

What is a custom data type?

Custom data type is a data type used on a step property.

The screenshot shows the cDevWorkflow v8.5 interface. The left sidebar contains navigation options: Dashboard, Definitions, Instances, Tasks, Forms, Business Intelligence, Steps, Enterprise Service Bus, Archive, Users, Status, and Help. The main area is titled 'Steps / Manage Step Inputs and Returns'. It displays configuration for a step named 'decision'. Fields include Step Name, Step Description (Computes decisions), Step Index (0), Step Category (Engine), Step Namespace (cDevWorkflow.cDevDecisionEngine.decisionStep), and DLL Path (cDevDecisionEngineSteps.dll). There are 'Update' and 'Validate' buttons. Below this is an 'Upload icon file for the steps:' section with a 'Choose File' button and 'No file chosen' text, and an 'Upload Icon' button. The 'Step Inputs' section has 'Add', 'Move Down', and 'Move Up' buttons. A table below lists the step inputs:

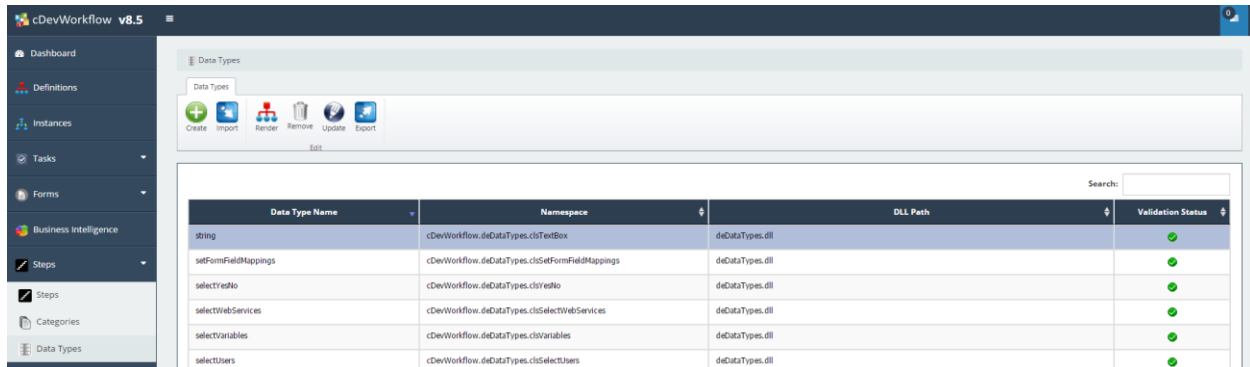
Input Name	Input Description	Data Type
condition	condition to evaluate	string

In reference to the above diagram, the decision step has 1 property called “condition” that is “string” data type. When the step is rendered within the workflow designer, the “condition” property is rendered as a textbox.

The screenshot shows a workflow designer interface. It displays a step configuration for a step named 'POAmount greater than 100'. The step has 1 incoming connection and 2 outgoing connections. The 'Name' field contains 'POAmount greater than 100'. The 'Description' field is empty. The 'condition to evaluate:' field is highlighted with a red box and contains the expression 'Variable.POAmount > 100'. Below this is a 'Log message:' field with a placeholder '!' and a 'Documentation:' field. A 'Save' button is visible at the bottom. A 'Variables' section is shown at the bottom left with a plus icon.

Custom data types

Custom data types can be configured using the data type screen of the cDevWorkflow Configuration Manager.



The screenshot shows the 'Data Types' configuration screen in the cDevWorkflow v8.5 interface. The left sidebar contains navigation options: Dashboard, Definitions, Instances, Tasks, Forms, Business Intelligence, Steps, Categories, and Data Types. The main content area shows a table of data types with the following columns: Data Type Name, Namespace, DLL Path, and Validation Status. The 'string' data type is selected and highlighted.

Data Type Name	Namespace	DLL Path	Validation Status
string	cDevWorkflow.deDataTypes.csTextBox	deDataTypes.dll	✓
setFormFieldMappings	cDevWorkflow.deDataTypes.csSetFormFieldMappings	deDataTypes.dll	✓
selectYesNo	cDevWorkflow.deDataTypes.csYesNo	deDataTypes.dll	✓
selectWebServices	cDevWorkflow.deDataTypes.csSelectWebServices	deDataTypes.dll	✓
selectVariables	cDevWorkflow.deDataTypes.csVariables	deDataTypes.dll	✓
selectUsers	cDevWorkflow.deDataTypes.csSelectUsers	deDataTypes.dll	✓

When the “string” data type is rendered, it looks as follows:



Writing a custom data type

Custom data type can be written easily by implementing the “IDataType” interface. The following code shows the actual code for the “string” data type.

```
public class clsTextBox : IDataType
{
    public string render(string controlName, string[] defaultValues, string selectedValue, HttpRequest oRequest)
    {
        TextBox oBox = new TextBox();
        oBox.ID = controlName;
        oBox.Width = Unit.Percentage(99);

        if (selectedValue != null)
        {
            oBox.Text = selectedValue;
        }

        StringWriter stringWriter = new StringWriter();
        HtmlTextWriter writer = new HtmlTextWriter(stringWriter);

        oBox.RenderControl(writer);

        return (stringWriter.ToString());
    }
}
```

Data type interface “IDataType” has one single method that needs to be implemented:

```
public string render(string controlName, string[] defaultValues, string selectedValue, HttpRequest oRequest)
```

The return from the “render” function is to return a HTML string to render the HTML controls for the property.

Configuring the custom data type

Once the custom data type is compiled into a DLL, it can be configured through the cDevWorkflow Configuration Manager. Navigate to the “Data Types” page within the Configuration Manager.

Data Type Name	Namespace	DLL Path	Validation Status
checkbox	cDevWorkflow.d4DataTypes.csCheckbox	d4DataTypes.dll	●
clnEnterLinkObjects	cDevWorkflow.d4DataTypes.clnEnterLinkObjects	d4DataTypes.dll	●
clnEnterReplyOptions	cDevWorkflow.d4DataTypes.clnEnterReplyOptions	d4DataTypes.dll	●
clExpirations	cDevWorkflow.d4DataTypes.clExpirations	d4DataTypes.dll	●
clMinutesDayHours	cDevWorkflow.d4DataTypes.clMinutesDayHours	d4DataTypes.dll	●
clMinutesDayHoursMonthsYears	cDevWorkflow.d4DataTypes.clMinutesDayHoursMonthsYears	d4DataTypes.dll	●

Click the “Create Data Type” button to configure the definition for the new step.

Create Data Type

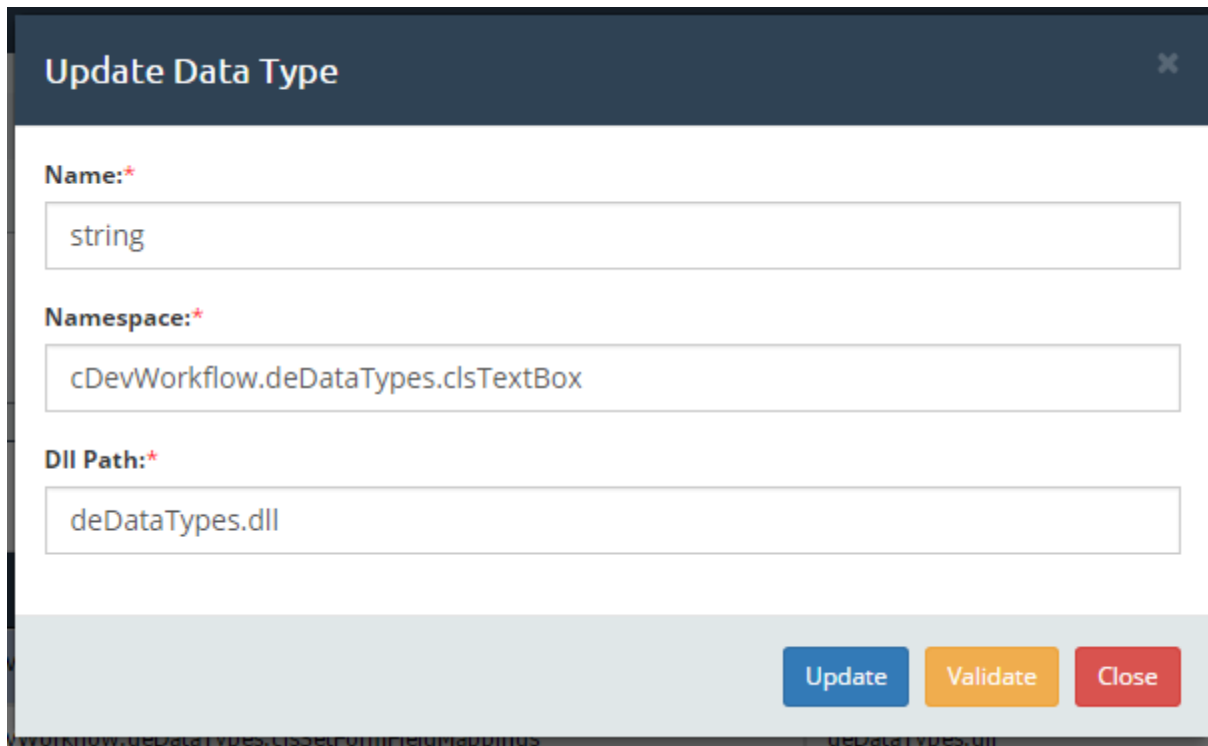
Name:*

Namespace:*

Dll Path:*

Create
Validate
Close

Here's what the configuration looks for the "string" data type step:



Update Data Type

Name:*
string

Namespace:*
cDevWorkflow.deDataTypes.clsTextBox

Dll Path:*
deDataTypes.dll

Update Validate Close

Once the data type is configured, data type can be used for defining properties for a given step.

Building and configuring an advanced data type

Advanced data types are data types that perform complex operations and store the value in a property. Data type "SelectUsers" is one of them. The following is the actual code for the step:

```
public class clsSelectUsers : IDatatype
{
    public string render(string controlName, string[] defaultValues, string selectedValue, HttpRequest oRequest)
    {
        StringWriter stringWriter = new StringWriter();
        HtmlTextWriter writer = new HtmlTextWriter(stringWriter);

        Button oBtn = new Button();
        oBtn.Text = "...";
        oBtn.OnClientClick = string.Format("window.open('deSelectUsers.aspx?fileName={0}', '', 'resizable=1,status=1,scrollbars=1,width=500,height=350');", controlName);

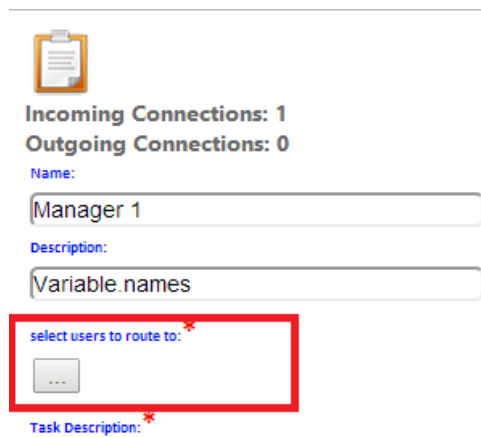
        oBtn.UseSubmitBehavior = false;

        oBtn.RenderControl(writer);

        string sHTML = HttpUtility.HtmlDecode(stringWriter.ToString());

        return (sHTML);
    }
}
```

The above code renders a button for the property and the button launches the window for user selection.



The screenshot shows a web application interface with the following elements:

- A clipboard icon at the top left.
- Summary statistics: "Incoming Connections: 1" and "Outgoing Connections: 0".
- A "Name:" label followed by a text input field containing "Manager 1".
- A "Description:" label followed by a text input field containing "Variable names".
- A red-bordered box highlighting a label "select users to route to:" with a red asterisk, and a small button with a dropdown arrow below it.
- A "Task Description:" label with a red asterisk at the bottom.

Clicking the button will display the following User selection dialog:

