ComponentOne

MultiSelect for WinForms

GrapeCity US

GrapeCity
201 South Highland Avenue, Suite 301
Pittsburgh, PA 15206
Tel: 1.800.858.2739 | 412.681.4343
Fax: 412.681.4384
Website: https://www.grapecity.com/en/
E-mail: us.sales@grapecity.com

Trademarks

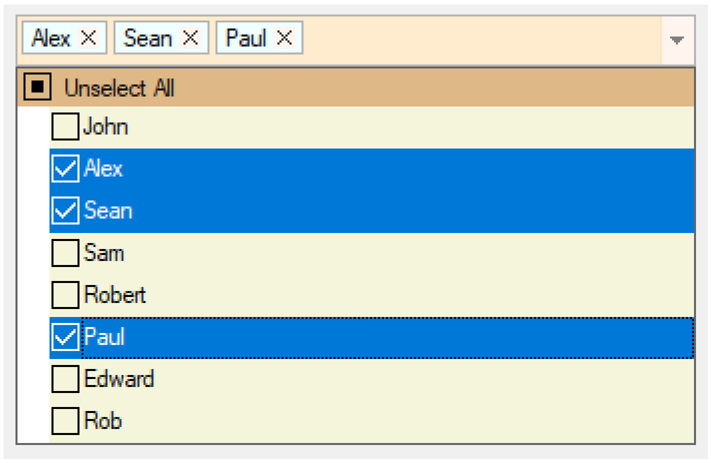
Warranty

Copying and Distribution

Table of Contents

MultiSelect for WinForms

MultiSelect for WinForms is a control that provides the ease of selecting multiple objects from a list or a collection of selected items. It comprises two elements, C1TagEditor and C1CheckList, which can be used as stand-alone controls as well. Flexible C1TagEditor gives you an option to display the selected items either as strings or as tags so that you can easily give your application an Office365 Outlook-like interface. Not just this, this control is smart enough to display the summarized text or tag instead of all selected items if the count goes beyond a specified limit. Moreover, C1CheckList element of the MultiSelect control allows you to highlight your selection as a checklist or as a simple list of items. Above all, the control also supports data binding with TagDataSource and DataSource apart from the unbound mode.



Help with WinForms Edition

For information on installing **ComponentOne Studio WinForms Edition**, licensing, theming, technical support, namespaces and creating a project with the control, please visit Getting Started with WinForms Edition.

Redistributable Files

MultiSelect for WinForms is developed and published by GrapeCity, Inc. You may use it to develop applications in conjunction with the Microsoft Visual Studio or any other programming environment that enables the user to use and integrate the control(s). You may also distribute, free of royalties, the following Redistributable Files with any such application you develop to the extent that they are used separately on a single CPU on the client/workstation side of the network.

The following assemblies get referenced automatically on adding MultiSelect control to the Form.

- C1.Win.4.dll
- C1.Win.C1Input.4.dll
- C1.Win.TreeView.4.dll

Site licenses are available for groups of multiple developers. Please contact Sales@ComponentOne.com for details.

Key Features

- **Effortless Selection**
  MultiSelect allows you to select specific or all items from a defined list of items, which are displayed in the control header post selection. It provides you control over selection of items through Single, Multiple, Extended modes which define how items in the list can be selected. MultiSelect also allows you to access selected items from the list which further can be used as data source for any other control to display items as per the requirement.

- **Add/Remove Items**
  MultiSelect allows you to add and remove items from the list through code or directly from the control header.

- **Smart Header**
  MultiSelect allows you to control the number of items to be displayed in header. The control header displays selected items if the number of selected items is less than or equal to the value set for the MaxHeaderItems property and if number of selected items is greater than MaxHeaderItems the header displays the count of selected items.

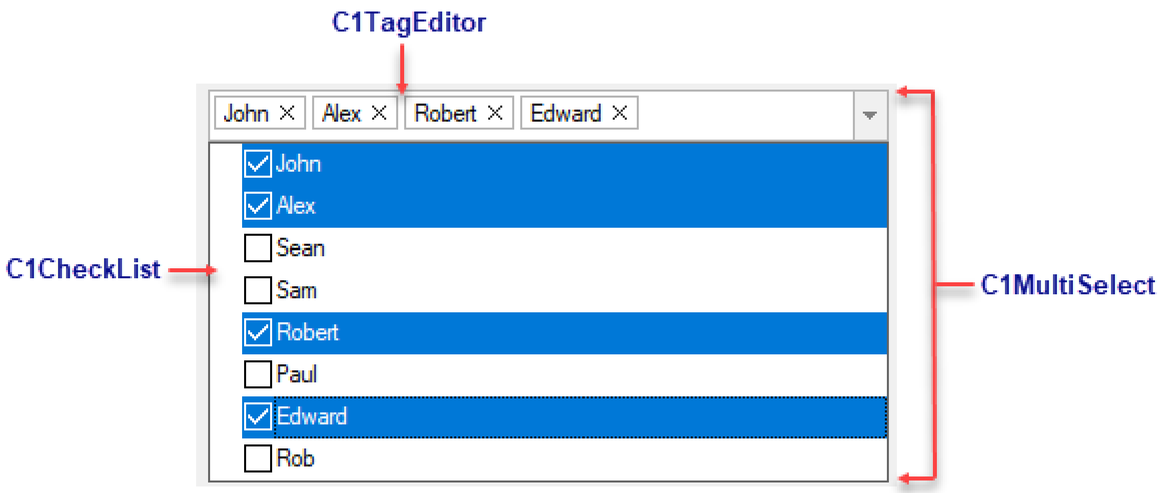
- **Edit Mode**
  MultiSelect supports text input which makes the selected tags editable in both the appearances – comma separated strings and tags. To change an item, you can simply double click on it to edit.

- **Bind to DataSource**
  MultiSelect can be bound to two different data sources, TagsDataSource and DataSource. TagsDataSource can be used if the application author wants to propagate end-user selection to some other data source, probably with other data structure than the first one. On the other hand, DataSource is used to fill the list of all available items that are shown in dropdown checklist.

Elements

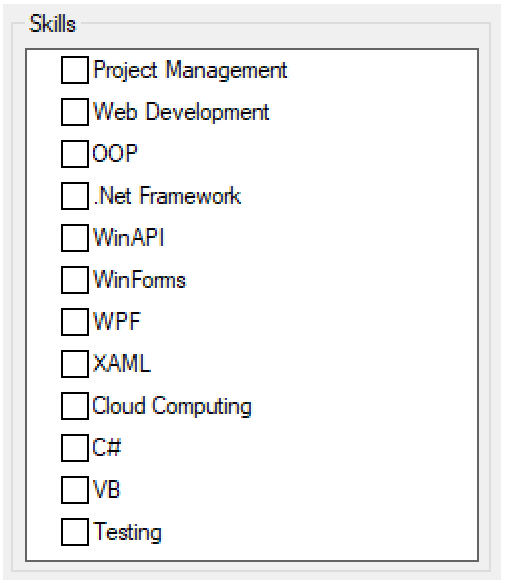
The MultiSelect control mainly consists of two elements which are available in **C1.Win.Input.C1MultiSelect.dll** and can be used as stand-alone controls as well.

- **C1CheckList**
- **C1TagEditor**



C1CheckList

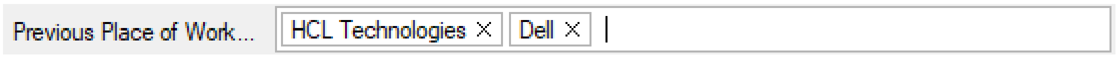
The C1CheckList control, instantiated using C1Checklist class, displays a collection of items in a static list, and allows user to select desired items from a defined list. For example, C1CheckList can be used to display a list of skills in a CV form on a job portal as shown in the following image.



To understand implementation of the control, refer CVForm sample available at the default installation folder.
Documents\ComponentOne Samples\WinForms\MultiSelect

C1TagEditor

The C1TagEditor control, instantiated using C1TagEditor class, provides the user with a textbox area where each tag behaves as an individual entry which can be inserted, edited, and removed individually. For example, C1TagEditor can be used to enter the name of all the previous workplaces in a CV form on a job portal. These names are added in the control as individual tags which can be edited or removed later.



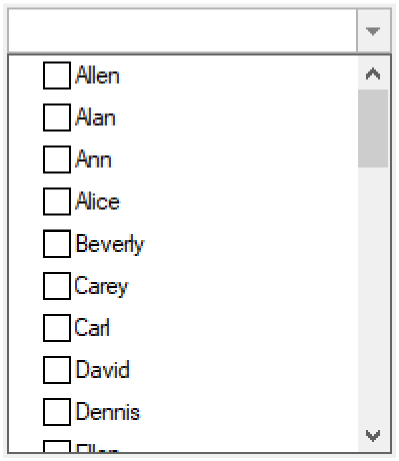
Quick Start

This quick start will guide you through the steps of adding C1MultiSelect to a project and binding the control to a data source.

Complete the steps given below to see how the MultiSelect control appears after data binding.

1. **Adding MultiSelect control to the Application**
2. **Binding MultiSelect to a DataSet**

The following image shows how the MultiSelect control appears after data binding.



Step 1: Adding MultiSelect control to the Application

1. Create a new **Windows Forms App** in **Visual Studio**.
2. Drag and Drop the **C1MultiSelect** control from the toolbox onto the form. The C1MultiSelect control is added to the form.

Step 2: Binding MultiSelect to a DataSet

1. In **Properties** window, navigate to the **DataSource** attribute in the **BindingInfo** property.
2. Click **Add Project Data Source** to open the **Data Source Configuration Wizard**. The Data Source Configuration Wizard appears with a database selected as a data source.
3. Click **Next** to select a database model.
4. Click the **New Connection** button to locate and connect to a database. The **Add Connection** dialog box opens.
5. Click the **Browse** button and locate the **C1NWind.mdb** file. Select it and click **Open**.
6. Click the **Test Connection** button to make sure that you have successfully connected to the database or server and click **OK** in the dialog window confirming the connection succeeded.
7. Click **OK** to close the **Add Connection** dialog box and return to the **Data Source Configuration Wizard**. The new string appears in the data connection drop-down box.
8. Click the **Next** button to continue. A dialog box appears asking if you would like to add the data file to your project and modify the connection string. Click **No**.
9. In the **Choose Your Database Objects** window, you can select the tables and fields that you would like in your dataset. We have selected the Customer table, the fields within the Customer table should all be selected as well.
10. Click **Finish** to exit the wizard. The dataset, binding source and table adapter now appear on your form.
11. Switch to the code view to see that the following code is added to the **Form_Load** event:
    ```
    this.customerTableAdapter.Fill(this.c1NWindDataSet.Customer);
    ```
12. Switch back to the design view and navigate to the **Properties** window.
13. In **BindingInfo** property, set **DisplayMemberPath** attribute to **FirstName**. Observe that the

DataSource attribute is set to **customerBindingSource**.

14. Run the application to display all customer names from Customer table in **C1MultiSelect** control list.

Use MultiSelect Control

MultiSelect allows you to add, remove, and access specific items with minimal code. It also lets you display or hide the check boxes and dropdown button appearing in the control. Learn how they can be implemented.

- **Add an item**
- **Remove an item**
- **Access specific item**
- **Show/Hide check boxes**
- **Show/Hide dropdown button**

Add an item

To add items to the MultiSelect control, use Add method of C1CheckListItemCollection as shown in the following code. For example, the following code adds "Edward" in dropdown list of the MultiSelect control:

C#

```
c1MultiSelect1.Items.Add("Edward");
```

MultiSelect also allows you to add an item at a specific position using Insert method of **C1CheckListItemCollection** and specify an index value for the new item in it. For example, the following code inserts a name, Robert, to the fifth position, adjusting the position of the other items in the dropdown list:

C#

```
c1MultiSelect1.Items.Insert(4, "Robert");
```

Back to Top

Remove an item

MultiSelect lets you delete an item from the list using RemoveAt method of **C1CheckListItemCollection** as shown in the following code. The method takes one argument, index, which specifies the item to remove. For example, the following code deletes sixth entry from the list.

C#

```
c1MultiSelect1.Items.RemoveAt(5);
```

MultiSelect also allows you to delete a selected item from the MultiSelect control using Remove method as shown in the following code:

C#

```
c1MultiSelect1.Items.Remove(c1MultiSelect1.SelectedItem);
```

To remove all the entries from the list, use Clear method of **C1CheckListItemCollection** as shown in the following code:

C#

```
c1MultiSelect1.Items.Clear();
```

Back to Top

Access specific item

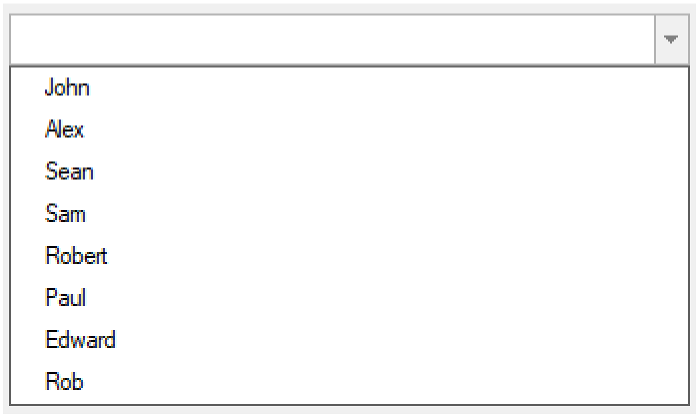
To access a specific item from the MultiSelect control dropdown list, use Items property of C1MultiSelect class and specify the index of that item. Furthermore, you can change the value of an item in the list using Value property of the C1CheckListItem class. For example, to access and change the value of third item from the list, write the following line of code:

C#
```
c1MultiSelect1.Items[2].Value = "Jake";
```

Back to Top

Show/Hide check boxes

By default, list of items in the MultiSelect control are displayed with check boxes. However, you can disable the default style by setting ShowCheckBoxes property to **false**.



To hide the check boxes from the list, use the following code:

C#
```
c1MultiSelect1.ShowCheckBoxes = false;
```

Back to Top

Show/Hide dropdown button

MultiSelect displays the dropdown button to show the list of available items. However, you can hide the dropdown button in the control by setting ShowDropDownButton property to **false**.

To hide the drop down button, use the following code:

C#
```
c1MultiSelect1.ShowDropDownButton = false;
```

Back to Top

Data Binding

MultiSelect provides data binding support that lets you populate data in the control. It allows you to bind the control to complex objects and data sources. To bind MultiSelect to a data source, you need to access the BindingInfo property of the control. This property contains information about data source and other data binding options. The **BindingInfo** property contains all the important attributes for binding which are DataMember, DataSource, CheckedMemberPath, DisabledMemberPath, DisplayMemberPath, TagsDataSource, and TagsMemberPath. To bind the MultiSelect control, you need to assign the DisplayMemberPath property of object or data source.

You can bind MultiSelect using any of the following ways:

Bind MultiSelect to a data source
    Learn how to bind MultiSelect to a data source in code.
Bind MultiSelect to object collection
    Learn how to bind MultiSelect to object collection in code.
MultiSelect in unbound mode
    Learn how to implement MultiSelect in unbound mode through code.

Bind MultiSelect to a Data Source

To bind MultiSelect to a data source, follow these steps:

1. Create a connection string and fetch data from a database to a data set.

C#

```
static string GetConnectionString()
{
      string conn = @"Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=C:\Users\GPCTAdmin\Documents\ComponentOne Samples\Common\C1NWind.mdb;";
      return string.Format(conn);
}
DataTable GetDataSource(string connectionString)
{
      // 接続文字列を設定します。
      string conn = GetConnectionString();

      // SQL文を設定します。
      string rs = connectionString;

     // データをDataSetに取得します。
     OleDbDataAdapter da = new OleDbDataAdapter(rs, conn);
     DataSet ds = new DataSet();
     da.Fill(ds);

     // データテーブルを返します。
     return ds.Tables[0];
}
```

2. Set the DataSource and DisplayMemberPath properties of the MultiSelect control.

C#

```
private void Form1_Load(object sender, EventArgs e)
{
       c1MultiSelect1.BindingInfo.DataSource = GetDataSource("Select * from
Employees ");
       c1MultiSelect1.BindingInfo.DisplayMemberPath = "FirstName";
}
```

Bind MultiSelect to Object Collection

You can bind MultiSelect to a list of complex data objects which can have multiple properties. To bind the control to a list of data objects, follow these steps:

1. Create a class named Customer using the following code.

C#

```
public class Customer
    {
        string name;
        string customerID;
        string mobile;
        string email;
        public Customer(string _name, string _custId, string _mobile, string
_email)
        {
            this.name = _name;
            this.customerID = _custId;
            this.mobile = _mobile;
            this.email = _email;
        }

        public string Name
        {
            get
            {
                return name;
            }
        }
        public string CustomerID
        {
            get
            {
                return customerID;
            }
        }
        public string Mobile
        {
            get
            {
                return mobile;
            }
        }
        public string Email
        {
            get
            {
                return email;
            }
        }
    }
```

2. Add objects of Customer class to a BindingList and set C1MultiSelect's DataSource and DisplayMemberPath properties at Form_Load event.

C#

```
private void Form1_Load(object sender, EventArgs e)
   {
    BindingList<Customer> customers = new BindingList<Customer>();
    customers.Add(new Customer("John", "C001", "8888832141","john@gmail.com"));
    customers.Add(new Customer("Alex", "C002", "8888832142", "@gmail.com"));
    customers.Add(new Customer("Shawn", "C003", "8888832143",
"shawn@gmail.com"));
    customers.Add(new Customer("Sam", "C004", "8888832144", "sam@gmail.com"));
    customers.Add(new Customer("Neo", "C005", "8888832145", "neo@gmail.com"));
    customers.Add(new Customer("Paul", "C006", "8888832146", "paul@gmail.com"));

     c1MultiSelect1.BindingInfo.DataSource = customers;
     c1MultiSelect1.BindingInfo.DisplayMemberPath ="Name";

   }
```

MultiSelect in Unbound Mode

In unbound mode, you can populate the control with data by generating its content. For generating content, you need to add items either at design time, or at run time.

The following code populates C1MultiSelect control with customer names when the form loads at run time.

C#

```
private void Form1_Load(object sender, EventArgs e)
{
    c1MultiSelect1.Items.Insert(0,new C1.Win.Input.C1CheckListItem("John"));
    c1MultiSelect1.Items.Add(new C1.Win.Input.C1CheckListItem("Alex"));
    c1MultiSelect1.Items.Add(new C1.Win.Input.C1CheckListItem("Sean"));
}
```

Features

This section comprises all the features available in MultiSelect control.

Appearance
    Learn how to customize the appearance of the control elements through code.
Display mode
    Learn how to change the display modes through code.
Multi-Selection mode
    Learn how to change the selection modes through code.
Selection
    Learn how to implement selection related features through code.
Tag Appearance
    Learn how to display tags in different ways through code.
Tag Wrap
    Learn how to wrap the header items through code.

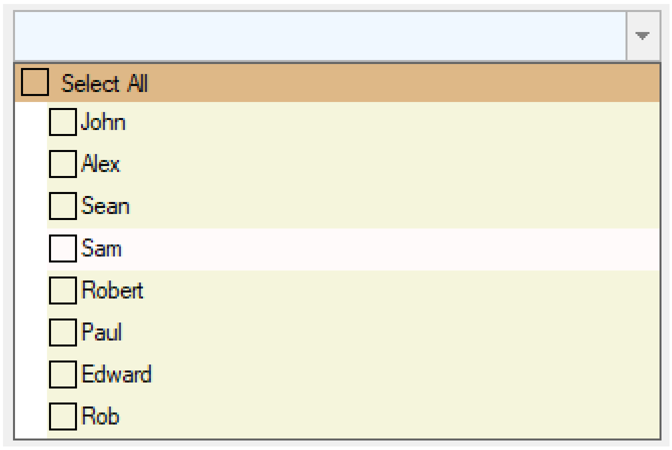
Appearance

MultiSelect allows you to customize the appearance of all the individual elements of the control and manage its overall appearance.

- **Apply Styles to CheckList**
- **Apply Styles to DropDownButton**
- **Apply Styles to TagEditor**
- **Apply Theme**

Apply Styles to CheckList

The following image shows styles applied to the CheckList element of MultiSelect.



To apply style to CheckList element of MultiSelect, use the following code. You can also set other attributes of Styles.CheckList property to further customize its appearance.

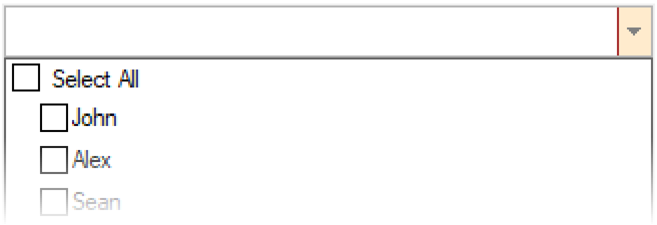
C#

```
c1MultiSelect1.Styles.CheckList.Hot.BackColor = Color.Snow;
c1MultiSelect1.BackColor = Color.AliceBlue;
c1MultiSelect1.Styles.CheckList.Default.BackColor = Color.Beige;
c1MultiSelect1.Styles.CheckList.Header.BackColor = Color.BurlyWood;
```

Back to Top

Apply Styles to DropDownButton

The following image shows styles applied to the DropDownButton element of MultiSelect.



To customize the appearance of DropDownButton element of C1MultiSelect, use the following code. You can also set other attributes of Styles.DropDownButton property for further customization.

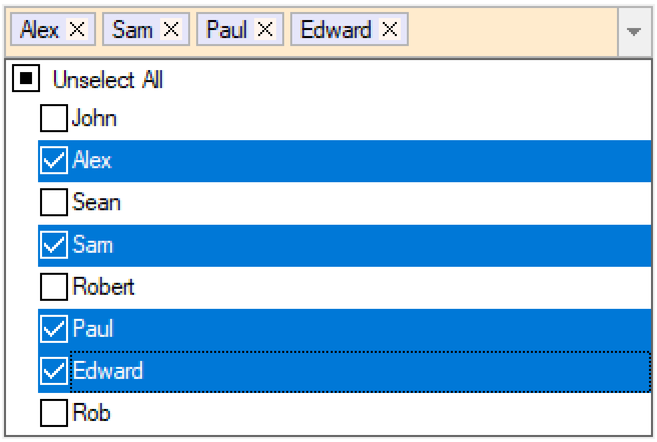
C#

```
c1MultiSelect1.Styles.DropDownButton.BackColor = Color.BlanchedAlmond;
c1MultiSelect1.Styles.DropDownButton.BorderColor = Color.Brown;
```

Back to Top

Apply Styles to TagEditor

The following image shows styles applied to the TagEditor element of MultiSelect.



TagEditor element styles can be accessed via the Styles.TagEditor.Common property. The styles used to paint the tags can be accessed via the Styles.TagEditor.Tag property and Styles.TagEditor.RemoveButton can be used to access styles to paint Remove button in a tag. To change the appearance of tag editor, tags and Remove button, use the following code. You can also set other attributes of Styles.TagEditor.Common, Styles.TagEditor.Tag and Styles.TagEditor.RemoveButton properties for further customization.

C#

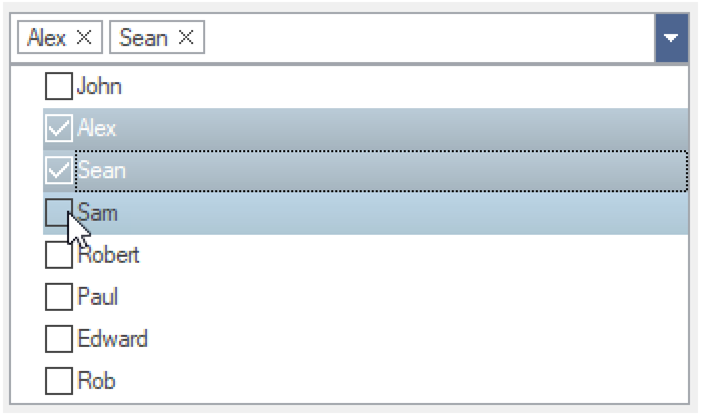
```
c1MultiSelect1.Styles.TagEditor.Common.BackColor = Color.BlanchedAlmond;
c1MultiSelect1.Styles.TagEditor.RemoveButton.BackColor = Color.SeaShell;
c1MultiSelect1.Styles.TagEditor.Tag.BackColor = Color.Lavender;
```

Back to Top

Apply Theme

You can customize the appearance of the MultiSelect control using built-in themes or by designing your own themes. To apply themes to the control, you can use **C1ThemeController** which provides many built-in themes and an easy to use theme designer to create your own themes. For more information on themes, see Theming.

The following image shows the MultiSelect control with MacBlue theme applied to it.

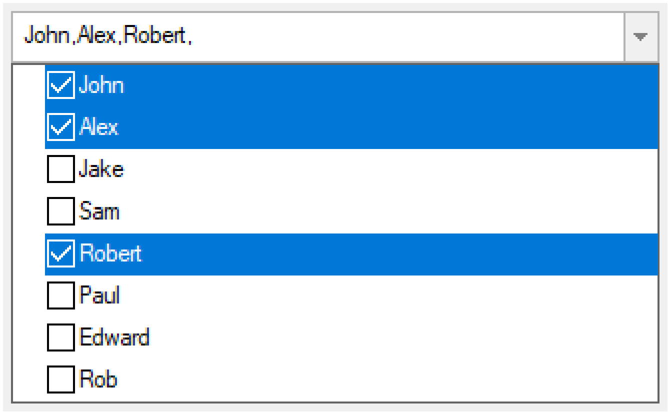


Back to Top

Display Mode

MultiSelect allows you to choose the appearance of items in header between text or tags. In text mode, the text is displayed as strings separated by a separator character. On the other hand, in tag mode, the tags appear like labels separated by a space. You can choose how the items appear in the TagEditor element using DisplayMode property that accepts the values from DisplayMode enumeration.

The following image shows the selected items displayed as text in the header.



To display the selected items as text in the header, use the following code:

C#

```
c1MultiSelect1.DisplayMode = C1.Win.Input.DisplayMode.Text;
```

By default, the MultiSelect control uses comma (,) as separator character to separate the items in header. However, you can specify a separator character of your choice using Separator property.

C#

```
c1MultiSelect1.Separator = "/";
```

Multi-Selection Mode

MultiSelect provides SelectionMode property to determine whether you can select one or more than one item from the header. This property lets you choose between the following selection modes through SelectionMode enumeration:

- **Single**: Allows you to select only one item at a time.
- **Multiple**: Allows you to select multiple items without holding down a modifier key.
- **Extended**: Allows you to select multiple consecutive items while holding down the corresponding modifier key.

To set the selection mode to multiple, use the following code:

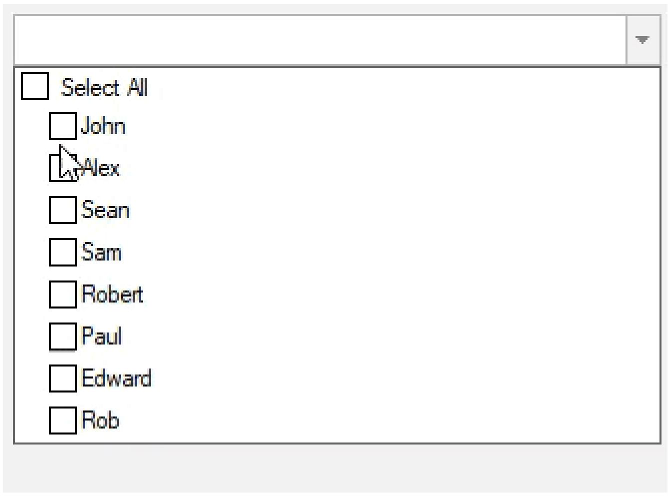
C#

```
c1MultiSelect1.SelectionMode = C1.Win.Input.SelectionMode.Multiple;
```

Selection

MultiSelect provides you with an option to select/deselect all items in the list with single selection. To enable this option, you need to set the ShowSelectAll property to true. On setting this property to true, the Select All check box appears on top of the list of items. When you deselect the Select All check box, all items in the list are deselected.

The following GIF shows the Select All and Unselect All check box appear in the MultiSelect control.



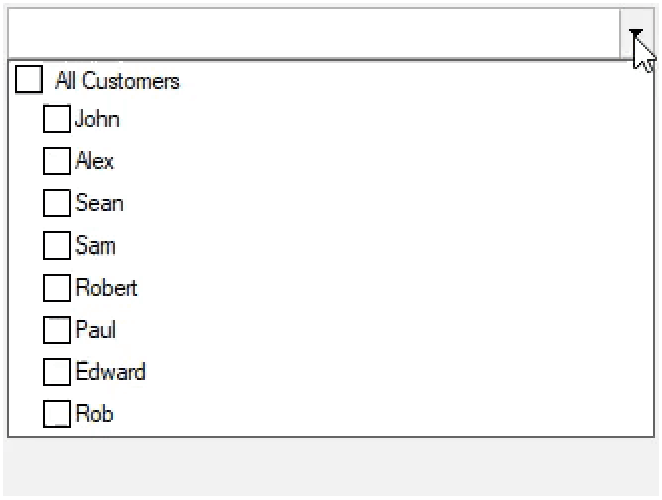
To enables Select All option in the MultiSelect control, use the following code:

C#

```
c1MultiSelect1.ShowSelectAll = true;
```

In addition, MultiSelect also allows you to change the caption of Select All option using SelectAllCaption property and caption of Unselect All option using UnselectAllCaption property of C1MultiSelect class.

The following GIF shows the changed caption of Select All and Unselect All options in the MultiSelect control.



To change the caption, use the following code:

C#

```
c1MultiSelect1.SelectAllCaption = "All Customers";
c1MultiSelect1.UnselectAllCaption = "Uncheck All";
```

Tag Appearance

MultiSelect has customizable header which, by default, displays all the items selected from the list. However, you can control this default behavior to adjust the maximum number of selected items to display in the header using MaxHeaderItems property of the C1MultiSelect class. For example, when you set the value of **MaxHeaderItems** property to 3, as soon as you select the fourth item, the header starts showing the total count of selected items as "4 items selected".

The following GIF shows how the items in the header appear on setting the MaxHeaderItems property.

To display maximum three items in the header, use the following code:

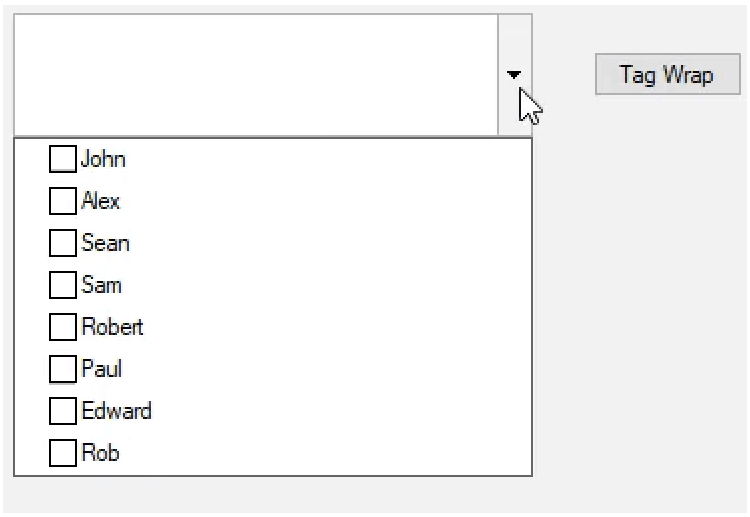
C#

```
c1MultiSelect1.MaxHeaderItems = 3;
```

Tag Wrap

MultiSelect provides a way to wrap the items within the control header. This can be achieved using TagWrap property of C1MultiSelect class.

The following GIF shows how the items in the header get wrapped on setting the TagWrap property.



To wrap items in header, use the following code:

C#

```
c1MultiSelect1.TagWrap = true;
```