picker.

Choose a time:

Select a time

11 ▼ : 25 ▼ AM ▼

OK  Cancel

validationtext

This input type enables the user to enter a string value. A regular expression can be used with this option to restrict the entered string to a specific format.

This input type has these attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hintMessage</td>
<td>Specifies the text to display when the control is empty.</td>
</tr>
<tr>
<td>invalidMessage</td>
<td>Specifies the tooltip text to display when the content in the text box is invalid. By default, no message is displayed.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>missingMessage</td>
<td>Specifies the tooltip text that appears when the text box is empty but text is required. By default, no message is displayed.</td>
</tr>
<tr>
<td>promptMessage</td>
<td>Specifies the tooltip text that appears when the text box is empty and the text box is selected. By default, no message is displayed.</td>
</tr>
<tr>
<td>regExp</td>
<td>Specifies the regular expression pattern to use for validation. This syntax comes directly from JavaScript Regular Expressions.</td>
</tr>
<tr>
<td>required</td>
<td>Specifies whether any input text is required. Valid values are true and false. The default value is false.</td>
</tr>
<tr>
<td>width</td>
<td>Specifies the width of the control. This value can be in percent (%), em, or px. By default, SAS Studio sizes the control based on the available width and content.</td>
</tr>
</tbody>
</table>

The code for the Sample Task creates a text box called **Validation text**.

```xml
<Option name="labelVALIDATIONTEXT" inputType="string">An example of a validation text. A regular expression of 5 characters has been applied.</Option>
<Option name="validationTextExample" defaultValue="99999" inputType="validationtext"
  promptMessage="Enter a string 5 characters long."
  invalidMessage="Invalid value. You must specify a string of 5 characters."
  regExp="\d{5}">Validation text: 99999</Option>
```

When you run the code, here is the resulting user interface:

```
An example of a validation text. A regular expression of 5 characters has been applied.

Validation text:
99999
```

If you remove the default value from this box, the **Enter a string 5 characters long** message appears.

When the user begins entering a value, this message appears: **Enter a string 5 characters long.**

If the specified value is more than five characters, the message for an invalid value appears as a tooltip and in the Task Console.
weekpicker

The weekpicker element enables you to choose a week and a year.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>required</td>
<td>Specifies whether a value is required. By default, no value is required.</td>
</tr>
<tr>
<td>minValue</td>
<td>Specifies the minimum value. Any values less than the minimum value are disabled. By default, the minValue attribute is not set.</td>
</tr>
<tr>
<td>maxValue</td>
<td>Specifies the maximum value. Any values greater than the maximum value are disabled. By default, the maxValue attribute is not set.</td>
</tr>
<tr>
<td>width</td>
<td>Specifies the width of the control. This value can be in percent (%), em, or px. By default, SAS Studio sizes the control based on the available width and content.</td>
</tr>
</tbody>
</table>

If you specify the defaultValue, minValue, and maxValue attributes for this input type, the value must be in ISO8601 format yyyy-Www.

```xml
<Option name="oWeek" inputType="weekpicker" defaultValue="2020-W07"
    minValue="2019-W15" maxValue="2020-W25">
    Select a week:
</Option>
```

Organizing Options into a Table Component

The OptionTable element defines a table component that contains one or more custom-defined columns. Each column contains one CTM option. Each individual column can contain a different CTM option. Here are the available CTM options:

- checkbox
- combobox
- numbertext
Each row in the column has the same CTM control. If you specify the addRemoveRowTools attribute, users can add and delete rows from the table.

Here is an example from the sample task:

An example of an option table control. This control allows for each column of the table to render a different CTM control. Valid control types are: checkbox, combobox, numtext, numstepper, inputtext.

<table>
<thead>
<tr>
<th>Text</th>
<th>Number Stepper</th>
</tr>
</thead>
<tbody>
<tr>
<td>hello</td>
<td>3</td>
</tr>
</tbody>
</table>

Use these attributes to create the table:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Specifies the name assigned to the option.</td>
</tr>
<tr>
<td>label</td>
<td>Specifies the label for the table in the user interface.</td>
</tr>
<tr>
<td>indent</td>
<td>Specifies the indention for this option in the task interface. Here are the valid values:</td>
</tr>
<tr>
<td></td>
<td>1 – minimal indention (about 17px)</td>
</tr>
<tr>
<td></td>
<td>2 – average indention (about 34px)</td>
</tr>
<tr>
<td></td>
<td>3 – maximum indention (about 51px)</td>
</tr>
<tr>
<td>addRemoveRowTools</td>
<td>Specifies whether to enable the user to add and remove rows from the table. Valid values are true and false. When this value is set to true, icons for adding and removing rows appear above the table. By default, this value is false, so the task interface contains only the number of rows that you specified using the initialNumberOfRows attribute.</td>
</tr>
<tr>
<td>initialNumberOfRows</td>
<td>Specifies the number of empty rows in a new table. This value must be greater than or equal to 1. By default, this value is 1.</td>
</tr>
<tr>
<td>maximumRows</td>
<td>Specifies the maximum number of rows in the option table. The default value is 0.</td>
</tr>
<tr>
<td>minimumRequiredRows</td>
<td>Specifies the minimum number of rows that must be completed. This value must be greater than or equal to 1. The default value is 1.</td>
</tr>
</tbody>
</table>
Specifies whether incomplete rows are allowed in the table. Valid values are **true** and **false**. The default value is **false**. If this attribute is set to **true**, the task cannot run if there are any incomplete rows in the table.

**showColumnHeadings**

Specifies whether to show the column headings in the table. Valid values are **true** and **false**. The default value is **false**, and no column headings are displayed.

The **OptionTable** element can have only one child, the **Columns** element. The **Columns** element can contain multiple **Column** elements. Each **Column** element describes a column in the table.

Each column must be defined in an **Option** element in the metadata. In the **Option** element, the values for the **name** and **width** attributes are ignored. Specify the initial column width by using the **width** attribute in the **Column** element.

You can use these input types for the columns in the option table:

- checkbox
- combobox
- numbertext
- numstepper
- textbox

Here are the attributes for the **Column** element:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Specifies the name of the column. This attribute is required.</td>
</tr>
<tr>
<td>label</td>
<td>Specifies the label of the column.</td>
</tr>
<tr>
<td>defaultValues</td>
<td>Specifies a list of default values for the first several rows. These values apply only when the table is created. If this attribute is not specified for the column, the value of <code>defaultValue</code> for the cell is used instead. The <code>defaultValues</code> column attribute takes precedence over the <code>defaultValue</code> cell attribute.</td>
</tr>
<tr>
<td>width</td>
<td>Specifies the initial width of the column. This width is in pixels. If you do not specify a width, the column width is an estimate based on the properties of the column widget.</td>
</tr>
</tbody>
</table>

Here is an example that uses the **OptionTable** element:

```xml
<OptionTable name="optionTable" initialNumberOfRows="3" addRemoveRowTools="false">
  <Columns>
    <Column name="colNumberText" label="NumberText" labelKey="alphaKey">
```

```xml
```
Specifying a Return Value Using the returnValue Attribute

When you specify the returnValue attribute on an Option element, the string that is specified for the returnValue attribute is returned instead of the name.

For input types (such as combobox and select) that enable users to select from a list of choices, the default behavior is to return the name of the selected item in the list. However, because the name attribute must be unique for every option, this default behavior could be limiting in some scenarios.

For options that support the dataType attribute (such as dualSelector and mutlientry), the returnValue attribute must be specified when the data type is number or date.

- When the data type is number, the return value must be a number.
- When the data type is date, the return value must be in the ISO format (yyy-mm-dd).

The following example is available from the Advanced Task. In this example, the $vegetables Velocity variable has the value of 1, 2, or 3, depending on what option item the user selected in the user interface. If you do not specify the returnValue attribute, the Velocity variable returns carrots, peas, or corn.

```xml
<Options>
  <Option name="RETURNVALUETAB" inputType="string">RETURN VALUE</Option>
</Options>
```
This tab shows an example of the option's `returnValue` attribute. This attribute can be used in the `OptionChoice` controls to customize Velocity return values.

Select the vegetables:

- [ ] Carrots
- [ ] Peas
- [ ] Corn

If you run the Advanced Task, here is the resulting **Return Value** tab.

This tab shows an example of the option's `returnValue` attribute. This attribute can be used in `OptionChoice` controls to customize velocity return values.

Populating the Values for a Select Control from a Source Control

About Data Linking

Data linking is a way to populate a control based on the contents of another control. Data linking is currently supported when a select control links to data from a role or
from the mixed effects control. If the select control links to anywhere else, any children in the OptionChoice element are ignored.

The combobox and select controls can be the recipient of the data. For these controls, specify the source by defining the sourceLink attribute and using the name of the source control. When the receiving option is linked to a source option, any OptionChoice children are ignored.

The Velocity code that is returned for the select control uses the same Velocity structure that you would expect from the source control.

This example is from the Advanced Task.

```xml
<Option name="DATALINKINGTAB" inputType="string">DATA LINKING</Option>
<Option name="DATALINKINGTEXT" inputType="string">This tab shows examples of data linking. Data linking allows controls to be populated based on data from another control</Option>
<Option name="ROLELINKING" inputType="string">LINKING TO ROLES</Option>
<Option name="selectRoles" inputType="select" multiple="true" sourceLink="dataVariables">This select is populated from the Variables selected from the Data tab.</Option>
<Option name="MEBLINKING" inputType="string">LINKING TO MIXED EFFECTS CONTROL</Option>
<Option name="selectMEB" inputType="select" multiple="true" sourceLink="mixedeffects">This select is populated from the output of the Mixed Effects Control.</Option>

...<UI>
  <Container option="DATALINKINGTAB">
    <OptionItem option="DATALINKINGTEXT"/>
    <Group option="ROLELINKING" open="true">
      <OptionChoice option="selectRoles"/>
    </Group>
    <Group option="MEBLINKING" open="true">
      <OptionChoice option="selectMEB"/>
    </Group>
  </Container>
...</UI>
```

If you run the code for the Advanced Task, here is the resulting Data Linking tab.
This tab shows examples of Data Linking. Data linking allows controls to be populated based on data from another control.

**LINKING TO ROLES**

This select control is populated with the variables selected on the Data tab.

No items

**LINKING TO MIXED EFFECTS CONTROL**

This select control is populated from the output of the mixed effects control.

No items

**Linking to a Role**

If a select control is linked to a role, the values in the select control are the current list of roles in the roles option. In this example, the name of the role variable is NUMVAR (specified in the `name` attribute). In the select control, the `sourceLink` attribute links to NUMVAR.

```xml
<DataSources>
  <DataSource name="PRIMARYDATA">
    <Roles>
      <Role type="N" maxVars="0" order="true" minVars="0" name="NUMVAR" exclude="VAR">Numeric Variable</Role>
    </Roles>
  </DataSource>
</DataSources>
```
The Velocity variable that is created for the select control is $roleList. The contents of the $roleList variable mimic the output of a typical role control. For more information, see “Working with Role Elements in the Velocity Code” in SAS Studio: Developer’s Guide to Writing Custom Tasks.

Linking to Effects from the Mixed Effects Control

If a select control is linked to a mixedeffects input type, the values in the select control are the list of effects in the mixed effects control.

An additional attribute called sourceType can be used to set a filter on the data that is sent to the select control. Currently, the only defined filter is ‘filterClassification’. When this filter is specified, only classification effects appear in the select control.

In this example, the mixedeffects control is named MEC. In the select control, the sourceLink attribute links to MEC, and the sourceType attribute specifies the ‘filterClassification’ filter. As a result, only classification effects appear in the source control.

The Velocity variable that is created for the select control is $mecList. The contents of the $mecList variable mimic the output of the mixed effects control. For more information, see “mixedeffects” in SAS Studio: Developer’s Guide to Writing Custom Tasks.

Another example is in the Linear Regression task. In this task, the effects listed in the mixed effects control are the options for the Select the effects to test option on the Options tab.

The Variables pane in the Model Effects Builder lists the variables that the user assigned to either the Classification variables role or the Continuous variables role. The user can create main, crossed, nested, and polynomial effects. These effects appear in the Model effects pane.
On the **Options** tab, all classification effects are available from the **Select effects to test** option.

- **Multiple Comparisons**
  - **Perform multiple comparisons**
    - **Select effects to test:**
      - [ ] line
      - [ ] product

Here are the relevant portions of code from the Linear Regression task:

```xml
<Option inputType="string" name="modelGroup">MODEL EFFECTS</Option>
<Option inputType="string" name="modelTab">MODEL</Option>
<Option inputType="mixedeffects" name="mixedEffects" excludeTools="POLYEFFECT,TWOFAC,TREFFAC,NFACTPOLY" roleClassification="classVariable" roleContinuous="continuousVariables" width="100%">Model</Option>
...
<Option inputType="string" name="multCompareGroup">Multiple Comparisons</Option>
```
1 Creates the mixed effects control on the Models tab. Classification variables and continuous variables can be used to create the model effects.

2 Creates the Select effects to test option. The sourceLink attribute specifies that the initial list of values for this option is the list of model effects in the Model Effects Builder. The sourceType attribute filters the list generated by the sourceLink attribute. The filterClassification filter specifies that only effects that include the classification variable should be available in the Select effects to test option.

In the Perform multiple comparisons option, the initial list of model effects includes region, line, product, region(line), line(product), and cost. However, cost is a continuous variable. When this list is filtered, only the model effects that involve classification variables (region, line, and product) are listed as values for the Select effects to test option.

---

Specifying a Help Message

A help message can be associated with most options as well as roles by using the helpMessageRef attribute. This attribute can refer to a string or a markdown option. When a help message is associated with an option, a help icon is displayed to the right of the option's label.

This example shows adding a help message to the textbox input type.

```xml
<Options>
  <Option name="Cont1" inputType="string">TAB</Option>
  <Option name="helpString" inputType="string">This is a helpful message</Option>
  <Option name="helpMarkdown" inputType="markdown">
    Numbered list:
    
    1. wash
    2. rinse
    3. repeat

    ## Links

    [example]: http://sas.com "Optional title here"

    This is an example of a reference-style link: [link][example].

    [link text](http://support.sas.com)

    ## Images

  </Option>
</Options>
```
// Wire the helpMarkdown into the 'textbox' input type using the
// 'helpMessageRef' attribute
<Option name="txtMarkdown" inputType="textbox" helpMessageRef="helpMarkdown">
    Click the '?' to see the help message in markdown:</Option>

// Wire the helpMarkdown into the 'textbox' input type
// using the 'helpMessageRef' attribute
<Option name="txtString" inputType="textbox" helpMessageRef="helpString">
    Click the '?' to see the help message as a simple string:</Option>

</Options>
</Metadata>

<UI>
    <Container option="Cont1">
        <OptionItem option="txtMarkdown"/>
        <OptionItem option="txtString"/>
    </Container>
</UI>
The UI Element

About the UI Element

This element is read by the UI engine to determine the layout of the user interface. Only linear layouts are supported. The UI tag is for grouping purposes only. There are no attributes associated with this tag.

Option References

To include an option in the UI section, use one of these option tags.

<table>
<thead>
<tr>
<th>Child</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DataItem</td>
<td>A reference to an input data source. This tag has only one attribute, data. The string for this option is the value of the string input type in the Metadata element.</td>
</tr>
<tr>
<td>RoleItem</td>
<td>A reference to a role. This tag has only one attribute, role. The string for this option is the value of the string input type in the Metadata element.</td>
</tr>
<tr>
<td>OptionItem</td>
<td>A reference to an option that has a single state. This type of option is either on or off, or has a single value (such as a series of radio buttons). This tag takes the option attribute only. The option attribute refers to the metadata name attribute for the option. The string for this option is taken from the metadata string value.</td>
</tr>
</tbody>
</table>
### OptionChoice

A reference to an option that has a choice of values. The `OptionChoice` element uses the `OptionItem` or `OptionValue` element to represent the choice of values.

These input types can use the `OptionChoice` element in the user interface:

- `combobox`
- `distinct`
- `dualselector`
- `multiedit`
- `select`

This tag takes the `option` attribute only. The `option` attribute refers to the metadata name attribute for the option. The string for this option is taken from the metadata string value.

### OptionValue

A value choice. This tag is valid only as a child of the `OptionChoice` element.

---

## Containers

Options can be grouped into a container (which appears as a tab in the user interface) or a group (which appears as a collapsible pane in the window). By default, the options are laid out linearly. To lay out the option horizontally, use the `HorizontalLayout` element.

<table>
<thead>
<tr>
<th>Child</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Container</td>
<td>A page or tab that contains any options for the task. For example, you might want to display the option for selecting the input data and assigning columns to roles on the same page. The UI engine displays these options sequentially. A label is created for the tab. The <code>Container</code> tag takes only one attribute. The string for this option is the value of the <code>string</code> input type in the <code>Metadata</code> element.</td>
</tr>
<tr>
<td>Group</td>
<td>A title for a group of options. The UI engine displays these options sequentially. The group can be open or collapsed. This tag takes these attributes: The <code>option</code> attribute is an option name in the metadata. This string is the same as the string value for the metadata option. The <code>open</code> attribute specifies whether a group is expanded or collapsed. By default, <code>open=&quot;true&quot;</code>, and the group is open in the user interface. To collapse the contents of a group by default, specify <code>open=&quot;false&quot;</code>.</td>
</tr>
</tbody>
</table>
### HorizontalLayout

Options are displayed horizontally. The HorizontalLayout element has only one attribute, `option`. The `option` attribute is optional. You need to specify it only if the HorizontalLayout tag needs to be targeted through a dependency.

---

**Example: UI Element from Sample Task**

The code for the Sample Task creates a group for each input type. Here is the code for the first three groups:

```xml
<UI>
  <Container option="DATATAB">
    <Group option="DATAGROUP" open="true">
      <DataItem data="DATASOURCE" />
    </Group>
    <Group option="ROLESGROUP" open="true">
      <RoleItem role="VAR"/>
      <RoleItem role="OPTNVAR"/>
      <RoleItem role="OPTCVAR"/>
    </Group>
  </Container>

  <Container option="OPTIONSTAB">
    <Group option="GROUP" open="true">
      <OptionItem option="labelEXAMPLE"/>
    </Group>
    <Group option="GROUPCHECK">
      <OptionItem option="labelCheck"/>
      <OptionItem option="chkEXAMPLE"/>
    </Group>
    <Group option="GROUPCOLOR">
      <OptionItem option="labelCOLOR"/>
      <OptionItem option="colorEXAMPLE"/>
    </Group>
  </Container>

  ...
</UI>
```

When you run this code, the **Data** and **Options** tabs appear in the interface. The **Data** tab displays a selector for the input data source and three roles.
The Options tab contains several groups. The previous code creates the Groups, Check Boxes, and Color Selector groups. The first group is expanded by default because the open attribute is set to true. (The Sample Task includes code to create the remaining groups on the Options tab.)
About the OptionDependencies Element

To define dependencies between prompts, use the OptionDependencies element.

When using the OptionDependencies element, remember the following information:

- The OptionsDependencies element is intended to be simple. Only the ShowTarget and EnableTarget elements are available. There are no show or hide attributes. When the value of the TriggerOption matches the value of TriggerValue, everything is displayed. All other trigger values hide or disable the controls.
- If two different TriggerOption values match, the result is the union of both conditions.
- You can target groups with option dependencies instead of targeting a large number of options with a single option dependency. However, you should use either target groups or target individual elements. Mixing these two behaviors can lead to unexpected results.
- Use either Dependencies elements or OptionsDependencies elements. If the task has any Dependencies elements, the OptionDependencies are ignored.

OptionDependency Element

The OptionsDependencies element can have 0 to n OptionDependency elements. Each OptionDependency element defines the relationship between the trigger option and any targets for the trigger. The OptionDependency element has no attributes. It must contain one TriggerOption element and 1 to n TriggerValue elements.
<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TriggerOption</td>
<td>The TriggerOption element has one required attribute, option. Use this element to specify the option whose host value is compared to the TriggerValue elements to determine whether to show or hide or enable and disable an option. The TriggerOption element has no children.</td>
</tr>
<tr>
<td>TriggerValue</td>
<td>The TriggerValue element has one attribute, value. This value is compared to the value of the option in the TriggerOption element. If the value in the TriggerOption is equal to the value of the TriggerValue element, the target children are shown or enabled. The TriggerValue element can have 1 to n children that are either ShowTarget or EnableTarget elements.</td>
</tr>
<tr>
<td>ShowTarget</td>
<td>The ShowTarget element has one required attribute, option. Use this attribute to specify the name of the option that should be shown if the value matches the TriggerOption value. The ShowTarget element has no children.</td>
</tr>
<tr>
<td>EnableTarget</td>
<td>The EnableTarget element has one required attribute, option. This attribute is used to specify the name of the option that should be enabled if the value matches the TriggerOption value. The EnableTarget element has no children.</td>
</tr>
</tbody>
</table>

**Example: OptionDependencies Element**

In this example, when the value of checkbox is 1, the inputtext control is shown. If the value of the checkbox is not 1, the inputtext control is hidden.

```xml
<Metadata>
  <Options>
    <Option name="cbxValue" inputType="checkbox">Specify value</Option>
    <Option name="txtValue" inputType="inputtext">Enter a value</Option>
  </Options>
</Metadata>

<UI>
  <OptionItem option="cbxValue"/>
  <OptionItem option="txtValue"/>
</UI>

<OptionDependencies>
  <OptionDependency>
    <TriggerOption option="cbxValue"/>
    <TriggerValue value="true">
      <ShowTarget option="txtValue"/>
    </TriggerValue>
  </OptionDependency>
</OptionDependencies>
```
</OptionDependency>

</OptionDependencies>