



Contents

- [1 What is Compound Controls??](#)
- [2 How to create Compound control?](#)
- [3 How to gets an indexed component of a compound control??](#)
- [4 How to add two or more controls ?](#)

What is Compound Controls??

Compound controls are controls that contain other controls. Other controls is either window-owning controls or non-window-owning controls.

Window-owning controls includes.

Window-owning controls have the same size and position as a window in the display. Each window has a one-to-one relationship with the control that covers it, and shares its behavior with that control.

Examples of window-owning controls include:

1. top-level control in Traditional Symbian OS architecture
2. the subpanes in the status pane
3. pop-up windows, when a sense of layering is required

Non-window-owning controls

Non-window-owning controls typically cover only part of a window on the display, and must be contained in window-owning controls. They are faster and require fewer resources than window-owning controls.

Examples of non-window-owning controls include:

1. Command Buttons (Like [CEikTextButton](#))
2. Edit Windows (Like [CEikEdwin](#), [CEikFloatingPointEditor](#), [CAknNumericSecretEditor](#), etc.)
3. Labels (Like [CEikLabel](#))

How to create Compound control?

Construct the control in the parent control(where you want to implement ur control), which is the control that owns the window in which the control appears(for most cases it will be container/appview class of your application). The control should be constructed in the constructor for the window. Lets take example of implementing Label control as follows.

```
void CMyViewContainer::ConstructL(const TRect& aRect)
{
    CreateWindowL();

    //Creates a CEikLabel object, which is a class that supports the display of text in the parent
    iLabel = new (ELeave) CEikLabel;

    //Assigns the non-window-owning control to the window-owning control
```

How_to_implement_Compound_Controls?

```
iLabel->SetContainerWindowL( *this );

//Sets a text for the label.
iLabel->SetTextL( _L("Testing Compound controls in symbian.") );

//In affect calls the SizeChange() of the Control
SetRect(aRect);

//Sets the control as ready to be drawn.
ActivateL();

}
```

Add a method to the owning control that returns the number of controls in the compound control. An example of an implementation is as follows:

```
//We need to overrides CCoeControl's CountComponentControls to return the number of controls in t
TInt CMyViewAppContainer::CountComponentControls() const
{
    //This method will return number of controls inside this compound control. so we only one contro

        if(iLabel)
return 1;
        else
return 0;

}
```

How to gets an indexed component of a compound control??

Within a compound control, each component control is identified by an index, where the index depends on the order the controls were added: the first is given an index of 0, the next an index of 1, and so on. There are 2 ways to implement it in compound control. One way is to override CCoeControl's ComponentControl function. The other way is to use the CCoeControlArray functionality. All child controls should be accessible by CCoeControl::ComponentControl at any time, regardless of whether they are visible or not. The visibility of a control should be adjusted using the CCoeControl::MakeVisible method.

```
CCoeControl* CMyViewAppContainer::ComponentControl(TInt aIndex) const
{
    switch ( aIndex )
    {
        case 0:
            return iLabel; // return a pointer to the iLabel
        default:
            return NULL;
    }
}
```

The compound control usually owns the child controls and therefore it is responsible of their construction and destruction. The compound control should also set the positions and sizes of its child controls; it must ensure that all child controls are inside the compound control rectangle and visible child rectangles do not overlap each other.

How to add two or more controls ?

For adding each control, we need to modify `ComponentControl()` and `CountComponentControls()` method of our class. Lets take an example for adding 3 labels in a class.

```
void CMyViewContainer::ConstructL(const TRect& aRect)
{
    CreateWindowL();

    //Creates a CEikLabel object, which is a class that supports the display of text in the parent
    iLabel = new (ELeave) CEikLabel;
    //Assigns the non-window-owning control to the window-owning control
    iLabel->SetContainerWindowL( *this );
    //Sets a text for the label.
    iLabel->SetTextL( _L("Testing Compound controls in symbian.") );

    //Creates second label.
    iLabel2 = new (ELeave) CEikLabel;
    //Assigns the non-window-owning control to the window-owning control
    iLabel2->SetContainerWindowL( *this );
    //Sets a text for the label.
    iLabel2->SetTextL( _L("label2") );

    //Creates third label.
    iLabel3 = new (ELeave) CEikLabel;
    //Assigns the non-window-owning control to the window-owning control
    iLabel3->SetContainerWindowL( *this );
    //Sets a text for the label.
    iLabel3->SetTextL( _L("label3") );

    //In affect calls the SizeChange() of the Control
    SetRect(aRect);

    //Sets the control as ready to be drawn.
    ActivateL();
}
}
```

Modify `CountComponentControls()` as.

```
TInt CMyViewAppContainer::CountComponentControls() const
{
    return 3; // as we have 3 controls in a class.
}
}
```

Modify `ComponentControl()` as.

```
CCoeControl* CMyViewAppContainer::ComponentControl(TInt aIndex) const
{
    switch ( aIndex )
    {
        case 0:
            return iLabel; // return a pointer to the iLabel
        case 1:

```

How_to_implement_Compound_Controls?

```
        return iLabel1; // return a pointer to the iLabel1
case 2:
        return iLabel2; // return a pointer to the iLabel2
default:
        return NULL;
    }
}
```